A Thesis

Entitled

Beamlet Transform Based Technique for Pavement Image Processing and Classification

By

Liang Ying

Submitted as partial fulfillment of the requirements for

The Master of Science Degree in Electrical Engineering

_____

Advisor: Dr. Ezzatollah Salari

_____

College of Graduate Studies

The University of Toledo

December 2009

An Abstract of

Beamlet Transform Based Technique for Pavement Image Processing and Classification

Liang Ying

Submitted as partial fulfillment of the requirements for

The Master of Science Degree in

Electrical Engineering

The University of Toledo

December 2009

The goal of this thesis is to develop and implement an algorithm to automatically detect and classify cracks from pavement images using digital image processing techniques. The proposed method uses a pavement distress image enhancement algorithm to correct the non-uniform background illumination by calculating the multiplicative factors that eliminate the background lighting variations. To extract the linear features such as surface cracks from the pavement images, the image is partitioned into small windows and a beamlet transform based-algorithm is applied. The crack segments are then linked together and classified into four types, vertical, horizontal, transversal, and block types. Simulation results show the method is effective and robust in the extraction of cracks on a variety of pavement images.

Acknowledgements

I would like to thank Dr. Salari, my advisor, who gave me the opportunity to do this research work at UT. I really appreciate the help and guidance provided by him, who patiently answered my various questions about the work, carefully read the thesis, corrected errors, pointed out weak points, and suggested additional topics. In addition, I would like to thank Dr. Jamali and Dr. Miller for being my defense committee. I also want to thank all my friends at UT, who gave me company as well as help both on my study and on my life in US. Hope you all have a brighter future! Also, I want to dedicate my gratitude to my family behind me. Though I didn't live with you in the past several years, my heart is always there with you.

I would like to thank the Michigan Ohio University Transportation Center (MIOH-UTC), U.S. Department of Transportation, for their support.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Background

Statistics data published by the Federal Highway Administration indicate that maintenance and rehabilitation of highway pavements in the United States require over $17 billion a year [1]. Conventional visual and manual pavement distress analysis approaches, where the inspectors traverse the roads and measure the distressed objects when they are found, are very costly, time-consuming, dangerous, labor-intensive, tedious, and subjective. They also have a high degree of variability which means they are unable to provide meaningful quantitative information and almost always lead to inconsistencies in distress detail over space and across evaluations [3].

With the development of technology of electronic sensors and computer systems, various systems, named Pavement Imaging Systems (PIS), have been developed. They are used for automated pavement data collection and distress detection.

Figure 1-1 and 1-2 show the global view of the pavement distress detection and classification system. There are two major steps in automated pavement evaluation: the distress data acquisition and the distress data analysis. The equipment needed for image acquisition is comprised of cameras, lenses, and computer hardware for the digitization. This step includes the recording of the pavement distress data by a moving vehicle. The

distress data analysis includes the digital image processing and the image interpretation [2].

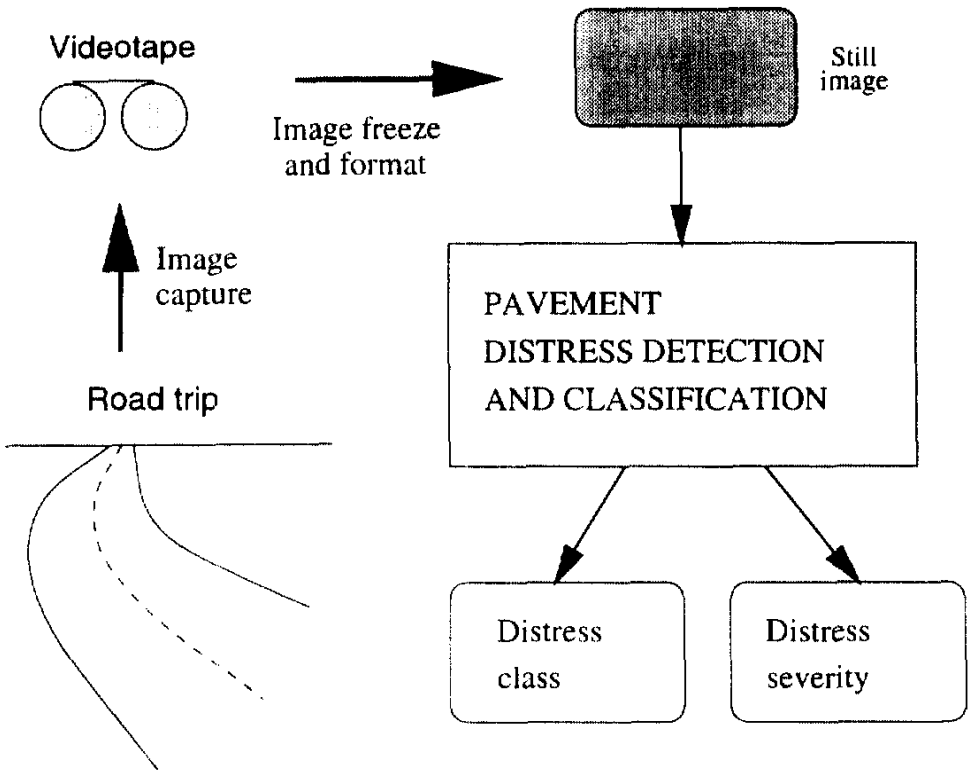


Fig. 1-1 Block diagram of pavement distress classification system [3]

Fig. 1-2 The automatic pavement inspection system [19]

1.2 Introduction of Pavement Image Processing

After pavement images have been obtained, the pavement images are processed to extract the crack information. Figure 1-3 shows four steps that are needed for the pavement distress detection.

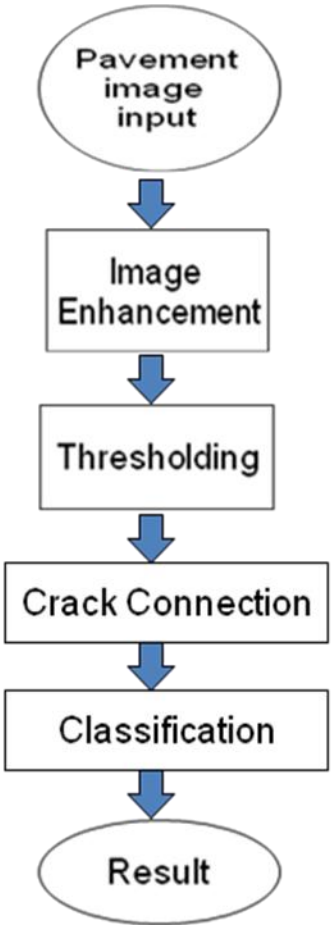


Fig. 1-3 Steps of pavement image processing

- **Image Enhancement**

  Pavement images are composed of background, noise, and cracks. The noises on the image, objects on the road, the pavement patterns, and the non-uniform background cause difficulties for crack detection and even fail the threshold process. In order to recognize distress with fidelity on the road surfaces,

many algorithms have been developed to eliminate noises and normalize the background.

- **Thresholding**

  Thresholding is a technique used to separate objects from the background. Since cracks are always darker than the surroundings, the threshold value should be a relatively low value. Fuzzy thresholding is a thresholding technique that is implemented by defining a fuzzy function and projects the pavement image into a fuzzy crack domain between 0 and 1.

- **Crack Connection**

  The binary images extracted from pavement images are usually noisy. The cracks in the binary images are discontinuous. In order to get the dimensional information of cracks, the discrete crack points need to be connected. Also, by setting a threshold to the crack size, noises can be eliminated.

- **Classification**

  According to length, width, and orientations, cracks are classified into four categories, horizontal, vertical, diagonal, and block. The different types of pavement distress are shown in figure 1-4.

(a). vertical cracks

(b). horizontal cracks

(c). diagonal cracks

(d). block cracks

Fig. 1-4 Different types of cracks

1.3 Outline of Thesis

The remainder of the thesis is organized as follows. Chapter 2 is literature review. Chapter 3 proposes a non-uniform background improvement algorithm. In Chapter 4, a beamlet transform based technique is introduced and the process for extension check is explained. In Chapter 5, several test cases for different types of cracks are processed and the results are presented. Finally, conclusions and future works are presented in Chapter 6.

Chapter 2

Literature Review

The need for a fast, objective, and relatively inexpensive automated road inspection system is highly desirable. Video technologies and image processing techniques used as tools for inspection, monitoring, and diagnosis have long been adopted by various fields, most notably by medicine and remote sensing. However, its use in transportation is still not widespread. The reason is that the amount of data to be processed could be very large and repetitive, while the accuracy requirements may not be as strict as in, say, medical diagnosis. Conventional visual and manual pavement distress analysis approaches in which the inspectors traverse the roads and stop and measure the distress objects are very costly, time-consuming, labor-intensive, and unstable. Therefore, automated analysis and pattern recognition is highly desirable for pavement inspection. In general, the desired approach is to capture pavement images using video cameras mounted on a moving vehicle and then to use a computer to recognize and quantify the pavement distresses from these video images.

There has been a significant amount of research during the past two decades in developing automated pavement inspection. Xu and Huang developed a customized image processing algorithm for pavement cracking inspection in which an image is divided into small cells and a cell is classified as either crack or non-crack seeds based on its local characteristics [1]. After verification, a cluster of seeds is identified as a real crack. Maser

[2] used histogram equalization to improve the contrast of the images, and proposed a threshold-based segmentation. Li [3] used Sobel edge detectors and modified the automatic threshold determination method suggested by Kittler and Illingworth, in order to connect the crack segments to form a continuous cluster of object pixels. They relied on the assumption, that noise clusters had a perimeter of less than twenty pixels.

Koutsopoulos et al. [4] proposed a lighting variations compensation method by subtracting an average of a few non-distress images from the same series. For segmentation, instead of using ordinary binary segmentation that assigns a value of one to object pixels and a value of zero to background pixels, resulting in a binary image, a different approach is suggested, and it assigns values from 0 to 3 to each pixel, based on its probability of being an object pixel. Background pixels are drawn from the Gaussian distribution. Object pixels are drawn from a similar distribution with a lower mean and a higher variance. The threshold that meets various criteria can be obtained from these two distributions. Chou et al. [5] used moment invariants from different types of distress to obtain the features, and then used a back propagation neural network to classify the features. Georgopoulos et al. [6] proposed a method in which the distress can be represented by a set of vectors, approximating the cracks composing the distress. Then the direction vectors are grouped into two categories, horizontal and vertical, and the cracks are classified based on their presence.

Cheng et al. [7] proposed an approach based on fuzzy set theory. This method compares the darkness of the pixels and their neighbors by deciding the brightness membership function for gray levels in the difference image. Second, the fuzzified image is mapped into the crack domain by finding the crack membership values of the pixels.

Third, the connectivity of the darker pixels is checked to eliminate the pixels lacking in connectivity. Finally, an image projection algorithm is employed to classify the cracks.

The use of a discrete wavelet transform (DWT) has also been explored for crack image analysis. Using a fast wavelet transform, a pavement image can be decomposed into different frequency sub-bands. The magnitude of the wavelet coefficients represents the level of distress [8-10]. Javidi et al. [11] defined two wavelets which are, respectively, the partial derivatives along x and y of a two-dimensional smoothing cubic spline wavelet function. By measuring the evolution across scales of the wavelet transform maxima, the background noise can be separated from the important cracks. Then the crack map images are projected into the Hough transform domain to quantify the number of dominant cracks in a given image.

After cracks are segmented from the background, there is much work that needs to be done in order to keep the crack connectivity and classify the cracks. For example, H. D. Cheng and M. Miyojim [3] used a skeleton structure to check the connectivity of cracks. However, checking and connecting cracks pixel by pixel is a time-consuming work, especially with large size images.

In this thesis, an image enhancement algorithm is introduced, and then a method based on the beamlet transform is proposed to extract and classify the crack features from the images of the pavement. After cracks are segmented from the background, there is much work that needs to be done in order to keep the crack connectivity and classify the cracks. For example, Cheng, H. D., and Miyojim, M.[12] used a skeleton structure to check the connectivity of cracks. However, checking and connecting cracks pixel by pixel is a time-consuming work, especially with big size images.

First an image enhancement algorithm is introduced. The algorithm is developed based on Cheng H. D.'s proposed multiply factor method. Then the beamlet transform is introduced. Beamlet transform is proposed by Donoho, D. L. and Huo, Xiaoming [18]. Note that, the wavelet transforms provide localized information at multi-resolution scales for fixed regions of space; however, beamlets transform is specifically designed for localized multi-scale dyadically-organized line segments. A method based on the beamlet transform is proposed to extract and classify the crack features from the images of the pavement. Beamlet transform, is efficient, robust to noise, and easier to extract linear features for the cracks. The extracted pavement cracks are then classified into four types: vertical, horizontal, transversal, and block.

Chapter 3

Non-uniform Background Improvement

An obstacle to automatic detection and pavement distress classification is that road pavement images are usually obtained under non-uniform distributed lighting conditions. In order to recognize distress patterns with fidelity on the road surfaces, it is necessary to convert all source images to a standardized background lighting condition.

3.1 Introduction

In general, the product of illumination and surface reflectance determines the pixel intensity values in an image. For the automatic pavement crack detection system, due to non-uniform lighting conditions, and the pavement's reflectance, the background usually has different intensities in different areas. Consider a pixel P in a pavement image. Its intensity $I(p)$ can be understood as composed of,

(1) High-amplitude, low-frequency non-uniform background component $I_b(p)$;

(2) High-amplitude, and high-frequency pavement distress or non-distress irregularities component $I_c(p)$, and;

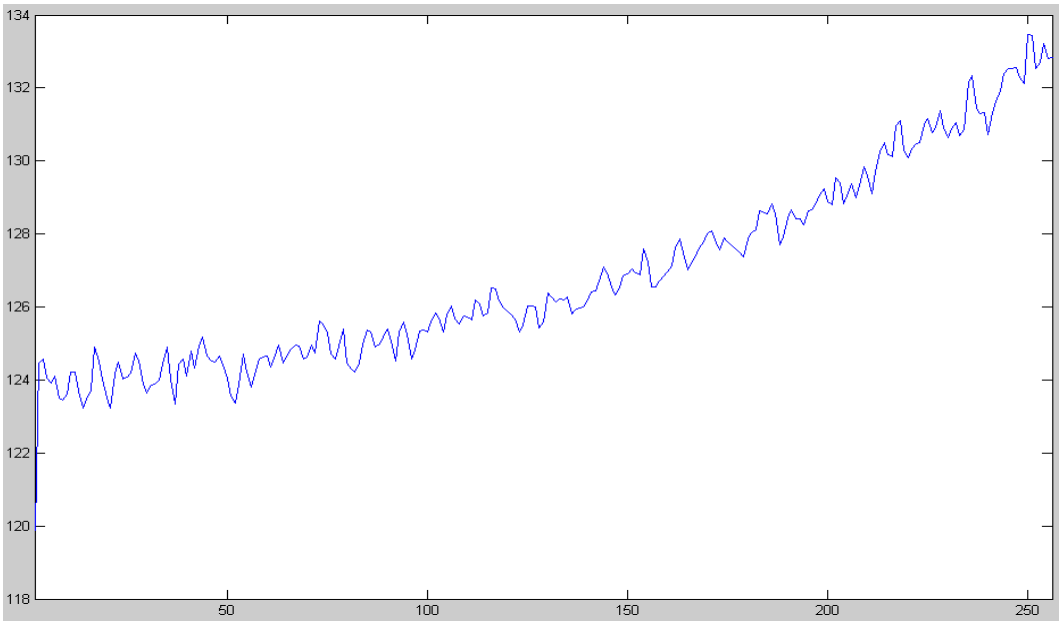
(3). A random, low-to-medium amplitude, high-frequency noise component $I_n(p)$, caused by heterogeneous materials and granularity.

The formula is shown in equation 3-1.

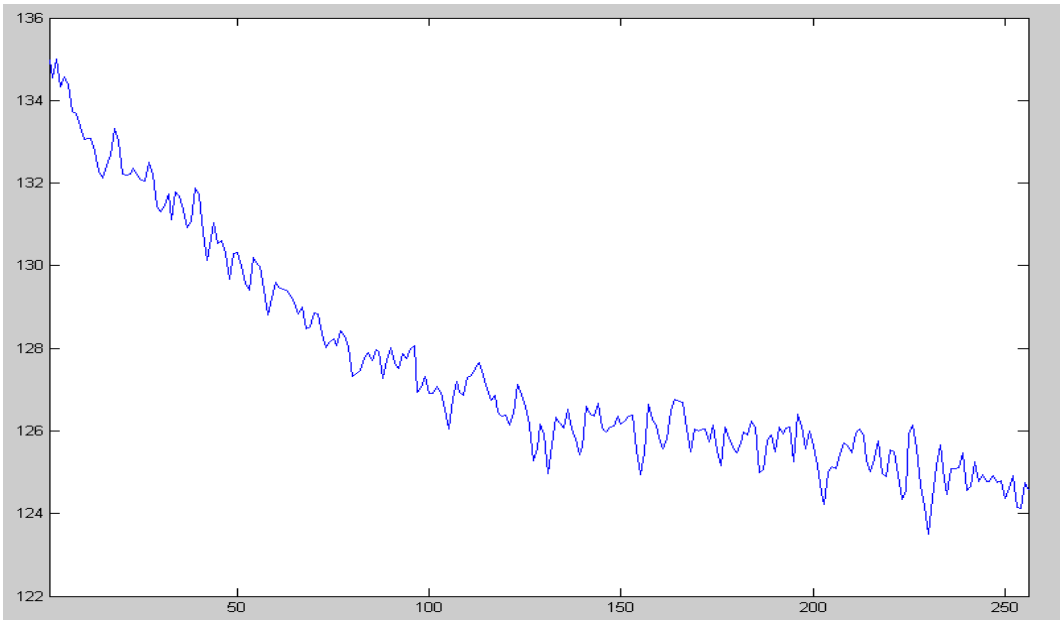
$$I(p) = I_c(p) + I_n(p) + I_b(p) \quad (3\text{-}1)$$

Figure 3-1 shows a pavement image without cracks. Plot (b) and (c) are the average gray level in the x and y directions of plot (a), respectively. It can be seen that in the x direction, the gray level gradually increases from left to right, while in the y direction, it decreases from top to bottom.

(a) Pavement image



(b) Average gray level plot in x direction



(c) Average gray level plot in y direction

Fig. 3-1 Pavement image sample without cracks

The brightness information is the most important part of crack detection. To extract the crack features, the pavement image needs to be thresholded. However, the

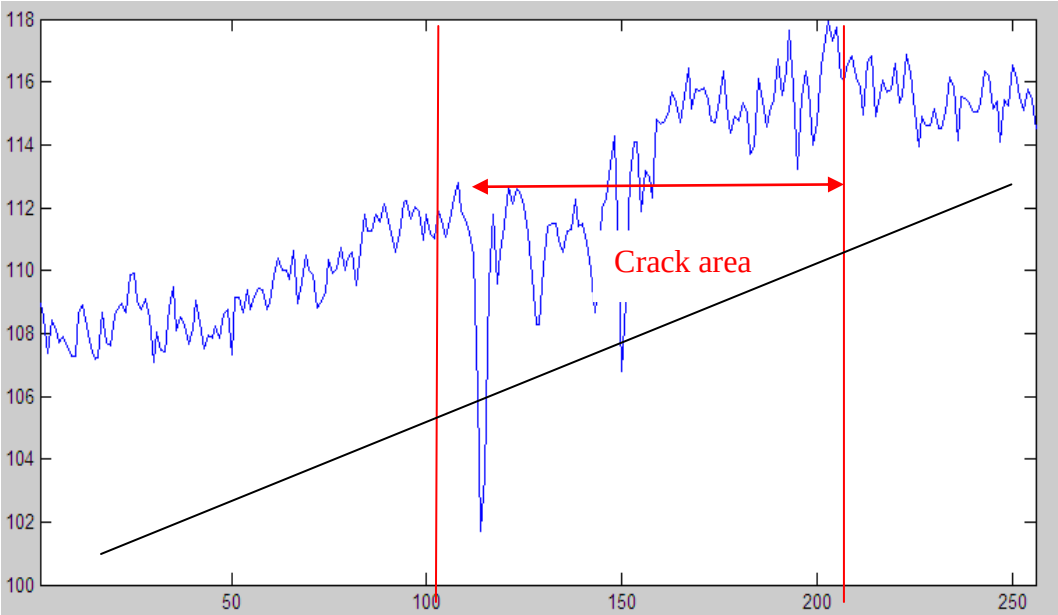
high-amplitude and non-uniform background component $I_b(p)$ may hide the distress component $I_c(p)$. The non-uniform background intensity effects must be eliminated so that the background has a uniform average intensity, while cracks remain at lower intensities.

Figure 3-2 (a) is a pavement image with a vertical crack. The average gray level in the x direction changes at different locations, which means the background illuminant changes gradually. In order to extract the distress information with fidelity from the original images, it is necessary to convert the background component $I_b(p)$ into a constant base intensity B,

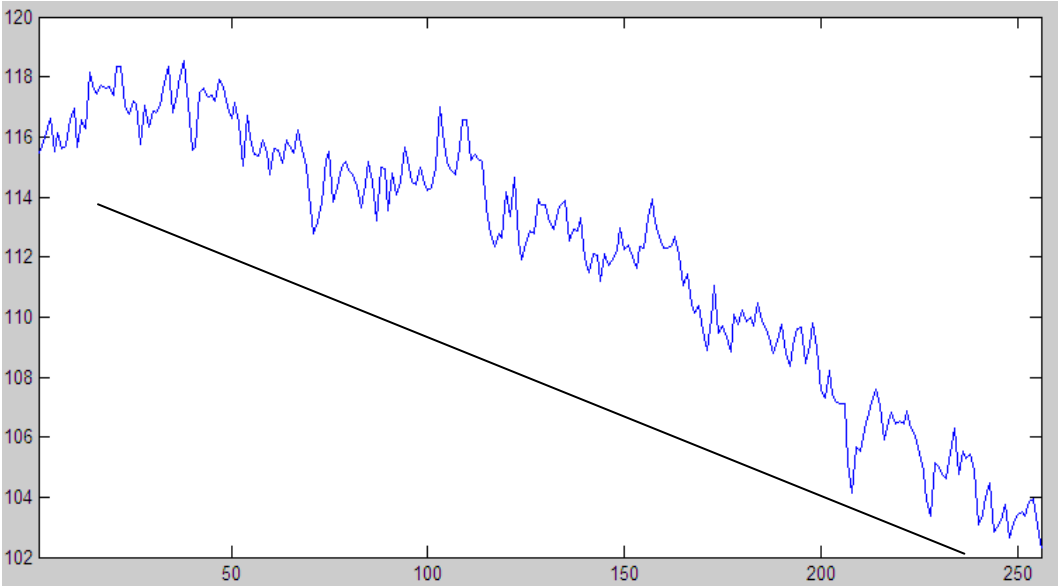
$$I_b'(p) = B \tag{3-2}$$

where B is an arbitrary chosen gray-level value.

(a) Pavement image



(b) Average gray level plot in x direction



(c) Average gray level plot in y direction

Fig 3-2 Pavement image sample with vertical cracks

3.2 Image Enhancement Algorithm

Most algorithms for removing the non-uniform background use statistical properties of the pavement distress images. In 1998, Cheng, H. D. and Miyojim, M. [3] proposed a multiply factor method. First of all, pavement image are divided into rectangle windows, the mean value of each window is checked, and then a multiplier is computed for each window which can transfer the mean value of each window to a target value. This method is based on the assumption that the illumination of the pavement image is smoothly changed. When there is a sudden drop, the window is considered to have cracks, and the mean value of this window is replaced by the average of neighboring windows. However, if there are cracks that cross more than one window, this algorithm doesn't work. In this thesis, based on Cheng, H. D. and Miyojim, M.'s work [3], a new non-uniform background removal method is proposed below.

Pavement images are partitioned into smaller windows. Considering images without cracks or noises, the intensity of the background is considered to be the mean value of the intensity of each window. The background can be made uniform by adjusting the mean value of each window to a target value B. However, for images that have cracks or noises, it is necessary to remove the effect of the cracks and noises. The proposed image improvement process for a non-uniform background includes the following steps:

1) Partition the image into rectangular windows. The size of the window can vary with the size and type of the input images. For example, figure 3-2 (a) is 256 by 256, and it is partitioned into small windows, 16 by 16 each. As shown in figure 3-3.

2) For each window, calculate the mean (G_{mean}), minimum ($G_{\min}$), and the maximum ($G_{\max}$) gray level.

3) For each window, set an upper limit (r_h) and a lower limit (r_l) for which the points with gray levels outside the limits are considered as suspicious points for noise, crack pixels, or other objects on the road. The range $[r_l, r_h]$ is determined by the following equations 3-3 and 3-4:

$$r_h = G_{mean} + (G_{\max} - G_{mean}) * f \qquad (3\text{-}3)$$

$$r_l = G_{mean} - (G_{mean} - G_{\min}) * f \qquad (3\text{-}4)$$

where f is the limiting factor. It can be varied for different images. From experiments, we set the limiting factor to be 60%.

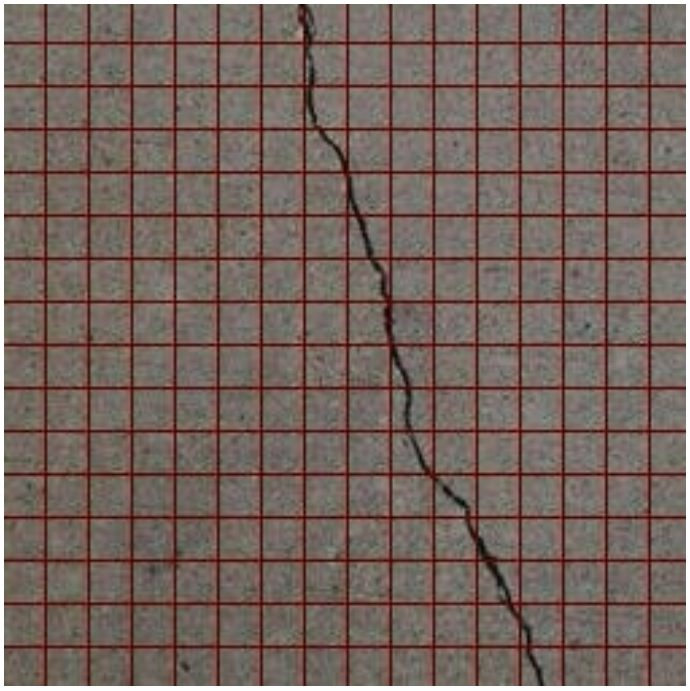


Fig 3-3 Pavement image with horizontal cracks partitioned into small windows

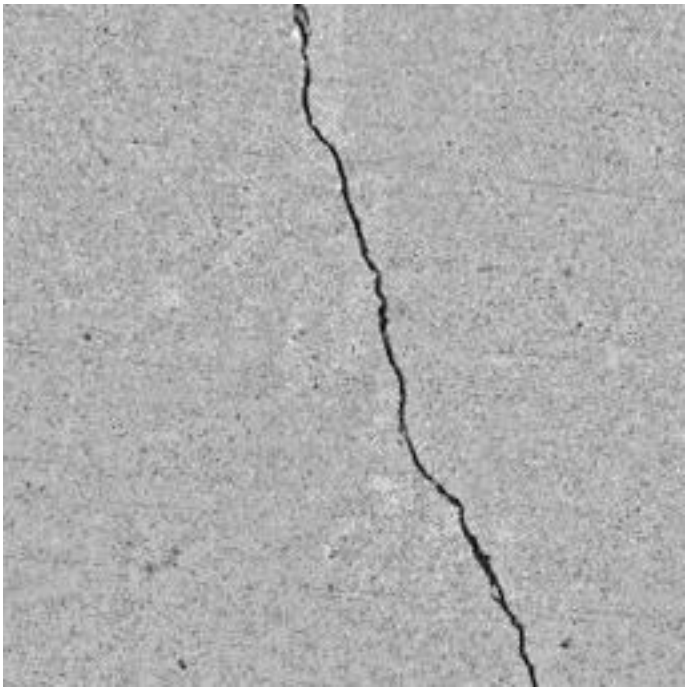
4) With the exemption of the suspicious points, recalculate the mean value of the gray level G'_{mean}. Note that G'_{mean} is the updated mean value of each window, without the factors of noises and crack pixels.

5) The amplitude correction factor is calculated as, $f = B/G_{mean}$, where B is the target background value, and in experiments, the mean value of the original image is used as B. Then the modified picture is obtained by multiplying the factor to each point of the original picture,

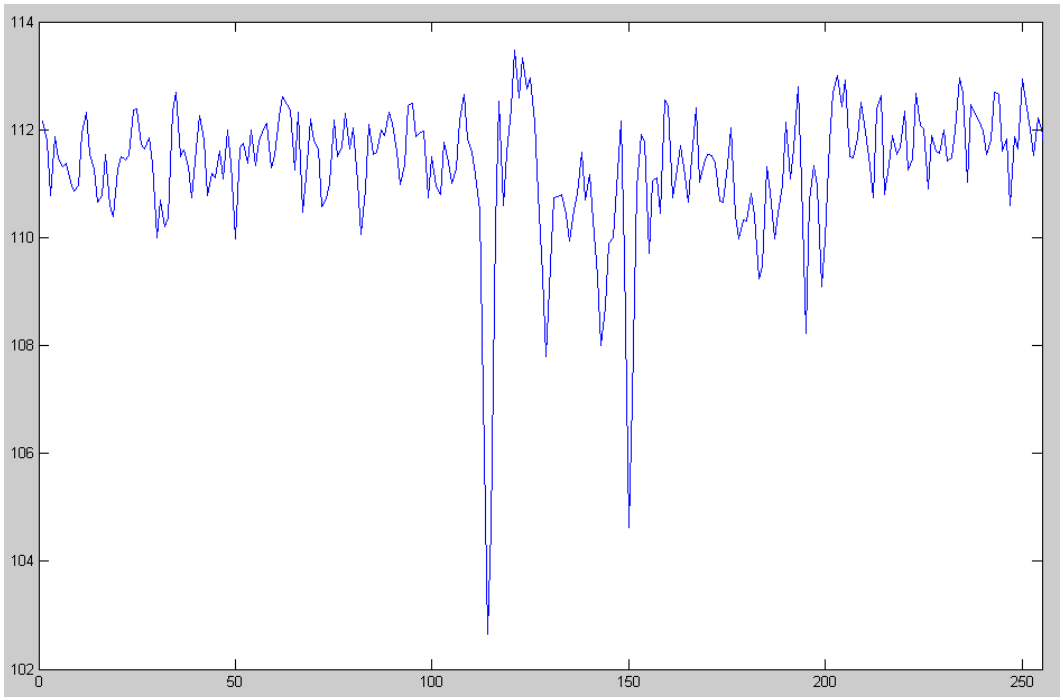
$$I' = I \times f \qquad (3\text{-}5)$$

where, I' is the improved image. The image after enhancement gives a uniform background in both x and y directions. However, if there are many crack pixels in a window, the intensity of the non-crack pixels may increase. Thus, for the pixels whose intensity values are higher than B, their original values either remain unchanged or are replaced by the value B.

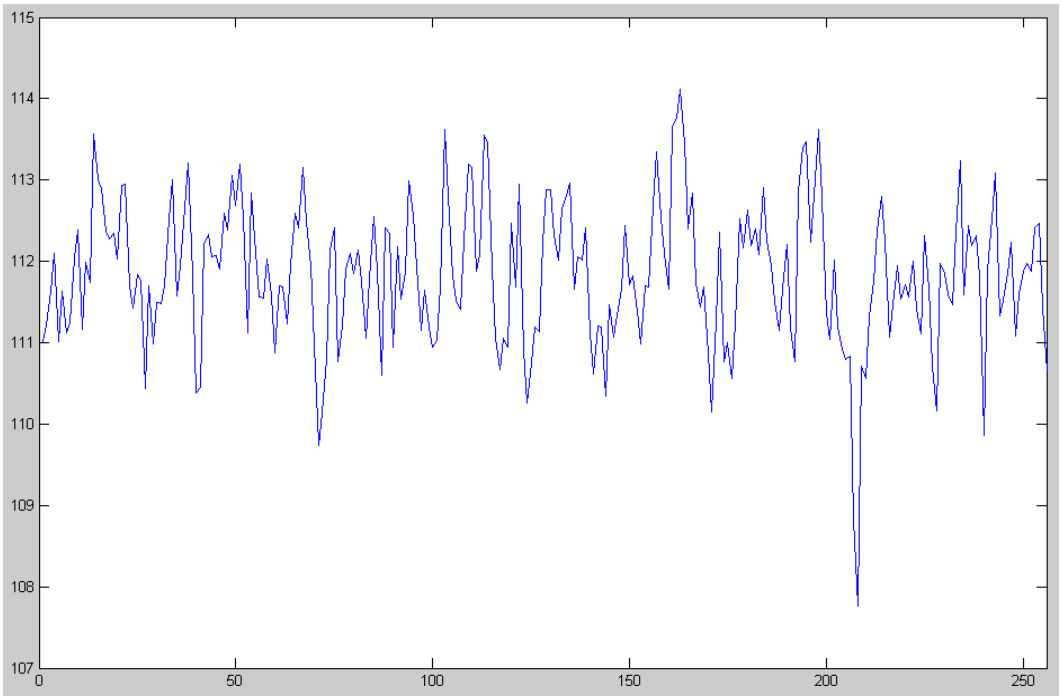
In figure 3-4, the improved image of figure 3-2 is shown. Figure 3-4 (a) is the improved pavement image, plots (b) and (c) show the average image intensity in the x and y direction, respectively. From analyzing plots (b) and (c), it can be easily seen that the illumination of image after improvement has a uniform distribution along both directions.



(a) Improved image



(b) Average gray level plot in x direction of the improved image



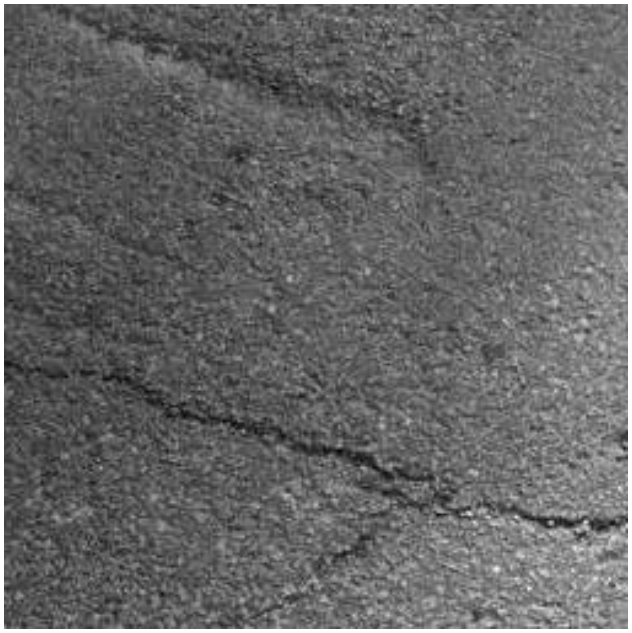
(c) Average gray level plot in y direction of the improved image

Fig 3-4 Improved pavement image from figure 3-1.

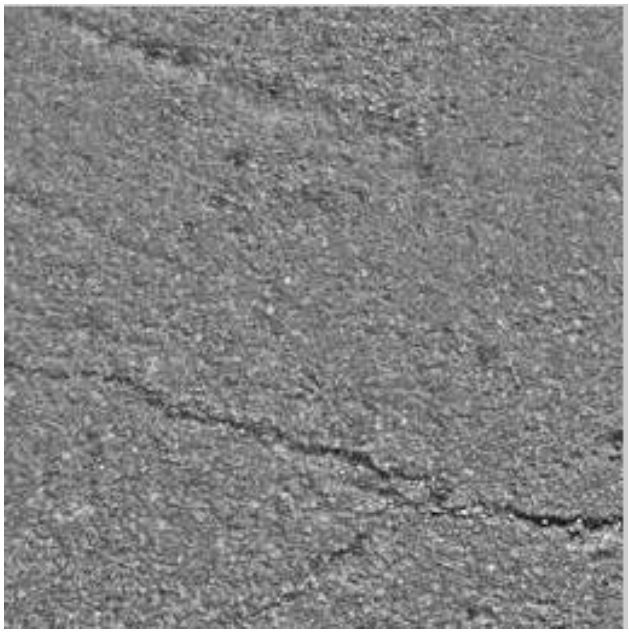
3.3 Enhancement Examples

In this section, several examples of background enhancement are presented.

Figure 3-5 (a) shows a pavement image with a non-uniform background. The intensity in the right bottom corner is much higher than that of the rest area of the image. There are two horizontal cracks in the image. The results of non-uniform background removal are shown in figure 3-5 (b).



(a) Original Image



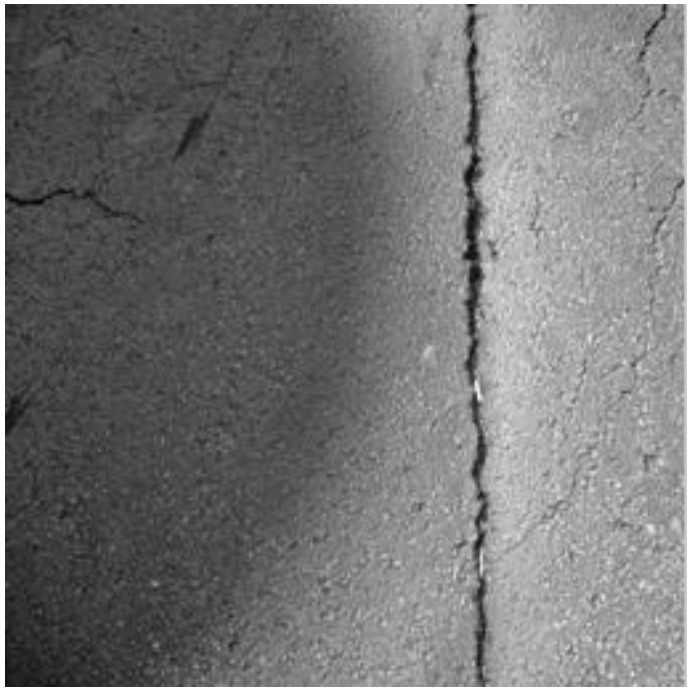
(b) Improved image

Fig 3-5 Comparison of pavement image improvement after non-uniform enhancement

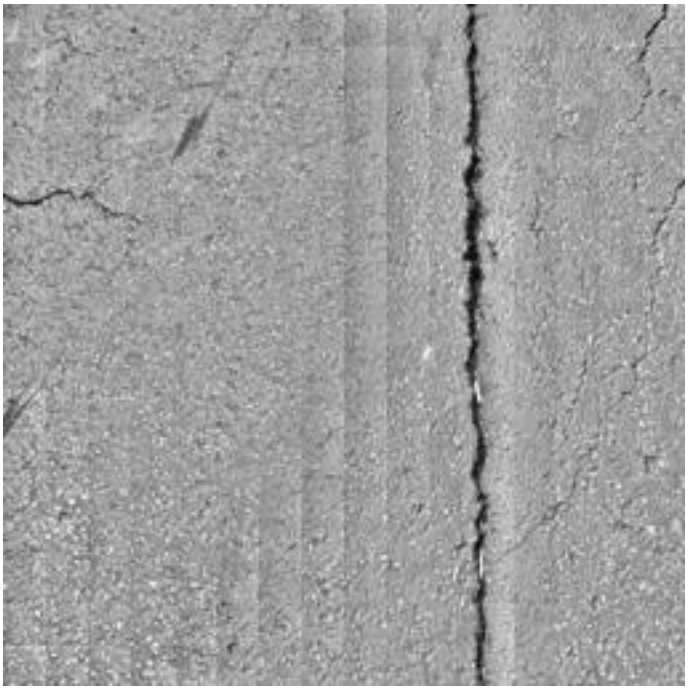
Figure 3-6 (a) shows an example of an image with a prominent shadow. The results after non-uniform improvement with a window size of 16 x 16 and 4 x 4 are

shown in figure 3-6 (b) and (c), respectively. It is shown that the smaller window size provides smoother results. However, this algorithm does not work for pavement defects with large areas, e.g., a pot hole whose area covers the entire window. Thus, smaller windows are more likely to cause errors.

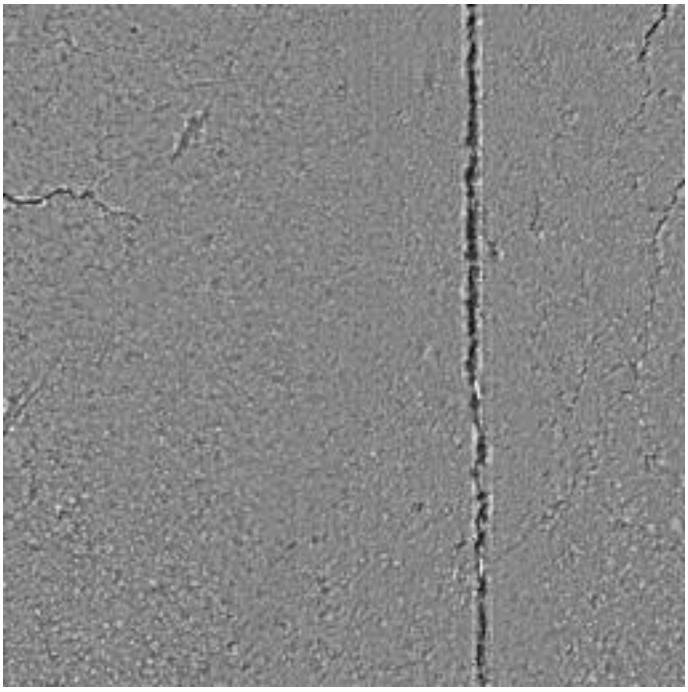
From the examples, the proposed new algorithm is proved to be able to correct the illumination of the background to make faithful thresholding of a wide variety of pavement distress source images feasible. However, this algorithm cannot work for pavement defects that cover a large area. For example, a pot hole whose area covers the whole window can fail the algorithm. After removing the non-uniform background information, the threshold method is applied to separate the background and the features. With uniform background images, it makes it possible to use an identical threshold to extract the crack images from original images.



(a) Original pavement image with shadow and cracks.



(b) Improved image after eliminating the non-uniform background with window size 16 by 16.



(c) Improved image after eliminating the non-uniform background with window size 4 by 4.

Fig 3-6 Compare of pavement image improvement from different window size

3.4 Thresholding

Thresholding is the simplest method of image segmentation. From the pavement gray scale image, thresholding can be used to create binary crack images. For pavement images, since crack pixels are always darker than the nearby pixels, if a pixel has an intensity value that is less than the threshold value, the corresponding pixel in the resultant image is considered as a crack seed, otherwise, it is considered as background or other non-crack information.

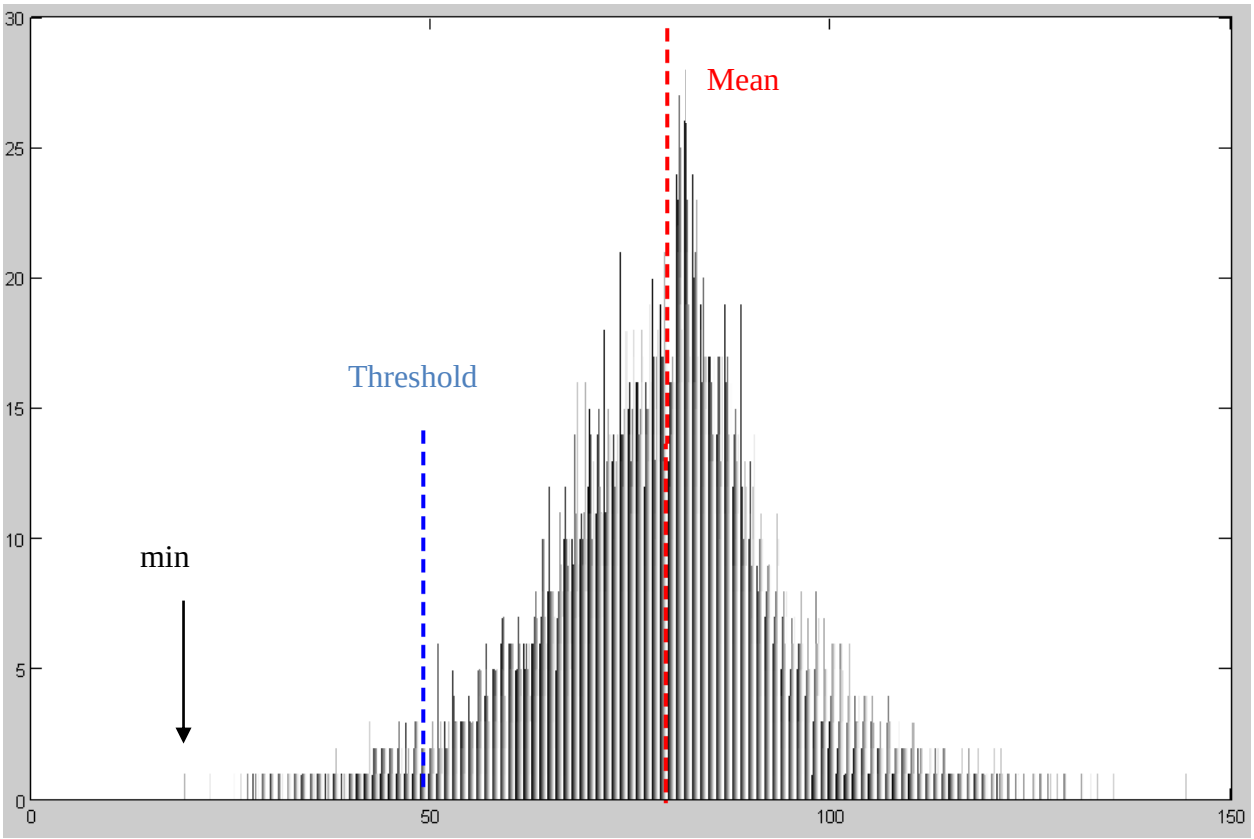


Fig. 3-7 Histogram of figure 3-5

Figure 3-7 is the histogram of figure 3-5. The selection of threshold varies for different pictures. For figure 3-5, the threshold is selected as,

$$T = mean - (mean - \min) \times 50\% \qquad (3\text{-}6)$$

The thresholded result is shown in figure 3-8. In figure 3-8, there are many noise points, and the cracks are discontinuous. With traditional pixel based algorithms, it is hard to eliminate these noises and the crack connection check is trivial, time-consuming, and error prone. A beamlet transform-based algorithm which can extract the linear feature of cracks is proposed in the following section.

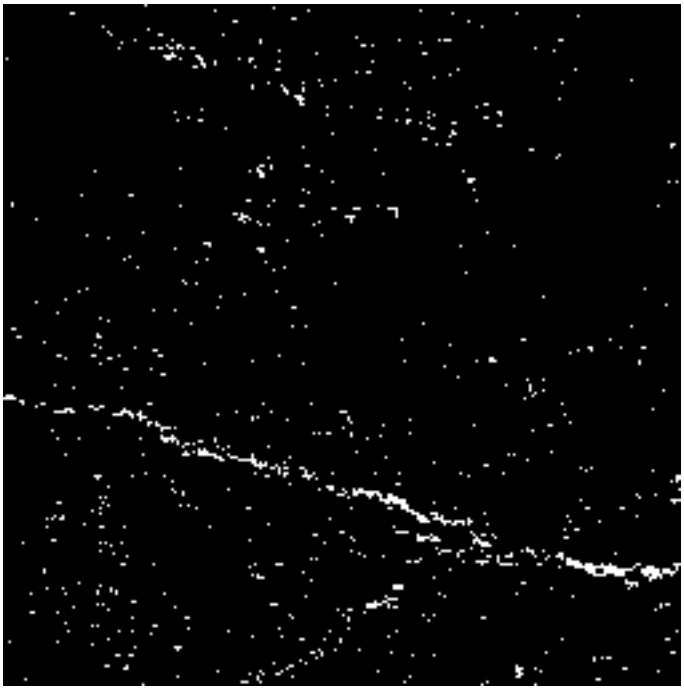


Fig. 3-8 Binary crack image of figure 3-5

Chapter 4

Crack Detection and Classification Based on Beamlet Transform

The concept of beamlet transform was first introduced by David L. Donoho and X. M. Huo as a tool for multi-scale image analysis [18]. Traditional signal detection algorithms for pavement crack detection are generally based on pixel-level processing, and most of them have very poor SNR ratios. Beamlet transforms are proven to be insensitive to noise, computationally efficient, and able to detect features with high accuracy. Beamlets are a simple dyadically organized collection of all line segments at different locations, orientations, and scales. The beamlet transform is the collection of line integrals along the set of all beamlets. This method allows for the extraction of linear features such as edges in noisy pictures, where traditional methods may fail.

4.1 Introduction of Beamlet Transform

Images are viewed as the continuum square $[0, 1]^2$ and the pixels as an array of 1/n-by-1/n squares arranged in a grid in $[0, 1]^2$. The following definitions are helpful for understanding the beamlet transform:

Definition 1 A dyadic square S is the collection of points $\{(x_1, x_2): [k_1/2^j, (k_1+1)/2^j] \text{ X } [k_2/2j, (k_2+1)/2^j]\}$, where $0 \le k_1, k_2 < 2^j$ for an integer $j \ge 0$.

Definition 2 Consider two vertices $v_1, v_2 \in [0, 1]^2$, within a dyadic square the line segment $b = \overline{v_1 v_2}$ is called a beam. There are $O(n^4)$ such beams if only beams connecting vertices $(k_1/n, k_2/n)$ are considered.

Definition 3 In order to reduce the cardinality, the concept of beamlet is introduced. Take the collection of all dyadic squares at scales $0 \le j \le J$ and fix resolution δ, the set of beamlets is the collection of all beams connecting vertices on the boundary of each dyadic square. There are $O(n^2 \log_2 n)$ beamlets [18]. Figure 4-1 shows beamlets at different scales. Fig. 4-1 gives some examples of beamlets at different scales.

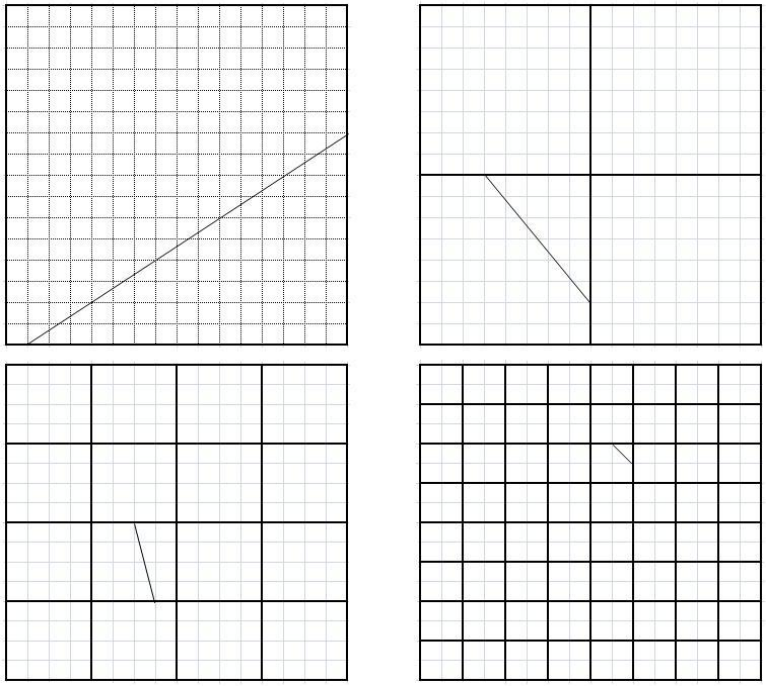


Fig 4-1 Beamlets at different scales

The beamlet transform is defined as the collection of line integrals along the set of all beamlets. Let $f(x_1, x_2)$ be a continuous function on 2-D space, where x_1 and x_2 are coordinates. The beamlet transform T_f of function f is defined as follows,

$$T_f(b) = \int_b f(x(l))dl\,,\ b \in B_E \qquad (4\text{-}1)$$

where B_E is the collection of all beamlets.

For a digital image, the beamlet transform is a measure of the line integral in the discrete domain. As figure 5 shows, the beamlet transform for all the points along the beamlet b is defined as,

$$f(x_1, x_2) = \sum_{i1,i2} f_{i_1,i_2}\phi_{i1,i2} \qquad (4\text{-}2)$$

where f_{i_1,i_2} is the gray level value of pixel (i_1, i_2), and ϕ_{i_1,i_2} is considered to be the weight function for each pixel. In this thesis, we use the following equation:

$$\phi_{i_1,i_2} = \frac{l_n}{\sqrt{L}} \qquad (4\text{-}3)$$

where L is the total length of the beam , and l_n is the length of a segment in each square pixel on the beam. Obviously,

$$L = \sum_n l_n \qquad (4\text{-}4)$$

Figure 4-2 shows the beamlet transform as a weighted sum of pixel values along the shaded line that the beamlet traverses. The gray level of each pixel is taken as the function value f of the corresponding square.

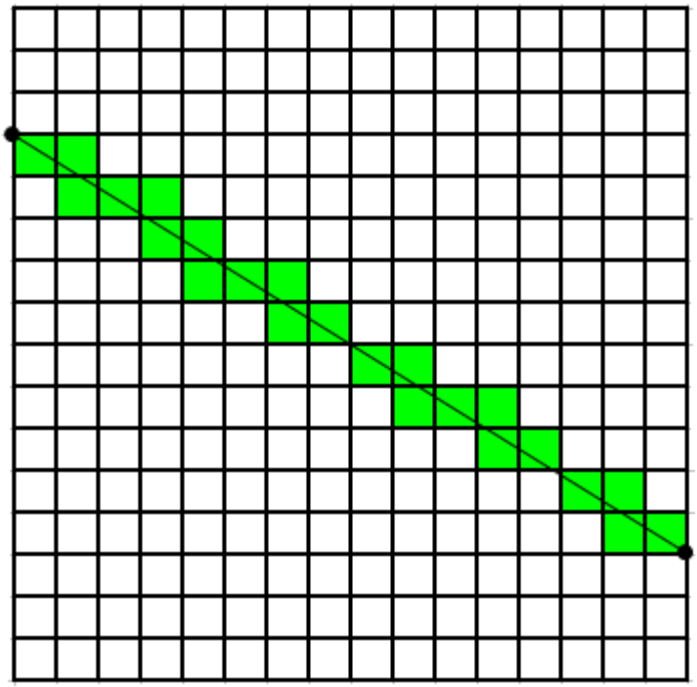


Fig. 4-2 Beamlet transform is a weighted sum of pixel values along the shaded line

4.2 Beamlet Transform Implementation for Crack Detection

As explained in the previous section, there are $O(n^2 \log_2 n)$ beamlets if multi-scaling is involved. In order to reduce the calculation. In this thesis, only single scaling is considered Thus there are $O(n^2)$ beamlets. The steps for performing beamlet transform are explained below.

(1) Partition image into smaller windows

As explained, the image needs to be partitioned into smaller rectangular windows. The large-sized window is robust to noise; however, it cannot provide detailed information.

Regardless of the window size, the total integration needed to be performed is $O(n^2)$; however, larger window size means longer beamlets, which increases the computational time. In this thesis, input images are with 256 by 256, and partitioned into windows sized with 16 by 16 for beamlet transform.

(2) Build Beamlets Dictionary

Beamlet Dictionary is a dyadically-organized library of line segments at a range of locations, orientations, and scales, which gives a multi-scale approximation to the collection of all line segments.

For single scale beamlet transform, all the windows have the same dimension, thus the same beamlet structures, so that the dictionary need only be calculated once, and be used for all the windows. For each beamlet, the following information is recorded,

a) The coordinators of the pixels that be considered to be on the beam.

b) The corresponding length l_n of each segment of the beam.

c) The total length of the beamlet L

d) From b) and c), the weights of the corresponding pixels are known.

The beamlets dictionary is saved. When a pavement image is processed, the dictionary is repeatedly used for each small window. In this way, it is easy to implement the algorithm in parallel and speed up the transform.

(3) Perform beamlet transform

After the beamlet dictionary is build and saved, it can be reused for each small window. For each window, the beamlet transform is applied and the beamlet which

provides the maximum value is selected if its value exceeds a threshold value (In this thesis, threshold T=1.0). The length of the beam determines the length of the cracks in the window. Keep the value for the block, and mark the corresponding beam. This is used for further analysis.

Fig. 4-3 Beamlet transform result of pavement crack image 3-8.

Perform beamlet transform as explained for the pavement crack image 3-8, the result is shown in figure 4-3.

4.3 Crack Connectivity Check

The connectivity check algorithm evaluates the extensibility of each crack pixel in the crack domain along all the eight directions, and also creates a link list for it. If the length is bigger than the threshold value, it is considered as a crack; otherwise, it is considered as a noise.

Here, a modified connectivity check algorithm is proposed. In previous sections, cracks have been detected using beamlet transform in each small block. The connectivity check can be performed even easier, and more efficiently.

To perform the crack connectivity check, four tables are required. See table 4-1. For each window, the max Beamlet transform value is defined as the crack length. The length of the crack is calculated by adding the crack length in each block along with the crack extension.

Table 4-1 Table needed to perform crack connectivity check

Table	**Description**	**Initialize**	**Value**
Status matrix	Record the check status of each block	"unchecked"	"unchecked", "no crack", or id
Length Table	Record the length of each branch.	0	Length of cracks
Branch Candidate Table	Record the start point of branch candidate	Empty	Each item includes start point coordinators, and mother branch number.

After the tables are built, the cracks are checked for connectivity and the following steps are followed. In these steps, every time a block has been checked, the corresponding item in the status matrix is changed to "no crack", or crack id. For example, for the 3rd branch of crack 2, the id should be "2-3".

1) Scan status matrix, block by block, and find the first block with a crack feature with “unchecked” status. Modify the status to the corresponding id number or “check” status, add the crack length in the block to the length table of the current branch, and then proceed to check all its eight neighboring blocks with “unchecked” status.

2). If one and only one neighbor is a crack block, add the crack length to the corresponding item in the length table, move to the neighboring block, and continue the process.

3). If there is more than one block detected with cracks, select one as the current branch extension direction and continue the extension check. Save all the others into the branch candidate table.

4). If there is no unchecked crack block remaining in the neighboring blocks, it means the branch extension has reached its end. If the branch length is shorter than a threshold, then it is not a real branch and will be ignored.

5). Find the next branch candidate from the branch candidate table, and continue the extension check until the table is empty.

6). The length of a crack is the sum of the length of all the branches contributing to that crack. Finally, if the length of the crack is shorter than a threshold, it is not considered to be a real crack.

The threshold used above for crack / branch length changes with the window size. From the experiment, the threshold T is calculated as,

$$T = 1.8S \tag{4-5}$$

where S is the size of the window.

Fig. 4-4 Connectivity check result for figure 4-3.

4.4 Crack Classification Standard

Generally, cracks in the pavement images possess linear features, embedded in noise, and are discontinuous. Additionally, the pavement images have specific patterns which make crack detection more difficult using traditional pixel based methods. The Beamlet transform will be a suitable algorithm for crack detection due to its robustness to line segment detection.

Following the above crack extension procedure, cracks are extracted and their projection in horizontal and vertical directions can be measured. This will in turn provide the information necessary for crack classification. Cracks are classified into four types: vertical, horizontal, transverse, and block types. The type of a crack is determined by its angle with the horizontal axis (Ω) and the number of branches in the crack, as

summarized in Table 1. Note that the angle Ω is calculated according to the start and end points of each crack. If branches exist, the crack is considered as block type irrespective of the angle of the cracks. For each window, the maximum beamlet transform value is defined as the crack length in the block. The total crack length is defined as the sum of all the blocks along the crack.

Table 4-2 Features for different types of cracks

Direction	Ω	Branches?
Vertical	$\Omega >= 60^{o}$	No
Horizontal	$\Omega <= 30^{o}$	No
Transverse	$60^{o} > \Omega > 30^{o}$	No
Block	-	Yes

Chapter 5

Test Results and Analysis

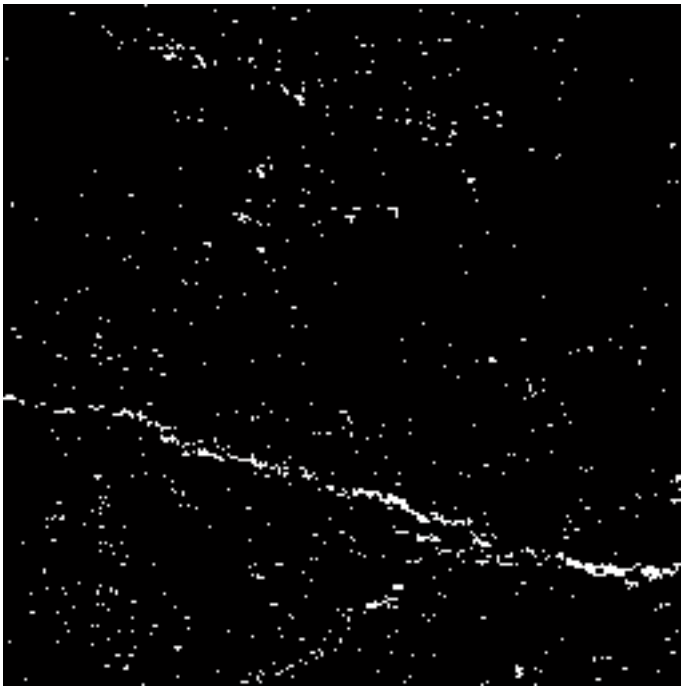
In this chapter, several pavement images with different types of cracks are processed with the proposed algorithm and the results are shown. According to the process that has been explained in the previous chapters, they are processed by non-uniform background improvement, thresholding, beamlet transform, and connectivity check.

The proposed algorithm has been implemented and its performance and simulation results are presented. Following shows different types of pavement cracks and the corresponding results from the Beamlet transform. Note that, example 1 is an image containing two horizontal cracks. Example 2 is a block crack which has four branches. Example 3 is composed of one horizontal and one vertical crack. Example 4 shows an image with a vertical crack. In example 5, it shows a block crack with two branches.

In each experiment, the original image, binary crack image, and the crack image after beamlet transform and extension check are presented. For each test, the classification result of the cracks is recorded in a table.

5.1 Experiment 1. Two Horizontal Cracks

(a) Original image



(b) Binary crack image after background enhancement and threshold

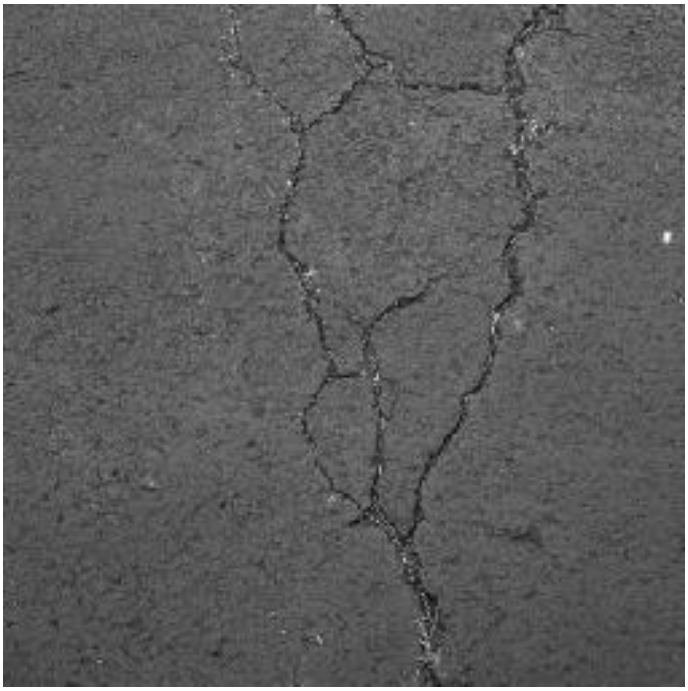
(c) Crack image after beamlet transform and extension check

Fig 5-1 Pavement image for two horizontal cracks

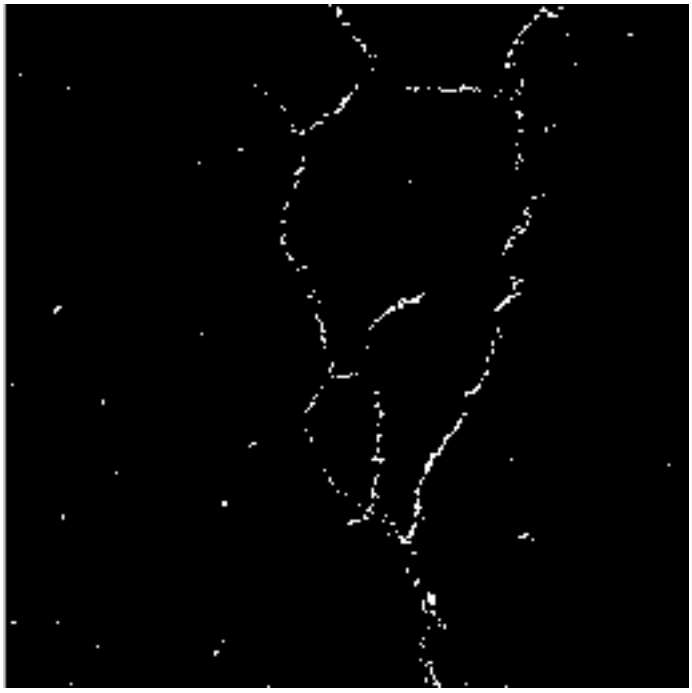
Table 5-1 Classification of two horizontal cracks shown in figure 5-1

Crack No.	Ω	Branches?	Length	Type
1	21^{o}	No	39.47	Horizontal
2	6^{o}	No	360.31	Horizontal

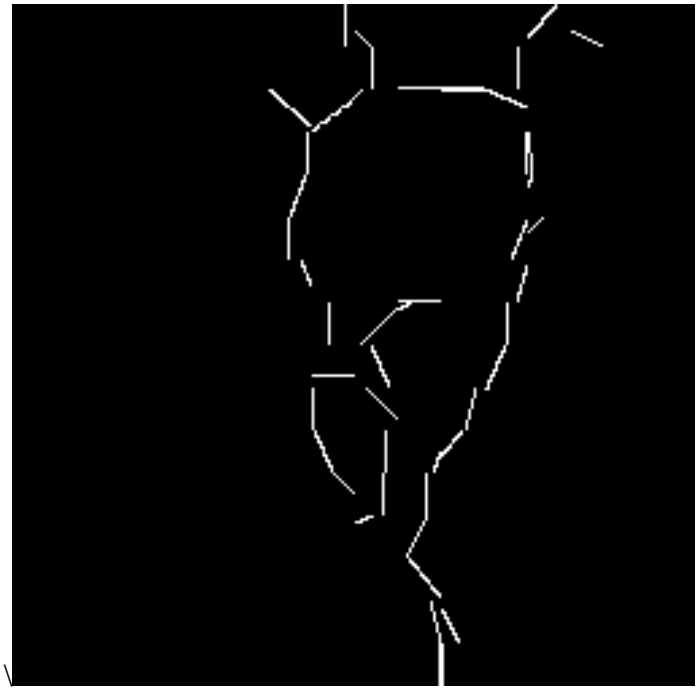
5.2 Experiment 2. A Block Type Crack



(a) Original image



(b) Binary crack image after background enhancement and threshold



(c) Crack image after beamlet transform and extension check

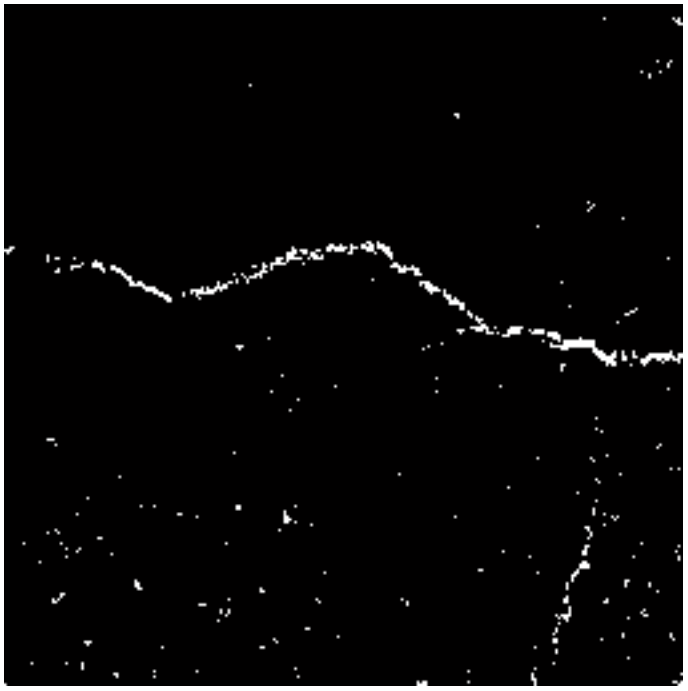
Fig 5-2 Pavement image for a block type crack

Table 5-2 Classification of block type cracks shown in figure 5-2

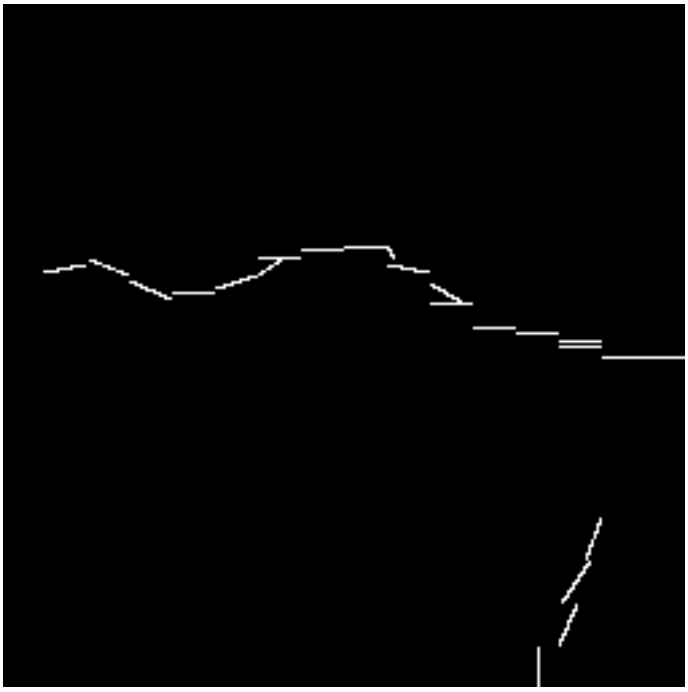
Crack No.	Ω	Branches?	Length	Type
1	-	4	993	Block

5.3 Experiment 3. A Horizontal and A Vertical Crack

(a) Original image



(b) Binary crack image after background enhancement and threshold



(c) Crack image after beamlet transform and extension check

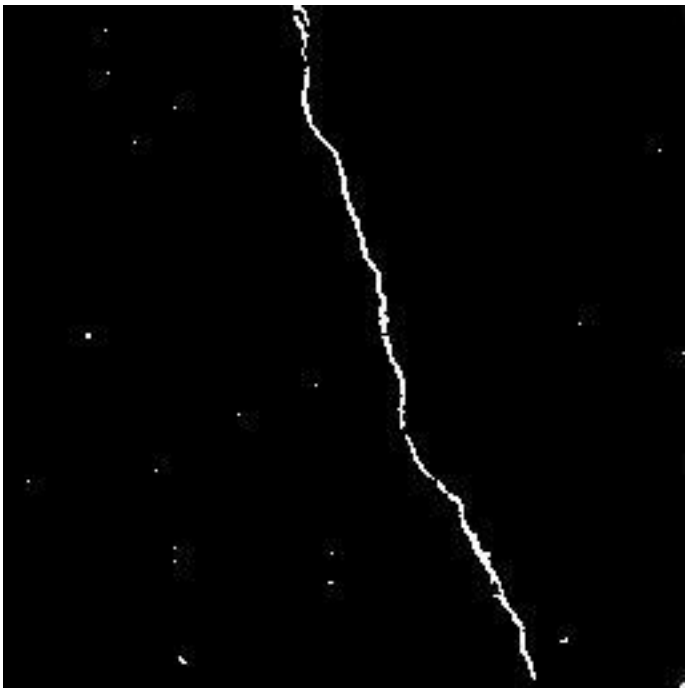
Fig 5-3 Pavement image for a horizontal crack and a vertical crack

Table 5-3 Classification of a horizontal and a vertical crack shown in figure 5-3

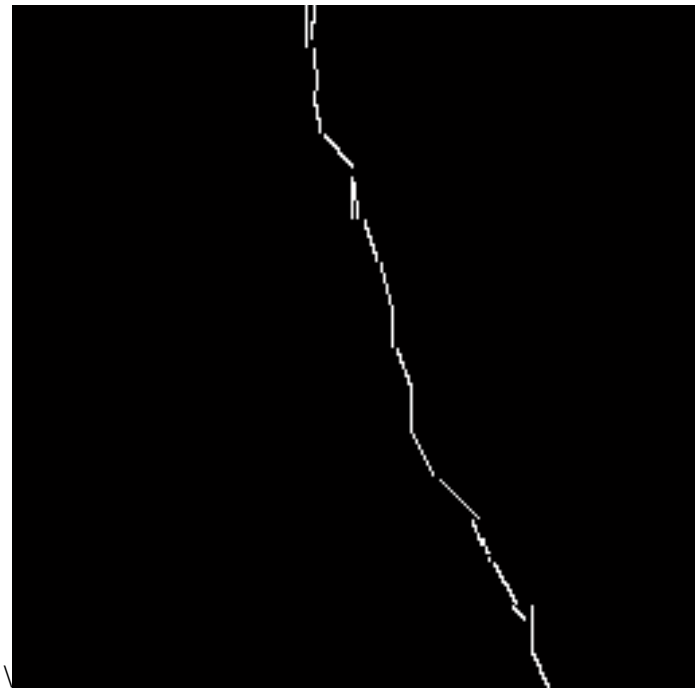
Crack No.	Ω	Branches?	Length	Type
1	9^o	No	308.07	Horizontal
2	67^o	No	108.10	Vertical

5.4 Experiment 4. A Vertical Crack

(a) Original image



(b) Binary crack image after background enhancement and threshold



(c). Crack image after beamlet transform and extension check

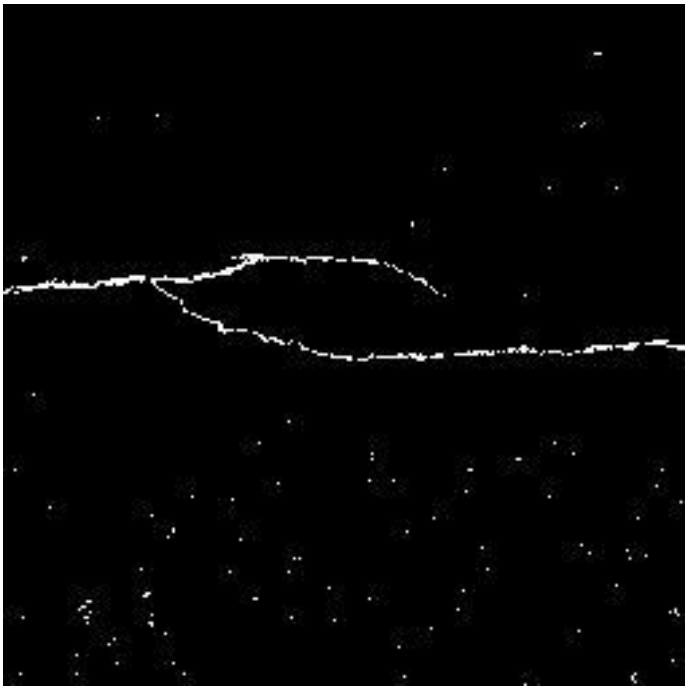
Fig 5-4 Pavement image for a vertical crack

Table 5-4 Classification of a vertical crack shown in figure 5-4

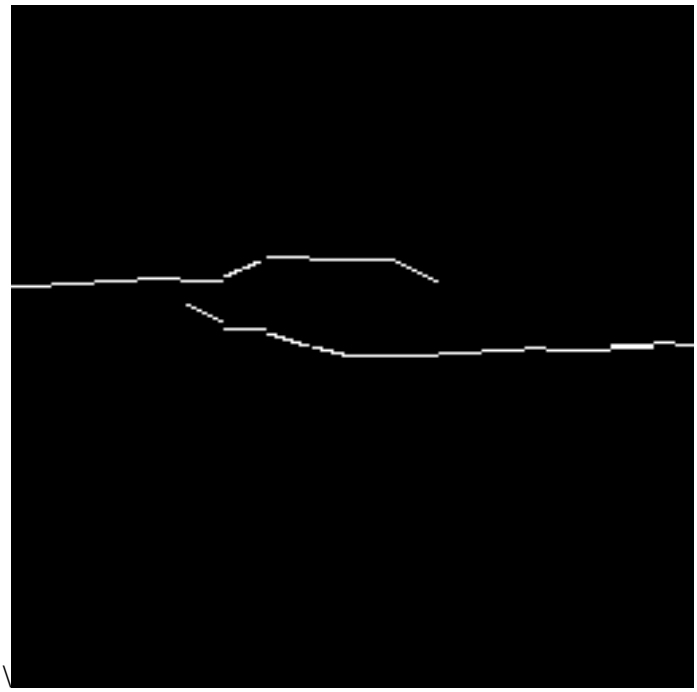
Crack No.	Ω	Branches?	Length	Type
71^o	No	341.15	Vertical	71^o

5.5 Experiment 5. A Block Type Crack

(a) Original image



(b) Binary crack image after background enhancement and threshold



(c) Result image

Fig 5-5 Crack image after beamlet transform and extension check

Table 5-5 Classification of a block type crack shown in figure 5-5

Crack No.	Ω	Branches?	Length	Type
1	-	2	425.73	Block

Chapter 6

Conclusion and Further Work

This thesis presents a beamlet transform based technique to extract the linear crack features from pavement images. Beamlet transform provides an effective method for the extraction of curvilinear features such as cracks in pavement images. Initially, an enhancement method is applied to reduce the effects of non-uniform background and undesired objects to facilitate the application of beamlet transform. Then, an identical threshold is applied and a binary crack image is obtained. By dividing the image into small windows and applying beamlet transform in each of them, the linear feature of a crack is extracted. Finally, the crack connection check is performed and is classified into horizontal, vertical, diagonal, or block types.

Experimental results provided in chapter 5 have demonstrated that the proposed beamlet transform based method is very effective with the presence of noise in pavement images. It can be applied on noisy pavement images and classify different types of cracks with a high rate of detection and very low rate of false detection. It could be a viable alternative to common pixel-based approaches for crack extraction.

The current algorithm can be improved by introduce the multi-scale pyramid analysis that can represent branches of cracks more efficiently. However, since the beamlet transform is used to extract linear features, it cannot be used to detect the defects with large area, such as pot holes.

Reference

[1] Tsao, S., Kehtarnavaz, N., Chan, P., and Lytton, R., "Image-based expert-system approach to distress detection on CRC pavement." *J. Transp. Eng.*, 120(1), 1994, pp. 52–64.

[2]. A. Georgopoulos, A. Loizos and A. Flouda, "Digital image processing as a tool for pavement distress evaluation", *Journal of Photogrammetry and Remote Sensing*, 50(1), 1995, pp 23-33.

[3]. H.D. Cheng, M. Miyojim, "Automatic Pavement Distress Detection System", *Journal of Information Sciences*, 108 1998, pp. 219-240.

[4]. Maser, K. R., "Computational techniques for automating visual inspection." Working Paper, 1987. Department of Civil Engineering, MIT, Cambridge, Mass.

[5]. Li L, Chan P, Rao A, Lytton R L., "Flexible pavement distress evaluation using image analysis", Proceedings of the Second International Conference on Applications of Advanced Technologies in Transportation Engineer, 18-21, August 1991, pp. 473-477.

[6]. J. Kittler, and J. Illingwort, 'Minimum error thresholding', Pattern Recognition, vol. 19, 1986, issue 1, pp. 41-47.

[7]. H. N. Koutsopoulos and A. B. Downey, 'Primitive-based classification of pavement cracking images', Journal of Transportation Engineering, vol. 19, 1993, issue 3, pp. 402 418.

[8]. JaChing Chou, O'Neill, W. A., and Cheng, H.D., "Pavement distress classification using neural networks", Systems, Man, and Cybernetics, 1994, IEEE international

Conference, vol. 1, pp. 397-401.

[9]. A. Georgopoulos, A. Loizos, and A. Flouda, “Digital image processing as a tool for pavement distress evaluation”, *ISPRS Journal of Photogrammetry and Remote Sensing*, 50(1), 1995, pp. 23-33.

[10]. Hosin “David” Lee, Jungyong “Joe” Kim, “Development of a Menual Crack Quantification and Automated Crack Measurement System”, Project TR-457, Public Policy Center, Civil and Enviromental Engineering, University of Iowa, 2005.

[11]. Hosin “David” Lee, Jungyong “Joe” Kim, “Development of a Crack Type Index”, *Tranportation Research Record: Journal of the Transportation Research Board*, No. 1940, 2005, pp. 99–109.

[12]. H. D. Cheng, J. R. Chen, C. Glazier, and Y. G. Hu, ‘‘Novel approach to pavement distress detection based on fuzzy set theory.’’, *J. Comput. Civ. Eng.*, 13(4), 1999, pp. 270–280.

[13]. Jian Zhou, Peisen S. Huang, Fu-Pen Chiang, ‘Wavelet-based pavement distress detection and evaluation’, *Optical Engineering*, 45(2), 2006, pp. 1-10.

[14]. Kelvin C. P. Wang, Qiang Li, and Weiguo Gong, “Wavelet-Based Pavement Distress Image Edge Detection with À Trous Algorithm”, *Transportation Research Record, Journal of the Transportation Research Board*, No. 2024, 2007, pp. 73-81.

[15]. Jian Zhou, Peisen Huang, Fu-Pen Chiang, “Wavelet-Based Pavement Distress Classification”, *Transportation Research Record: Journal of the Transportation Research Board*, vol. 1940, 2005, pp 89-98.

[16]. Bahram Javidi, Jack Stephens, Sherif Kishk, Thomas Naughton, John McDonald, Atef Issac, “Pilot for Automated Detection and Classification of Road Surface

Degradation Features", *JHR 03-293*, Nov. 2003.

[17]. S.Kother Mohideen, S. Arumuga Perumal, and M.Mohamed Sathik, "Image De-noising Using Discrete Wavelet Transform", *International Journal of Computer Science and Network Security*, vol. 8, No. 1, 2008, pp 213-216.

[18]. David L. Donoho, and Xiaoming, Huo, "Image Beamlet and Multiscale Image Analysis", *International Journal of Computer Science and Network Security*, vol. 8, No. 1, 2008, pp 213-216.

[19]. M. Gunaratne, Alex Mraz, Ivan Sokolic, "Study of the Feasibility of video logging with pavement condition evaluation", the Florida Department of Transportation, Report No. BC-965.

Appendix – Source Code for Matlab

```
%% Image Enhancement -- factor method
% input - I, matrix of original image
% output - Ip2, matrix of improved image
function Ip2 = imenhance(I);
[sx sy] = size(I);
%% Enhancement -- factor method
mi = min(min(I));
me = mean2(I);
ma = max(max(I));

sz = 4;         % window size is set to 16
kx = sx / sz;
ky = sy / sz;
In = zeros(sz,sz);
for i = 1:kx
    for j = 1:ky
        % sent value to small matrix In
        for m = 1:sz
            for n = 1:sz
                In(m,n) = I((i-1)*sz+m, (j-1)*sz+n);
            end
        end
        % Adjust the backgroud to me
        Io = afactor(In, me, ma, 0.3);
        for m = 1:sz
            for n = 1:sz
                Ip2((i-1)*sz+m, (j-1)*sz+n) = Io(m,n);
            end
        end
    end
end
figure,imshow(Ip2,[]);
title('uniform background image');


%%
```

```
% In - input matrix; b - target backgroud value; r - +-
range with mean
% Io - output matrix
function Io = afactor(In, b, maxv, r);

[sx sy] = size(In);

me = mean2(In);     %input mean
mi = min(min(In));
ma = max(max(In));

lb = me - (me - mi) * r
ub = me + (ma - me) * r

cnt = 0;
s = 0;

for i = 1:sx
    for j = 1:sy
        if(In(i,j)>=lb && In(i,j)<=ub)
            cnt = cnt + 1;
            s(cnt) = In(i,j);
        end
    end
end

m = mean(s);

fa = b / m;

hb = b + (maxv - b) * 0.5;

for i = 1:sx
    for j = 1:sy
        if(fa>=1 && In(i,j)>b)
            if(In(i,j)>hb)
                Io(i,j) = me;
            else
                Io(i,j) = In(i,j);
            end
        else
            Io(i,j) = In(i,j) * fa;
        end
    end
end
```

```
bsx = zeros(nx+ny,cnt);
bsy = zeros(nx+ny,cnt);

for i = 1:cnt  % For each beams
    x1 = Bm(1,i);
    y1 = Bm(2,i);
    x2 = Bm(3,i);
    y2 = Bm(4,i);
    L(i) = sqrt((Bm(1,i)-Bm(3,i))^2 + (Bm(2,i)-Bm(4,i))^2);

    % Solve the linear equation y=a*x+b
    if(x1==x2)  % horizontal
        ym = min(y1, y2);
        for j=1:ny-1
            bsx(j,i) = x1;
            bsy(j,i) = ym + j -1;
            len = 1;
        end
    elseif(y1==y2)  % vertical
        xm = min(x1, x2);
        for j=1:nx-1
            bsx(j,i) = xm + j -1;
            bsy(j,i) = y1;
            len = 1;
        end
    else % Others
        a = (y1-y2)/(x1-x2);
        b = y1 - a * x1;
        xm = min(x1, x2);
        xma = max(x1, x2);
        nnx = xma - xm + 1;
        ym = min(y1, y2);
        yma = max(y1, y2);
        nny = yma - ym + 1;
        for j = 1:nnx
            xc = xm + j - 1;
            yc = a * xc + b;
            bxt(j) = xc;
            byt(j) = yc;
        end
        for j = nnx+1:nnx+nny
            yc = ym + j - nnx - 1;
            xc = (yc - b) / a;
            bxt(j) = xc;
            byt(j) = yc;
        end
        % Sort them, and put into new array bxt2, byt2
```

```
        bxt2 = zeros(1,nx+ny);
        byt2 = zeros(1,nx+ny);
        for j = 1:nx+ny
            bxm = max(bxt);
            if(bxm == 0)    % empty
                break;
            end
            for k = 1:(nx+ny)  % Take this point out
%                if(bxt(k) == bxm)
                if(abs(bxt(k)-bxm)<1.0e-3)
                    bxt2(j) = bxt(k);
                    byt2(j) = byt(k);
                    bxt(k) = 0;  % Clear the get out point
                    byt(k) = 0;
                end
            end
        end
        ct = j;
        [sz1,sz2] = size(bxt2);
        %Get bsx, bsy, and len
        L(i) = 0;
        for j = 1:sz2-1
            bsx(j,i) = db2int(min(bxt2(j),bxt2(j+1)));
            bsy(j,i) = db2int(min(byt2(j),byt2(j+1)));
            len = sqrt((bxt2(j)-bxt2(j+1))^2 + (byt2(j)-
byt2(j+1))^2);
            L(i) = L(i) + len;
        end
    end
end

%% round off function
% input - bd
% output - b
function b = db2int(bd)

for b=0:10000
    if((b+1)>bd && (b<=bd))
        break;
    end
end


%% bt2 -- beamlet transform of beam start from (bx1, by1)
to (bx2, by2)
% I -- input matrix (square)
function [T, M, tmp] = bt2(I, bsx, bsy, len, L, th);
```

```
[nx, ny] = size(I);
M = zeros(nx,ny);

[bx, by] = size(bsx);
T = zeros(1, by);
Tm = 0;
tmp = 0;

for i = 1:by  % check each beam
    for j = 1:bx
        x = bsx(j,i);
        y = bsy(j,i);
        if(x==0 || y==0)
            break;
        end
        if(I(x,y)>0.2)
            T(i) = T(i) + len(j,i)/sqrt(L(i));
        end
%        T(i) = T(i) + I(x,y)*len(j,i)/sqrt(L(i));
    end
    if(T(i)>Tm)
        Tm = T(i);
        im = i;
        tmp = L(i);
    end
end
if(Tm > th)
    i = im;
    for j=1:bx
        x = bsx(j,i);
        y = bsy(j,i);
        if(x==0 || y == 0)
            break;
        end
        M(x,y) = 1;
    end
else
    tmp = 0;
end
```

```
%% search neighborhood for crack blocks
%input: ct - crack table, record the length; ic, jc -
coordinators;
%input: bc-branch counter; lac-table, record lable for crack
id;
%input:, lab-table, record lable for branch id;
%input: tmpm-record the start point of each branch;
%input: lfr - crack length from root;
%input: minLen - crack length threshold;
%input: bcb - branch counter backup
% output: ic, jc - coordinators, bc - branch counter;
% output: tmpm - record the start point of each branch;
% output: lCrack - the length of the found crack
% output: lac - table, record lable for crack id;
% output: Sta-status matrix, lBroot-flag, find a root.

function [ic, jc, bc, tmpm, lCrack, lac, Sta, lBroot] =
sneighb(ct, ic, jc, Sta, m, n, Lenth, bc, tmpm, lac, lab,
lfr, minLen, bcb);

[sx sy] = size(Sta);
iclist = [ic-1, ic, ic+1];
jclist = [jc-1, jc, jc+1];
lCrack = 0;
cnt = 1;
ict = ic; jct = jc;
tmpm = tmpm;
lBroot = 0;
bcb = bcb;
for im = ic-1:ic+1
    for jm = jc-1:jc+1
        if(im >0 && im < sx+1 && jm >0 && jm < sy+1 &&
Sta(im,jm)==0 && lac(im,jm) == 0)
            Sta(im,jm)=1;
            if(Lenth(im,jm)>0)      % add it to the ct list
                lCrack = 1;
                if(cnt == 1) % the first one, take as the
current branch
                    ict = im;  jct = jm;   % temp saved %
we will move to this point
                    ct(m,n) = ct(m,n) + Lenth(im,jm);
```

```
                    lac(im,jm) = m;
                else % add a branch candidate
                    if(lfr<minLen) bcb = bc; end
                    tmpm(bc,1) = im; tmpm(bc,2) = jm;
tmpm(bc, 3) = n;
                    lroot = 1; % reset
                    lBroot = 1;
                    bc = bc + 1;
                    lac(im,jm) = m;
%                   lab(im,jm) = bc;
                end
                cnt = cnt + 1;  % branch
            end
        end
    end
end

ic = ict;  jc = jct;
```

```
%% threshold to get binary crack image
% input: x - input image; pb - factor for threshold
% output - ttt,
function J = thr2(I, pb)

[sx sy] = size(I);
J = zeros(sx,sy);

mi = min(min(I));
ma = max(max(I));
me = mean2(I);

tb = me - (me - mi)*pb
%tc = double(me - (me - mi)*pc);

for x=1:sx
    for y=1:sy
        if(I(x,y)>=tb)  %cracks
            J(x,y) = 0.0;
        else
            J(x,y) = 1.0;
        end
    end
end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Main Function
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% read input images and transfer to gray image
a = imread('8.jpg');
I = rgb2gray(a);
[sx sy] = size(I);
b = imresize(a, 256/sx);
I = rgb2gray(b);
figure,imshow(I,[]);
title('Original Image');

sx = 256; sy = 256;

Ip = imenhance(I);

% update mi, me, ma
mi = min(min(Ip));
me = mean2(Ip);
ma = max(max(Ip));

thb = 1.0;  % threshold for beamlet transform
thc = 0.4;  % threshold for crack image

% hard thresholding
J0 = thr2(Ip, thc);
figure,imshow(J0,[]);
title('Crack Image');

%level 4
le = 4;
sxc = sx / (2^le);
syc = sy / (2^le);
for j=1:2^le
     for m=1:sxc
                for n=1:syc
                    Jt(m,n) = J0((i-1)*sxc+m, (j-1)*syc+n);
                end
            end
            %
            if(le==1)
                [T, M, tmp] = bt2(Jt, bsx1, bsy1, len1,
L1,th);
            elseif(le==2)
                [T, M, tmp] = bt2(Jt, bsx2, bsy2, len2,
L2,th);
            elseif(le==3)
```

```
                [T, M, tmp] = bt2(Jt, bsx3, bsy3, len3,
L3,th);
            elseif(le==4)
                [T, M, tmp] = bt2(Jt, bsx4, bsy4, len4,
L4,thb);
            elseif(le==5)
                [T, M, tmp] = bt2(Jt, bsx5, bsy5, len5,
L5,th);
            elseif(le==6)
                [T, M, tmp] = bt2(Jt, bsx6, bsy6, len6,
L6,th);
            elseif(le==7)
                [T, M, tmp] = bt2(Jt, bsx7, bsy7, len7,
L7,th);
            end
            lev(i,j,le) = max(max(T));    % save the max
value in lev
            max(max(T))
            for m=1:sxc
                for n=1:syc
                    if(le==1)
                        M1((i-1)*sxc+m, (j-1)*syc+n) =
M(m,n);
                    elseif(le==2)
                        M2((i-1)*sxc+m, (j-1)*syc+n) =
M(m,n);
                    elseif(le==3)
                        M3((i-1)*sxc+m, (j-1)*syc+n) =
M(m,n);
                    elseif(le==4)
                        M4((i-1)*sxc+m, (j-1)*syc+n) =
M(m,n);
%                         if(m==1 || m==sxc || n==1 ||
n==syc)
%                             M4((i-1)*sxc+m, (j-1)*syc+n) =
0.5;
%                         end
                    elseif(le==5)
                        M5((i-1)*sxc+m, (j-1)*syc+n) =
M(m,n);
                    elseif(le==6)
                        M6((i-1)*sxc+m, (j-1)*syc+n) =
M(m,n);
                    elseif(le==7)
                        M7((i-1)*sxc+m, (j-1)*syc+n) =
M(m,n);
                    end
```

```
                end
            end
            Lenth(i,j) = tmp;
        end
    end
        figure,imshow(M4,[]);
        title('binary image 4');
end

%% Connection Check
% scan block by block
Sta = zeros(2^le, 2^le);        % status matrix: 0-
unchecked/initialize value;
lac = zeros(2^le, 2^le);        % lable for crack id
lab = zeros(2^le, 2^le);        % lable for branch id

m = 0;      % count for crack number, start from 1
ct = zeros(16,100);
minLen = 30.0;
for i = 1:2^le
    for j = 1:2^le
        % scan for a block with crack
        if (Sta(i,j)==0 && Lenth(i,j)>0)   % new crack
block found
            m = m + 1;
            n = 1;  % count for branch number, start from
1, reset to 1 after finish check every crack
            bc = 1;    bcb = bc; % initialize branch
counter
            Sta(i,j) = 1;
            lac(i,j) = m;   lab(i,j) = n;
            tmpm(1, 1) = i;  tmpm(1,2) = j;   tmpm(1, 3) =
1; % tmpm record the start point of each branch; sp of
crack is sp of the 1st branch of the crack
            lfr = 0;  %Initialize
            % search the neighbor of it
            ic = i; jc = j;
            while bc>0     % search every branch
                ct(m,n) = ct(m,n) + Lenth(ic,jc);   % crack
table, record the length
                for cnt = 1:255
                    [ic, jc, bc, tmpm, lCrack, lac, Sta,
lBroot] = sneighb(ct, ic, jc, Sta, m, n, Lenth, bc, tmpm,
lac, lab, lfr, minLen, bcb);
                    lab(ic, jc) = n;
                    if(lCrack==0) break;  end % no further
crack (extension of this branch is over), break
```

```
                    if(lBroot == 1) lfr = 0; end    % Reset
                    ct(m,n) = ct(m,n) + Lenth(ic,jc);
                    lfr = lfr + Lenth(ic,jc);  % length
from root
                    if (lfr>minLen)     bcb = bc; end
                end
                if(m==1 && n ==5)
                    tt = ct(m,n)
                    [ic,jc]
                end
                if(lfr<minLen && ct(m,n)>minLen)  % less
than threshold, this is not a branch, continue another way
                    bc = bc - 1;
                    if(bc == 0) break; end
                    ic = tmpm(bc, 1); jc = tmpm(bc,2);  %
Get the last record branch, search from there
                    lfr = 0;   % reset
                    if (bc<bcb)  n = n + 1; bcb = bc; end
                    lab(ic,jc) = n;
                elseif(ct(m,n)<minLen)  % not a branch,
jump to previous branch
                    lab(ic,jc) = tmpm(bc, 3);
                    [ic,jc]
                    bc = bc - 1;
                    if(bc == 0) break; end
                    ic = tmpm(bc, 1); jc = tmpm(bc,2);  %
take new brach candidate out
                    ct(m,n) = 0;
                    lab(ic,jc) = n;
                    [ic,jc,n,bc]
                else    % this branch search over
                    bc = bc - 1;
                    if(bc == 0) break; end
                    bcb = bc;
                    ic = tmpm(bc, 1); jc = tmpm(bc,2);  %
take new brach candidate out
                    n = n+1;
                    lab(ic,jc) = n;
                end
                Sta(ic,jc) = 1;
            end  % End while loop, go to find next crack
            %calculate the total length of crack
        end
        Sta(i,j)=1;
    end
end
```