TRACKING AND DETECTION OF CRACK PATTERNS USING MINIMAL PATH TECHNIQUES

A Thesis
Presented to
The Academic Faculty

by

Vivek Kaul

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Electrical and Computer Engineering

Georgia Institute of Technology
December 2010

TRACKING AND DETECTION OF CRACK PATTERNS USING MINIMAL PATH TECHNIQUES

Approved by:

Professor Anthony Yezzi, Advisor
School of Electrical and Computer Engineering
Georgia Institute of Technology

Professor Patricio Vela
School of Electrical and Computer Engineering
Georgia Institute of Technology

Professor James Tsai, Co-Advisor
School of Civil and Environmental Engineering
Georgia Institute of Technology

Professor Ayanna Howard
School of Electrical and Computer Engineering
Georgia Institute of Technology

Professor Allen Tannenbaum
School of Electrical and Computer Engineering
Georgia Institute of Technology

Professor Sung Ha Kang
School of Mathematics
Georgia Institute of Technology

Date Approved: 27 August 2010

To all my teachers,

starting from my parents,

to whom I owe everything precious in my life.

ACKNOWLEDGEMENTS

First of all, I would like to thank my advisors Dr Anthony Yezzi and Dr James Tsai for guiding me through this doctoral thesis project. Dr Yezzi is an inspirational mentor and a even better human being. I have learned a lot from him and our mutual appreciation for St Francis made this relationship even more special. I thank him with all my heart. Dr Tsai supported me all through my doctoral thesis, when funding was hard to come back. He involved me in an intellectually challenging and satisfying project and I am grateful to him for doing that. I would also like to thank my previous advisors Dr Daniel Schrage in aerospace engineering and Dr Russell Mersereau for being such wonderful guides. Dr Schrage generously supported me during my masters in aerospace engineering and Dr Mersereau facilitated my transfer to electrical engineering. I want to also mention my favorite teachers in Georgia Tech who made learning such a pleasurable experience: Chris Heil, John Barry, Vladimir Kolchinsky, Alex Shapiro and Paman Illiev. I would like to thank my thesis reading committee members Dr Allen Tannenbaum and Dr Patricio Vela for taking time to go through my thesis and being there both for my proposal and thesis defense. Dr Ayanna Howard and Dr Sung Ha Kang graciously accepted to be on the thesis defense committee and I appreciate the time they have taken out of their busy schedules to be part of the committee.

I would like to acknowledge my labmates Guillermo Gallego, Vikram Appia, Balaji Ganapathy, Daniel Bishop, Ping-Chang Shih and Jun-Hee Heu for making the lab such aa pleasurable place to work in. Guillermo is a role model for all people in the lab because of his commitment and discipline towards his work. He went out of the way to help at each stage of my doctoral study and I would like to thank him for that. A

special mention for my friend Umair Altaf who in spite of his own work commitments went out of the way to help me with editing and proof-reading the thesis document. Without his help, I would not have graduated!

On the personal front, I would begin my thanking my parents Indira and Devinder Kaul for being constant sources of support for me throughout my life and allowing me to pursue my dreams. A special role has been played by two spiritual teachers in my life, Tripurari Swami and Radhanath Swami, and words are not enough to express my gratitude for them because they opened new horizons in my constricted and restricted life. In addition, I want to mention Vedasara who is both my teacher and my friend for being so kind to tolerate me and teach me valuable lessons in life. I also take inspiration from the exlemperory life of my Aunt Nancy Kaul whose sacrificing and non-complaining attitude makes all my difficulties and complaints look hollow and shallow.

Lastly, I would like to thank my innumerable friends for being there always for me. I have had a special relationship with Subodh Madhiwale and Nirmal Thakker and we shared so many fun times during our stay in Atlanta. I thank Kamlesh Nair, Nalini Polavarapu, Subramanyam Kalyanasundaram, Avishek Aiyar, Sumit Mishra, Vibhor Jain, Shawn Lankton, Kymry Jones and Katie Lebedev for making my stay in Atlanta memorable and unforgettable. Thanks also to all my friends from school and college who continue to tolerate me.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

The research in the thesis investigates the use of minimal path techniques to track and detect cracks, modeled as curves, in critical infrastructure like pavements and bridges. We developed a novel minimal path algorithm to detect curves with complex topology that may have both closed cycles and open sections using an arbitrary point on the curve as the sole input. Specifically, we apply the novel algorithm to three problems: semi-automatic crack detection, detection of continuous cracks for crack sealing applications and detection of crack growth in structures like bridges.

First, we provide the background of the problem of crack detection and critically assess the strengths and limitations of six current algorithms. Detection of cracks in these structures is very challenging because of multiple textures, shadows, variable lighting, irregular background and high noise present in the images, and this motivated our research into minimal path techniques. Next, a background of the minimal path techniques theory is provided. The current state of the art minimal path techniques only work with prior knowledge of either both terminal points or one terminal point plus total length of the curve. For curves with multiple branches, all terminal points need to be known. Therefore, we developed a new algorithm that detects curves and relaxes the necessary user input to one arbitrary point on the curve. The document presents the systematic development of this algorithm in three stages. First, an algorithm that can detect open curves with branches was formulated. Then this algorithm was modified to detect curves that also have closed cycles. Finally, a robust curve detection algorithm was devised that can increase the accuracy of curve detection. The robust algorithm tackles two problems: spurious detection of curve portions and inability to detect complex topological curves that have sharp corners

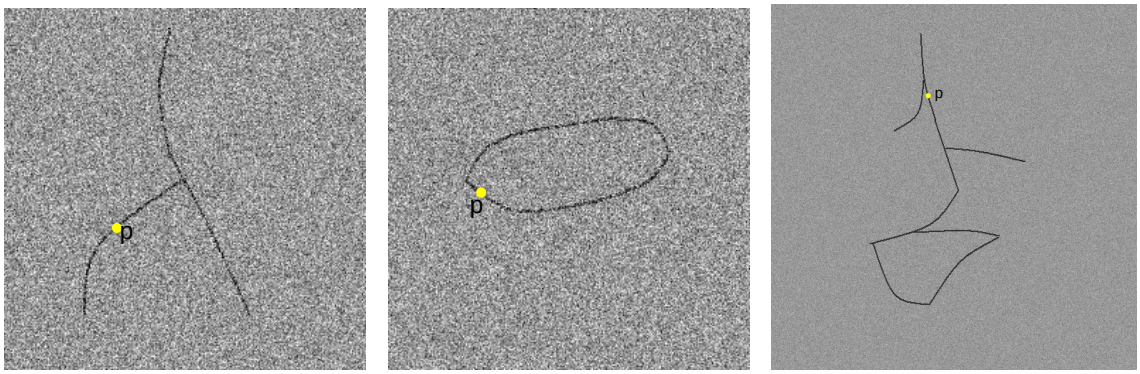
at branches. The algorithm was applied to crack images and the results of crack detection were validated against the ground truth. A new quantification measure called the buffered Hausdorff distance measure was developed for the experimental validation. In addition, the algorithm was also used to detect features like catheter tube and optical nerves in medical images. We finally conclude by giving some future research directions. In particular, the algorithm can be extended to detect higher dimensional curves and the computational speed of the algorithm can be improved by optimizing the use of redundant information.

CHAPTER I

INTRODUCTION

Tracking and detecting crack patterns in critical infrastructure is essential for taking corrective actions. Over the last few decades, increasing computational speed has enabled the use of computer vision and image analysis algorithms for tracking and detecting desired features of interest in different fields like remote sensing and medical imaging. There is a great potential for the use of computer vision algorithms in two civil engineering applications: monitoring of cracks in static critical structures like bridges, and the detection of cracks in pavements. Most often, after the crack in a structure or the distress in a pavement surface is projected onto 2D images, image analysis can be employed to extract physical or geometric quantities that are of engineering interest. Therefore, image analysis tools are being developed to detect and track structural damage in static images or dynamic image frames.

Cracks in images can be described as spatially elongated, narrow objects that have darker intensity compared to the background. These cracks can be modeled as curves that have low intensity values. The current research explores the use of minimal path techniques in tracking and detecting cracks in pavements and structures by modeling them as curves. The current state of the art minimal path theory works only with prior knowledge about both the terminal points or a terminal point plus total length of the curve. For more complex curves with branches, all terminal points are required to be known. The focus of the research will be to find curves under less restrictive prior knowledge assumptions. Next, we describe the research objectives in detail.



(a) (b) (c)

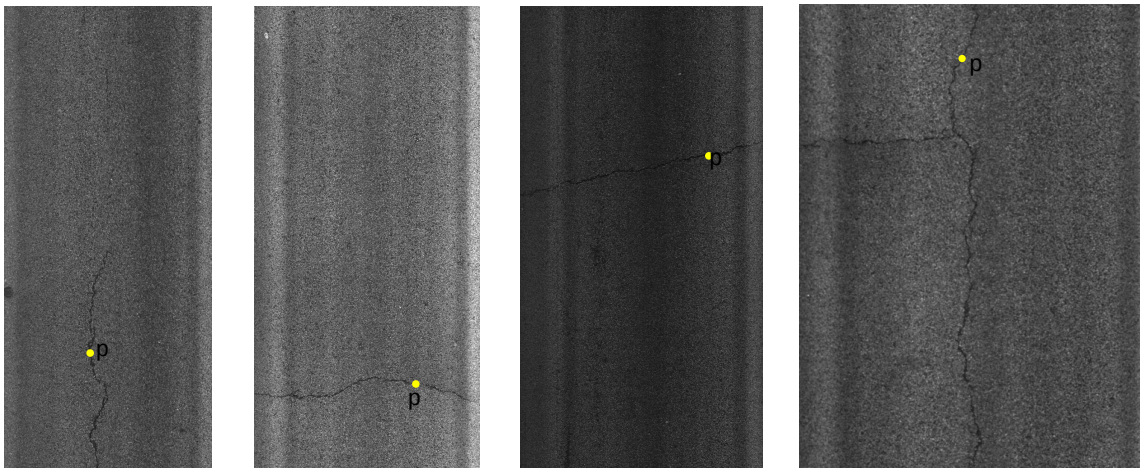
Figure 1.1: (a) Curve with branches and initial point p. (b) Closed curve and initial point p. (c) Complex topological curve with multiple branches and a cycle and initial point p.

1.1 Research Objectives

The primary objective of the research is to investigate minimal path techniques and develop an algorithm for detecting curves with complex topologies from a single arbitrary point. Figure 1.1 shows synthetic images with different types of curves and an arbitrary user defined initial point. We want the algorithm to detect the complete curve in each of these cases using just the user defined initial point. The algorithm will be a useful contribution in the area of minimal path methods because existing algorithms need prior knowledge of all endpoints to successfully detect complex curves. This algorithm will be applied to the following problems:

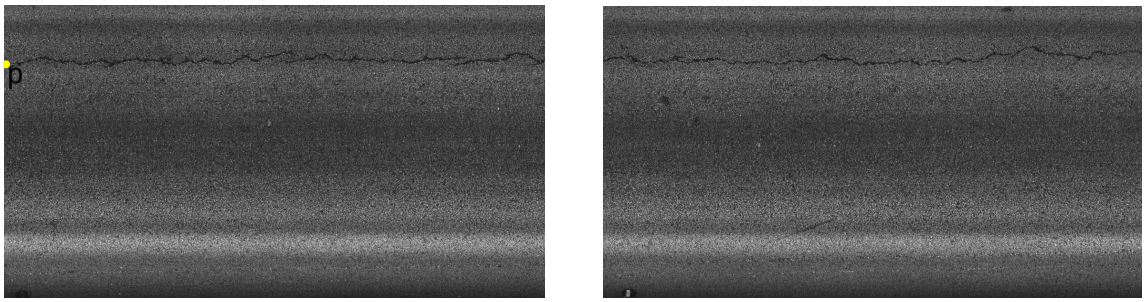
1. **Semi-automatic detection of cracks:** We propose to detect cracks in pavements when the user provides an arbitrary point on the cracks. Figure 1.2 displays different types of cracks that we want to detect using our algorithm.

2. **Detection of continuous cracks for crack sealing applications:** We propose to detect continuous cracks that extend over several miles by just providing the starting point on a crack as input to the algorithm. Figure 1.3 shows two consecutive images that have a continuous crack. Once, the algorithm detects



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

Figure 1.2: (a) Longitudinal crack with initial point p. (b) Transverse crack with initial point p. (c) Diagonal crack with initial point p. (d) Crack having branches with initial point p.



(a) (b)

Figure 1.3: (a) First image with continuous crack and initial point p. (b) Second image with the extended continuous crack.

the crack in the first image, the endpoint of the crack in the image can be used as the algorithm input for the next image. Hence, consecutive images can be used to detect very long, continuous cracks and the detected crack map can be utilized very efficiently to generate the optimal path for the automatic crack sealing machine.

3. **Tracking crack growth in critical structures:** Figure 1.4 illustrates the problem of tracking crack growth in critical structures. Given a user provided

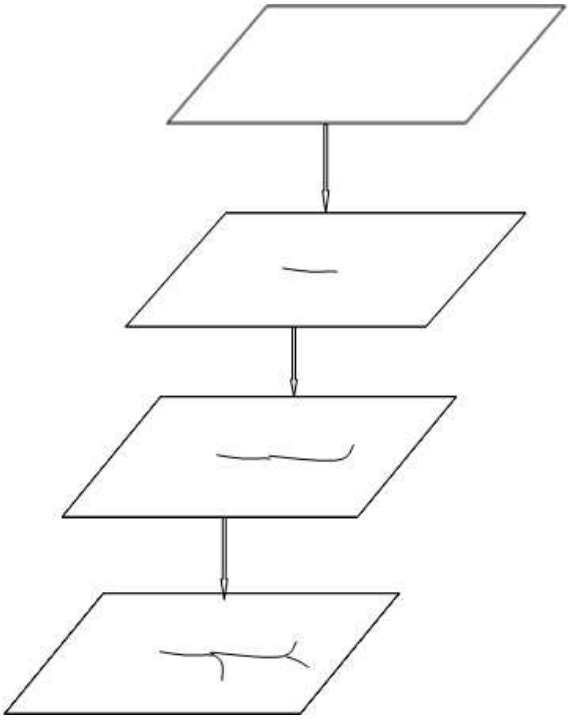


Figure 1.4: Figure illustrating crack growth in structures with time.

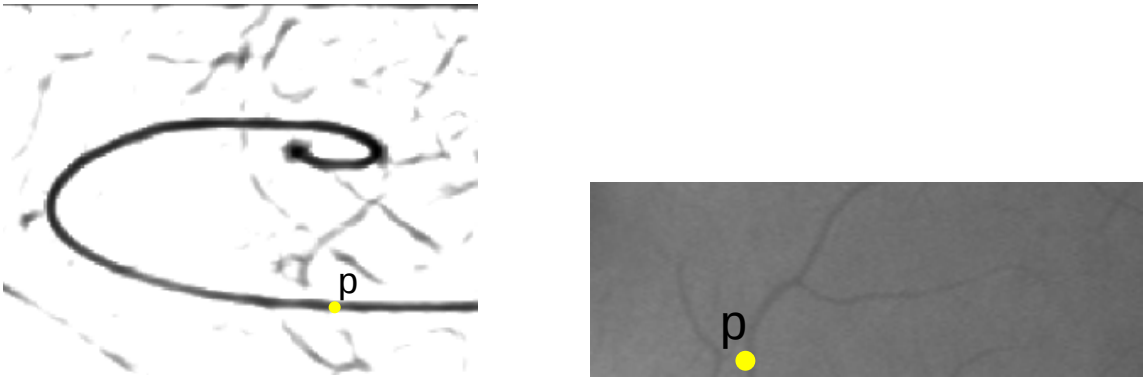
point on the crack at the crack initiation step, we want to find crack maps at various time instants to trace out the crack growth. The crack detected by our minimal path algorithm at one time frame can be used as input for the next time frame after registering the two temporal images.

4. **Medical applications:** We also want to use our algorithm to detect features in medical images that can be modeled as curves with a user provided initial point. Figure 1.5 shows two such medical images that contain a catheter tube and optical nerves in the eye retina.

1.2 Key Contributions

This thesis has three key contributions that are listed below:

1. The most fundamental contribution is the development of a self-terminating algorithm that can detect curves of complex topology with just one user provided point on the curve. Existing minimal path algorithms need the user to provide



(a) (b)

Figure 1.5: (a) Catheter tube image with initial point p. (b) Retinal Image with initial point p.

information about all endpoints to detect complex curves and our algorithm minimizes the user interaction significantly.

2. We apply the algorithm to the problem of crack detection in structures and pavements in unique ways. Specifically, we tackle the problems of semi-automatic crack detection, detection of long and continuous longitudinal cracks for crack sealing applications and tracking of crack growth in critical structures. A review of six existing crack detection algorithms and their limitations is also provided.

3. We developed a novel quantification measure based on the Hausdorff distance metric that can evaluate the performance of crack detection algorithms effectively. We compared it to four other quantification methods and demonstrated that our proposed method achieves better score separation between differently performing crack detection algorithms compared to the other methods. This method can also be used to assess the accuracy of algorithms used to detect other features that can be modeled as curves.

1.3 Organization of the Thesis

We begin by reviewing previous work in the area of crack detection in pavements and critical structures in Chapter 2. We also assess existing algorithms and motivate the

need to use a new algorithm based on minimal path techniques. Chapter 3 provides the literature review and technical background for minimal path methods. Then the detailed development of our algorithm for detecting complex topological curves is described step by step. In Chapter 4, we present a new quantification method to evaluate the performance of crack detection algorithms. This method is based on the buffered Hausdorff distance metric and it is compared to four other scoring measures. Chapter 5 contains experimental results for our method on pavement and structural crack images, and some medical images. It also describes performance analyses of the algorithm, obtained using synthetic data. The last chapter summarizes the presented work, draws conclusions from the thesis, and discusses possible directions for future work.

CHAPTER II

BACKGROUND OF THE PROBLEM OF CRACK DETECTION

This chapter briefly discusses the need for computer vision algorithms to track and detect cracks in critical structures like bridges, and pavements. In addition, there is a brief review of existing literature in these applications. The chapter also describes the limitations of the existing algorithms and the motivation for using the minimal path techniques for these applications.

2.1 Cracks in Critical Structures

Structural monitoring is needed for on-site inspection of critical structures like bridges, dams and sewer pipes. The potentially catastrophic consequences of fatigue cracking can be avoided by the early detection of fatigue cracks in on-site structures such as bridges. The frequently used method of inspecting bridge components for fatigue damage has been the elementary method of visual inspection. The most revealing sign of a crack is the existence of rust, oxide film and powder. However, since rust does not always appear immediately after a crack is formed, cracks may go undetected during visual inspection. Once a crack is observed or suspected, the structure is further tested to determine the extent or severity of the damage. A number of techniques are currently used to confirm the existence of a crack. Among them are vibration based methods, eddy current imaging, ultrasonic techniques [71], holographic interferometry, acoustic emission, photo elasticity, and Electronic Speckle Pattern Interferometry (ESPI). Of late, wireless sensor networks [22, 35] and mobile sensor networks [77] have been designed to monitor structures. Some of these techniques still require the initial

step of observation of a crack by a trained inspector. Additionally, to confirm the existence of a crack, many of these techniques require complex machinery, and/or cumbersome and expensive physical removal of the structural member in question. Visual inspection of structural components on-site at susceptible locations of the structure remains the first line of action. Remote monitoring used in conjunction with visual inspection has the potential to reduce the costs of bridge inspection and maintenance. Human intervention to manually analyze a significant number of images, as might occur during the imaging of large civil structures such as bridges, is a time-consuming and error-prone process. Therefore, computer vision algorithms can facilitate the process of on-site visual inspection. In particular, information on propagation of cracks across different time scales can be very helpful in determining the future course of corrective action.

The literature describes various computer vision algorithms that can potentially detect and track crack damage in structures. One approach consists of detecting cracks at each time frame independently. Qader et al. [1] did a comparison of four image analysis methods to detect cracks in bridge images: fast Haar transform (FHT), fast Fourier transform (FFT), Sobel edge detection, and Canny edge detection. Similarly, Hutchinon et al. [32] used Canny edge detector and wavelet based edge detector to detect cracks in bridges. Chen et al. [9] used the level-set contour-based approach to model the complex geometry and extent of cracks in concrete bridge images. The results of these algorithms are very sensitive to varying lighting conditions and noise levels of different images in a diverse image data set. Moreover, these algorithms do not exploit crack information available from previous time frames to track crack propagation in the current frame. Some methods described in the literature exploit temporal information across images at various times, and track crack propagation. Diaz et al. [20] studied propagation of cracks in concrete specimens using a non

contact digital image measurement system. This system incorporates a contrast correlation method to evaluate the level of plastic damage at each point of a specific area in the specimen. Two images of the specimen that are acquired before and after the introduction of fatigue deformation are used by the correlation method. Ryu and Nahm [51] observed crack propagation behavior successfully by combining the block matching method and the inclination threshold value method together. To monitor the crack changes (width and length), it is critical to transform the image co-ordinates of cracks extracted from each image into the fixed object co-ordinates of the concrete surface. In the study by Sohn et al. [27], such a geometric relationship was automatically recovered using the two-dimensional (2D) projective transformation based on the modified iterated Hough transform (MIHT) algorithm. MIHT automatically solves for the transformation parameters. One of the most popular optical methods used to measure structural deformation and displacements is the Digital Image Correlation (DIC) method [23, 50, 73, 72]. DIC is based on the maximization of a correlation that is determined by examining pixel intensity array subsets on two or more corresponding images and extracting the deformation mapping function that relates the images.

The above methods like DIC exploit temporal information between images at different time frames, and they are very effective in generating displacement fields for image registration. However, it is also important to find crack maps at various time instants to trace out the crack growth. The current research proposes an algorithm based on geodesic minimal paths to find crack maps that track the propagation of specific cracks that are identified at the crack initiation stage. This problem is solved together with the problem of crack detection in pavement images, which is discussed next.

2.2 Pavement Cracks

Pavement surface cracks measurement is an essential part of a pavement management system (PMS) for determining cost-effective maintenance and rehabilitation strategies. Visual surveys conducted by engineers in the field are still the most widely used means to inspect and evaluate pavements. However, such evaluations involve high degrees of subjectivity, hazardous exposure, and low production rates. Consequently, automated crack identification is gaining wide popularity among transportation agencies. For the past two decades, many researchers have been developing pavement distress detection and recognition algorithms using improved artificial and laser lighting. Still, fully automated pavement distress detection and classification has remained a challenge. According to the review by Transport Research Board Pavement Management Systems Committee [13], the current distress classification is highly subjective because of the lack of a standard classification protocol among all state agencies. Hence, it is difficult to achieve standardization in measuring the performance of any automatic distress classification system. Therefore, the current research focuses on the more fundamental step of crack map generation that can be standardized among different state transportation agencies. Crack map generation is defined as the process of extracting objects of interest, or cracks, from the background. All the pixels in the pavement image are identified as either crack or non-crack pixels. In the literature, this process is often called pavement *crack segmentation* and the two terms will be used synonymously in this document. A critical assessment of existing pavement crack segmentation methods was conducted as part of the research. Much of the discussion that follows is based on [57].

The National Cooperative Highway Research program synthesis document [14] contains a comprehensive summary on highway practice, research, and development efforts in the automated collection and processing of pavement condition data, which is typically used in network-level pavement management. It is in fully automated

methods of crack data reduction from images that the greatest amount of research and development work has occurred over the past decade. The survey noted several limitations of existing pavement crack segmentation methods. Firstly, all digital image analysis is limited by the quality and resolution of the images. The document notes that finer cracks that are not detected automatically may be detected manually because the human eye can perceive finer crack lines than the image can clearly display. For this reason, cracks with non-uniform widths may be identified as several shorter cracks. Secondly, certain types of pavement surface (e.g. chip seals), provide poor crack visibility, as does crack sealing material.

2.2.1 Crack Map Generating Algorithms

Thresholding and edge detection are the two principal crack segmentation approaches used. Thresholding-based segmentation methods are widely used in automated crack detection systems to extract pavement crack features from recorded images . These methods exploit the fact that cracks are darker than the background. Thresholding-based segmentation is widely used in automated systems to extract pavement crack features from recorded images . Kirschke et al. [40] presented initial work towards a histogram-based machine vision method for the automated sensing of unprepared cracks in highway pavement. In [42], a comparative study of different thresholding methods was done. These methods include the Otsu method, a regression method, a relaxation method, and Kittler's method. Koutsopoulos et al. [42] showed that the regression-based method is the best among these methods. After observing the difficulties of segmentation using the regression-based method, especially in presence of shadows, Oh et al. [47] proposed an iterated clipping method. This method iteratively uses mean and standard deviation values to eliminate noise while preserving crack pixels. A statistical approach was presented by Koutsopoulos and Downey [41] in which they recognized the imperfections of segmentation that cause difficulty in

distinguishing pavement crack types, especially between block and alligator cracks. Using the characteristics of pavement images, an innovative pavement crack segmentation algorithm based on mathematic morphology was proposed by Sun and Qiu [54]. Huang and Xu [30] implemented image processing algorithms in real time quite successfully which were a part of an automated pavement inspection system developed by TxDOT. In their algorithm, crack analysis is carried out on cells that are 8x8 pixels and each cell is classified as a crack or non-crack cell; then, its orientation is specified. These cells are connected together if the orientations of neighboring cells are similar and then crack information is extracted using parameters such as contrast and length. Cheng et al. [10] proposed a fuzzy logic-based method that exploits the fact that the crack pixels in pavement images are darker than their surroundings and are continuous. According to Cheng et al.'s analysis, there is inherent vagueness rather than randomness in pavement crack images because of grayness and spatial ambiguity in crack images. Consequently, conventional co-occurrence methods might not work well. Instead, they use fuzzy homogeneity as the criterion, and use fuzzy homogeneity vectors and fuzzy co-occurrence matrices to handle the grayness and spatial uncertainty among pixels. Cheng et al. [10] also did crack classification using fuzzy logic and neural networks and implemented a sound system that could run in real-time as part of a Utah Department of Transportation (UDOT) project. Recently, Hassani et al. (A. Hassani and Tehrani 2008) also used a fuzzy-logic based system for automatic pavement crack detection. Huang and Xu [30], as a part of an automated pavement inspection system developed by Texas Department of Transportation (TxDOT), implemented image processing algorithms in real time quite successfully. In their work, crack analysis is carried out on cells that are 8x8 pixels, where each cell is classified as a crack or a non-crack cell. The orientation of the crack cell is also specified. Kelvin Wang et al. [66, 62, 60, 63, 64, 65], as a part of a University of Arkansas initiative, also developed an automated real time pavement survey system.

In [62] and [63], the University of Arkansas team introduced a new automated system capable of collecting and analyzing pavement surface distress, primarily cracks, in real-time through the use of a high resolution digital camera, efficient image processing algorithms, and multi-computer, and multi-CPU based parallel computing. In [66] and [64], the image processing algorithms are assessed and a new low power laser based image acquisition technique is discussed. Hou et al. [28] assessed the feasibility of using 3D pavement stereo images for better automated crack analysis. Though the use of 3D stereo vision and laser technologies for pavement crack detection is still in preliminary stage, they have great potential of providing more information about the crack that can be used to generate an accurate crack map. These technologies can provide depth information of the pavement surface and researchers suspect that depth information will be more robust to texture, lighting and noise compared to 2D camera intensity images.

E1-Korchi et al [21] pointed out the importance of lighting in determining the fraction of distress that went undetected. Nazef et al. [46] from FDOT did a comprehensive evaluation of pavement distress systems looking into different fact.ors including spatial resolution, brightness resolution, optical distortion, and signal-to-noise ratio. Xu [69] as part of a TxDOT team posits that artificial lighting is the solution for eliminating all shadows in the image and for improving the data uniformity across different weather conditions. The TxDOT team has designed a Halogen light with a special reflector to accomplish this objective. Summarizing thresholding based method, we concluded that it is difficult to find a fixed thresholding criterion to find crack maps for diverse pavement images with different lighting conditions, shadows, oil stains and texture. Consequently, thresholding-based techniques have limitations.

Edge detection is another common image processing technique used for distress segmentation. Over the past 30 years, many simple but useful edge detectors have been developed and are widely used, such as Roberts, Sobel, Prewitt and LOG edge

detectors [17]. Another technique is Canny edge detection [8] that optimizes a performance index that favors true positive, true negative and accurate localization of detected edges. Canny's analysis, however, is restricted to a linear shift invariant filter. Motivated by the idea of Canny's edge detection algorithm, Mallat and Zhong [45] first proposed a decimated fast bi-orthogonal dyadic wavelet transform to obtain an image edge representation by computing the local maximum of the gradient of an image. Since then, this technology has been widely used in different areas. First, a pavement image is decomposed into different sub-bands by a wavelet transform. The wavelet modulus is calculated by combing the horizontal, vertical and diagonal details. Then, further analysis is carried out and edge information in each direction is extracted. Pavement images have various details at different scales of resolution and that has led to the popularity of wavelet-based edge detection methods in the last decade. Wavelet-based methods can analyze information at multiple scales simultaneously [15, 75, 74, 76]. The idea of Mallat and Zhong [45] provides the basis for most wavelet based algorithms used for pavement crack map detection. Recently Wang et al.[61] used an "atrous" algorithm based wavelet edge detection procedure for pavement distress segmentation. This algorithm is an un-decimated wavelet transform executed via a filter bank without sub-sampling. Wang et al. [59] developed a segmentation approach using fractal dimension. In order to improve the efficiency of the computational calculations, the improved approach of differential box-counting (DBC) was presented in the paper. As with the thresholding techniques, it is hard to find parameters to distinguish between edges, which correspond to cracks, and noise.

Outside the review of pavement crack specifically, a number of novel texture and distress segmentation techniques have been implemented. In [53], cell-segmentation is carried out using shape classification which is morphology driven. A two-phase approach to segmentation is used where an unsupervised clustering approach coupled with cluster merging based on a fitness function is used as the first phase to

obtain a first approximation of the cell locations. A joint segmentation-classification approach incorporating an ellipse as a shape model is used as the second phase to detect the final cell contour, which is compared to the ground truth. Tomczak and Mosorov [56], proposed a segmentation method based on Singular Value Decomposition(SVD), which does not require extensive test images. Texture defect detection has also been done using Gabor filters and its variations in literature [43, 48]. A method based on mathematical morphology and curvature evaluation which detects cracks in underground pipelines was proposed recently by Iyer et al. [33, 34]. They propose a three-step method to identify and extract cracks from contrast enhanced pipe images. Huo et al. [31] proposed a multi-scale detection of filamentary features using significance graphs. This method exploits beamlets-a dyadically organized, multiscale system of line segments-and associated fast algorithms for beamlet analysis to recover linear fragments from noisy images. Taking advantage of the new developments in mathematical statistics, a multi-scale approach is designed to detect filament or filament-like features in noisy images. Randen et al. [49] review most major filtering approaches to texture feature extraction and perform a comparative study. Filtering approaches included are ring/wedge filters, dyadic Gabor filter banks, wavelet transforms, wavelet packets and wavelet frames, quadrature mirror filters, discrete cosine transforms, eigen-filters, optimized Gabor filters, linear predictors, and optimized finite impulse response filters. Rivaz et al. [18] illustrate the use of complex dual tree wavelets in recovering edges in the presence of high noise. The complex wavelets are oriented in six different directions and the wavelet structure is able to segregate edges and noise effectively.

The overview of existing literature reflects some prominent areas of concern in the field of automatic pavement crack segmentation. Firstly, even though research is being carried out to improve image acquisition techniques and remove lighting defects, the performance of existing algorithms is severely hampered in the presence of shadows,

lighting effects and non-uniform crack widths. Full automation of pavement crack detection and classification has remained a challenge, especially for accurate and reliable segmentation. Secondly, many algorithms are able to perform well only in an image data set that has images that are not too different from each other. Otherwise, manual intervention is required to adjust the parameter inputs so that the algorithms can perform reasonably for different images. Lastly, algorithms that are able to tackle image variations and are robust to noise in image acquisition systems cannot be implemented in real time. As a result, these algorithms have to be used for off line testing until computationally faster machinery is employed in the future.

2.3 Assessment of Six Algorithms

During our research, six different algorithms for crack map generation were experimentally tested on pavement images to assess their strengths and weaknesses. Most of this section is based on [57], where all algorithms are described in extensive detail. The six algorithms investigated in the research were statistical thresholding [42], Canny edge detection [8], multi-scale wavelets method [45], crack seed verification method [30], iterative clipping method [47] and dynamic optimization-based method [3]. The results of these algorithms were quantitatively evaluated and compared with the ground truth cracks that were visually identified by GDOT pavement engineers. Dynamic optimization-based method formulates the crack map generation problem as an optimization task. Under all possible shapes and positions of the crack indication, one combination must be found that maximizes the score function *Crack*:

$$Crack = \underset{(n,p_1,\ldots,p_n)}{\operatorname{argmax}} f(p_1, p_2, \ldots, p_n), \tag{1}$$

where n is indication length (number of pixels) and the p_i's ($i = 1, \ldots, n$) are coordinates of the pixels along the crack indication. The function depends on four primary parameters: the minimum signal-to-noise ratio of the indications of interest

(controls algorithm sensitivity); the a priori probability of indication presence (influences detection of indications with gaps); the minimum and maximum widths of indication (define transversal resolution); and the minimal length of indication (defines longitudinal resolution). This algorithm [3], which has been used on digital radiography images of steel pipes, was used for the first time on pavement images.

There are two important characteristics of the algorithms that need to be analyzed: speed and accuracy. Overall, the accuracy of dynamic optimization-based method is consistently better than other methods. Figure 2.1 illustrates the superior crack map detection of the dynamic optimization-based method compared to the other five algorithms. The reason for the success of this method is the simultaneous use of both local and global properties of crack indicators. A global score function is maximized by the optimal use of connected pixels with predefined constraints like minimal crack length and crack width. However, the method only works for longitudinal cracks. Moreover, though this algorithm is more robust to noise and variation compared to the other five algorithms, the parameters in the algorithm need to be fine tuned heuristically if there is a larger set of diverse images, especially in a set containing images with different textures. Figure 2.2 depicts the result of dynamic optimization on an image with concrete texture where it fails to perform well. The statistical thresholding method uses only local intensity variation at each pixel location to classify a pixel as a crack or a non-crack pixel. Hence, it is able to perform well only when the intensity value difference between crack pixels and background pixels remains the same throughout the image. This problem is somewhat corrected in the iterative clipping algorithm by dividing images into tiles and segmenting the cracks independently in each tile. As thresholds are iteratively calculated using statistical parameters for each tile, the iterative clipping method gives reasonably good results for images that do not have shadows, high noise or extremely high variations

of pixel intensity values. The Canny edge detection algorithm fails to perform consistently well for all images because the optimal values of its parameters (edge strength and noise variance) are different for each image and the user has to choose these values heuristically. Another problem is that the use of a high noise variance value leads to detection of false edge contours because of the excessive smoothing effects of the Gaussian filter. These concerns make the algorithm unusable for images with high noise. A standard multi-scale wavelet algorithm is also unsuccessful in detecting cracks across images with varying conditions. The user has to adjust the parameters (numbers of scales, threshold values to separate edges from noise) manually according to the image being processed. Crack seed verification method performs reasonably well for images with uniform lighting conditions, but cracks in images with shadows and variable lighting are not segmented well. This is because of two reasons: the same crack seed verification threshold is used throughout the image, and there is no process in the algorithm to connect crack seeds together optimally.

Another downside of the dynamic optimization-based method is its high processing time (101 seconds per image) that makes the method unusable for real-time processing currently. Canny edge detection (1.6 seconds per image), the statistical thresholding method (0.8 seconds per image), the iterative clipping method (0.9 seconds per image) and the seed verification method (1.2 seconds per image) are all easy to implement in real time if the code is optimized. The multi-scale wavelet method is also slow to implement (50 seconds per image) and the implementation time increases with an increase in the number of scales.

All the above algorithms have input parameters that are sensitive to the lighting conditions, shadows, texture, oil stains and the noise in the image. These challenges make automatic crack map generation extremely difficult. To address a part of this problem, we used minimal path techniques for finding continuous cracks that extend over several image frames. Using overlapping images, crack information from one

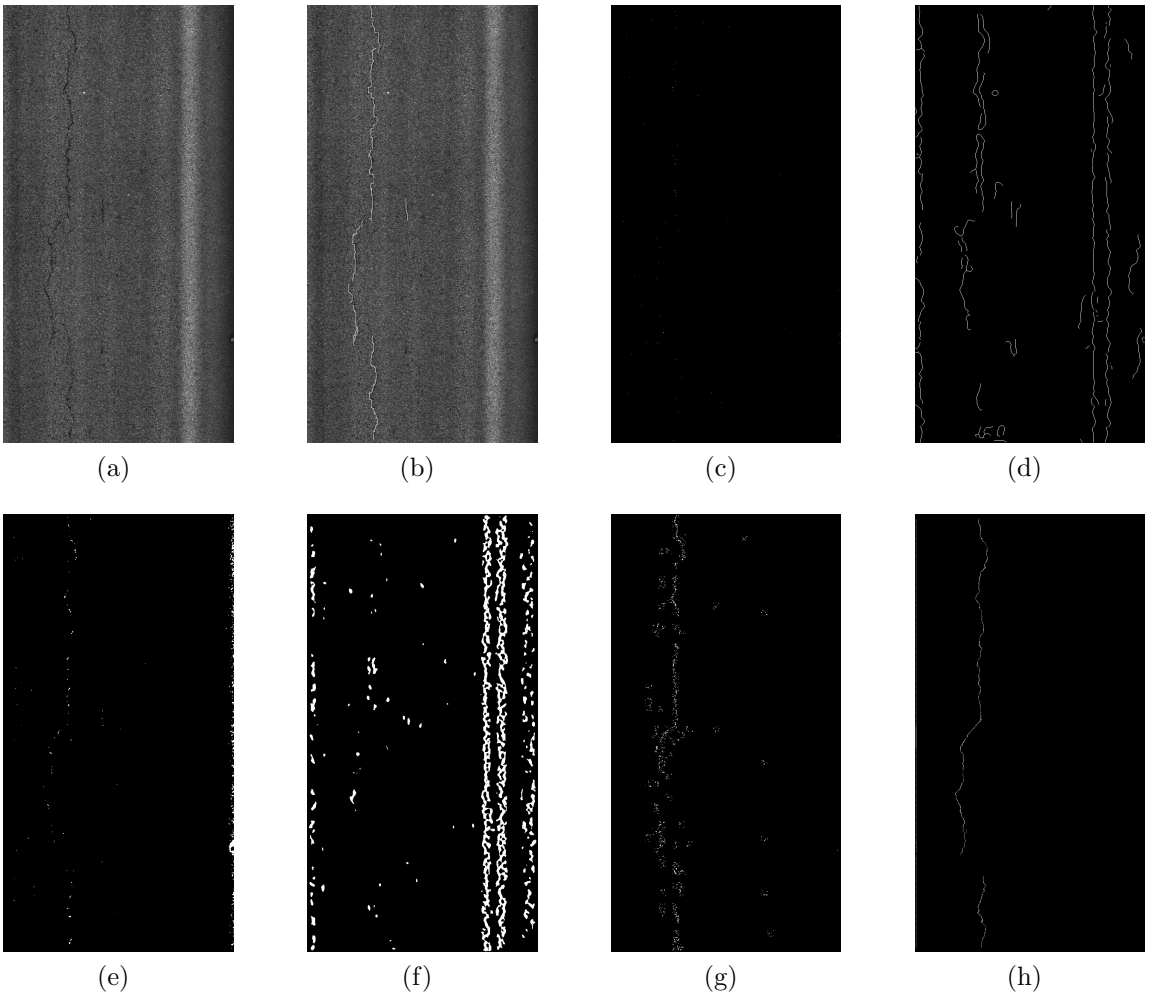
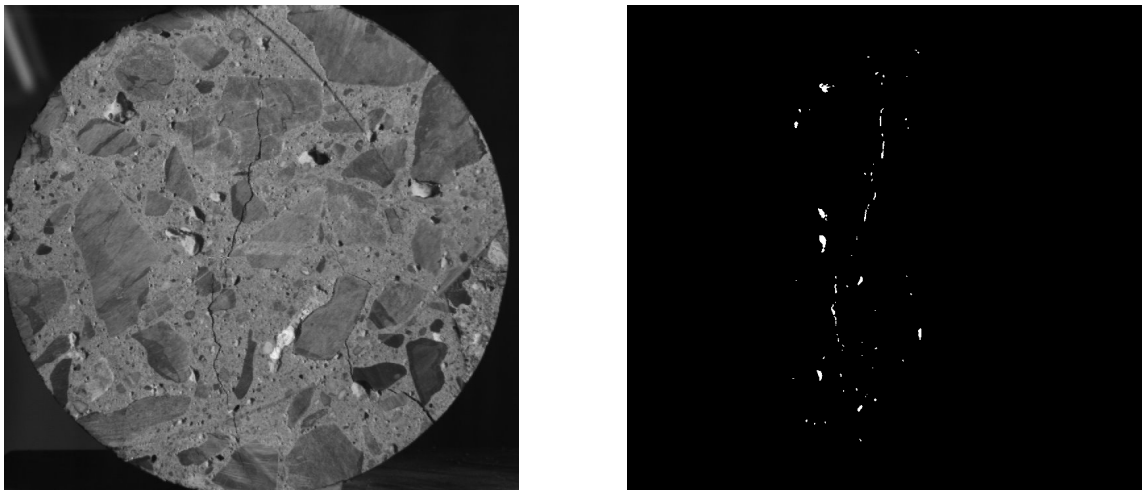


Figure 2.1: (a) Original image . (b) Ground truth image. (c) Statistical thresholding image. (d) Canny edge image. (e) Iterative clipping image. (f) Multi-scale wavelet image. (g) Crack seed verification image. (h) Dynamic optimization image.



(a) (b)

Figure 2.2: (a) Image with Concrete Texture. (b) Result of dynamic optimization-based method.

frame can be used to find cracks in subsequent frames. Other than that, minimal path techniques can be useful for semi-automatic crack map generation in pavement images where the user can provide inputs for crack end points. The research seeks to minimize user inputs and prior information requirements, while maintaining the accuracy of the results. Cracks occurring in both critical structures and pavements can be modeled as open curves under the minimal path formulation. Therefore, both problems will be tackled under the same general framework, which is explained in the next chapter.

2.4 Summary

This chapter identified the need for accurate crack map generation for both cracks in critical structures and pavements. For critical structures like bridges, the potential of finding crack growth by exploiting temporal information across structure images at multiple times is identified. For pavements, accurate crack map generation is useful both for crack sealing application and for accurate crack classification. Current research on pavement crack segmentation (crack map generation) algorithm

is comprehensively reviewed to identify the research need. As part of the research, six segmentation methods were tested using a diverse set of actual pavement images provided by the Georgia Department of Transportation (GDOT) with varying lighting conditions, shadows, and crack positions to differentiate their performance. The existing algorithms fail to perform on a diverse data set and this fact motivated the exploration of techniques based on minimal path techniques described next.

CHAPTER III

MINIMAL PATH TECHNIQUES

3.1 Introduction

Minimal path techniques are connected with variational energy minimization principles that were first introduced into computer vision by Terzopoulos et al. [36]. Terzopoulos et al. [36] proposed the active contour model or snakes that evolves an initially drawn curve in an image to minimize a certain energy functional. This energy consists of two terms: the external energy or the image feature term that guides the evolving curve towards the desired features or object boundary and the internal energy term that keeps the curve smooth as it evolves. However, the results of all active contour models depend on the local placement of the initial contour. Cohen and Kimmel [12] introduced a minimal cost path approach to tackle this problem. The minimal path approach captures the curve that minimizes the contour dependent energy between two user defined endpoints. The energy is selected such that it takes lower values at the desired feature of interest. For instance, an intensity based energy can be chosen to detect cracks because cracks are darker than the their immediate surroundings. A large number of researchers in different areas like geometric optics, computer vision, robotics and wire routing have previously considered such minimum-cost path problems. In computer vision, these problems have been tackled by graph search and dynamic programming principles. However, the graph based search method for a digitized image with a Cartesian sampling pattern finds only minimal path with segments that are vertical, horizontal or diagonal. Thus, graph search methods like Djiktra's algorithm suffer from inherent metrization errors, which have been documented by Cohen and Kimmell [12]. The minimal path method based

on active contours on the other hand, given its connection with active contours, formulates the problem in the continuous domain and is less susceptible for metrization errors.

The original minimal path technique can be extended for 3D tree-structured object extraction [19], but not for general 3D surface extraction. A topology-based saddle search routine is needed to extend the minimal path technique for closed curve extraction. Ardon et al. [4] proposed a more general scheme for 3D surface extraction between user supplied end-curves, but this scheme also requires the user supplied curves to be located precisely on the desired 3D boundary. Though they were introduced first as a boundary detection method, minimal path techniques have been successfully applied to problems such as contour completion [11], tubular surface extraction [19] and motion tracking [7].

All the above approaches require precise knowledge of the desired curve's endpoints (2D) or the desired surface's end curves (3D). In [6], a variant of the minimal path approach called Minimal Path Method With KeyPoint Detection (MPWKD) was devised to find curves under less restrictive prior information. The user needs to provide one endpoint on the curve that will lead to detection of representative keypoints along the curve using front propagation. For closed curves, a stopping criterion was derived that requires no further prior information from the user. Other researchers have used statistical shape priors [70, 25] and Principal Component Analysis (PCA) [44] based initial conditions for extracting organ shapes like lungs and kidneys as closed contours. However, for the simplest case of open curves, information about either both the endpoints or one endpoint plus the total length of the curve is required to find the complete curve [6]. For more complex open curves with multiple branches, all the endpoints need to be known up front. The motivation of the current work is to detect curves using less restrictive prior knowledge. In Section 3.2, the theory of minimal path techniques will be introduced. Some applications

to the problem of crack detection are shown. Section 3.3 will describe preliminary research efforts in investigating some novel methods of limited and varying success to detect curves with only one known point. Section 3.4 describes the most successful algorithm out of our investigations for detecting open curves with the knowledge of an arbitrary point (not necessarily an endpoint). Using this information, the algorithm can find the complete curve that includes endpoints and even branches. Section 3.5 will describe extensions of the algorithm to curves of more general topology that contain closed cycles as well as open segments. Finally, Section 3.6 presents two modifications of the general algorithm that make the curve detection algorithm more robust to noise and sharp corners at curve branches.

3.2 Theory of Minimal Path Techniques

The minimal path techniques capture the global minimum curve of a contour dependent energy between two user supplied endpoints. The active contour model combines an image feature term and a smoothing term in a composite energy functional

$$E(\gamma) = \alpha \int_0^1 \|\gamma'(s)\|^2 \mathrm{d}s + \beta \int_0^1 \|\gamma''(s)\|^2 \mathrm{d}s + \lambda \int_0^1 \Phi(\gamma(s))\mathrm{d}s, \tag{2}$$

where α, β and λ are real positive weighing constants, $\gamma(s) \in \mathbb{R}^n$ is a curve that is parameterized with respect to arc-length s, $\gamma'(s)$ and $\gamma''(s)$ are the first and second derivatives of the curve and Φ is a potential that depends on desired image features. In the minimal path technique, a simplied energy is chosen that is given by

$$E(\gamma) = \int_\Omega (\Phi(\gamma(s)) + \omega)\mathrm{d}s = \int_\Omega \tilde{\Phi}(\gamma(s))\mathrm{d}s, \tag{3}$$

where $E(\gamma)$ is energy computed along the curve, γ is the curve chosen in the domain Ω, ω is a real constant and $\tilde{\Phi} = \Phi + \omega$. For our problem description, we will assume that the potential Φ subsumes the constant ω and $\tilde{\Phi} = \Phi$. A potential $\Phi : \Omega \to \mathbb{R}^+$ is built from a given 2D or 3D image $I : \Omega \to \mathbb{R}^+$, where $\Phi > 0$, that takes lower values near the desired features of the image I. The choice of the potential Φ depends

on the application. For example, the potential function Φ for cracks can be taken to be a function of intensity value at each pixel because cracks are darker than the background. In some other applications, edge based potential functions can be used. The goal is to extract a curve that minimizes the energy functional $E : \mathcal{A}_{p_1,p_2} \rightarrow \mathbb{R}^+$ between two user defined points p_1 and p_2

$$E(\gamma) = \min_{\gamma \in \mathcal{A}_{p_1,p_2}} \int_\gamma \Phi(\gamma(s))\mathrm{d}s, \tag{4}$$

where $\mathcal{A}_{p_1,p_2}$ is the set of all paths connecting p_1 to p_2 and s is the arc length parameter. The curve connecting p_1 to p_2 that globally minimizes the energy $E(\gamma)$ is called the *minimal path* between p_1 and p_2 or the *geodesic*. The geodesic curve is denoted by C_{p_1,p_2}. The solution of this minimization problem is obtained through the computation of the minimal action map $U_1 : \Omega \rightarrow \mathbb{R}^+$ associated with p_1. The minimal action map is defined as the minimal energy integrated along a path between p_1 and any point x of the domain Ω:

$$\forall x \in \Omega, U_1(x) = \min_{\gamma \in \mathcal{A}_{p_1,x}} \int_\gamma \Phi(\gamma(s))\mathrm{d}s. \tag{5}$$

In this document, the minimal action map will also be called the *geodesic distance map*. The values of U_1 are the arrival times of a front propagating from the source p_1 with velocity $(1/\Phi)$ and $U_1(p_1) = 0$. U_1 satisfies the Eikonal equation:

$$||\nabla U_1(x)|| = \Phi(x) \quad \text{for} \quad x \in \Omega. \tag{6}$$

The minimal action map calculation can be generalized to the case of multiple sources. The minimal action map or the geodesic distance map associated with the potential $\Phi : \Omega \rightarrow \mathbb{R}^+$ with a set of n sources $S = \{p_1, \ldots, p_n\}$ is the function $U : \Omega \rightarrow \mathbb{R}^+$, where

$$\forall x \in \Omega, U(x) = \min_{1 \leq j \leq n} \{U_j(x)\}, \tag{7}$$

and

$$U_j(x) = \min_{\gamma \in \mathcal{A}_{p_j,x}} \int_\gamma \Phi(\gamma(s))\mathrm{d}s. \tag{8}$$

$U_j(x)$ is given by Equation (5) where p_1 is replaced by p_j. For all $p_j \in S, U(p_j) = 0$ and U satisfies the Eikonal equation given by Equation (6). Another important quantity important for the current work is the *Euclidean distance map*. It is the function $L : \Omega \to \mathbb{R}^+$ that assigns the Euclidean length of the minimal geodesic between x and the source S to every point x of the domain Ω.

$$\forall x \in \Omega, L(x) = \int_{C_{p_j,x}} \mathrm{d}s, \tag{9}$$

where

$$C_{p_j,x} = \min_{1 \leq i \leq n} C_{p_i,x}$$

is the minimum of all minimal paths from x to each source point p_i. $C_{p_i,x}$ is calculated using the potential Φ. In general if $\Phi \neq 1$ for all $x \in \Omega$, then the geodesic distance map U is always different from the Euclidean distance map L.

Centered finite difference schemes for solving the Eikonal equation are unstable. There are three algorithms described in [12] to compute the map U, and they are all consistent with the continuous energy formulation implemented in a rectangular grid. These three algorithms utilize level set methods, shape from shading methods, and Fast Marching methods. Fast Marching methods were favored because of their lower complexity compared to the other methods. Sethian [2, 52] proposed the Fast Marching method to solve the Eikonal equation that relies on a one-sided derivative that looks in the up-wind direction of the moving front. In the 2D case, at each point $x = (i, j)$, the numerical method gives the solution u approximating $U_{i,j}$:

$$\begin{aligned} &\left(\frac{\max\{u - U_{i-1,j}, u - U_{i+1,j}, 0\}}{\Delta x}\right)^2 + \\ &\left(\frac{\max\{u - U_{i,j-1}, u - U_{i,j+1}, 0\}}{\Delta y}\right)^2 = (\Phi(i,j))^2. \end{aligned} \tag{10}$$

Here Δx and Δy are the grid spacings in the x and the y direction respectively. This is the 4-neighbor scheme for the Fast Marching method. An alternate Fast Marching approach to solve Equation (10) was provided by Tsitsiklis [58] in the context of

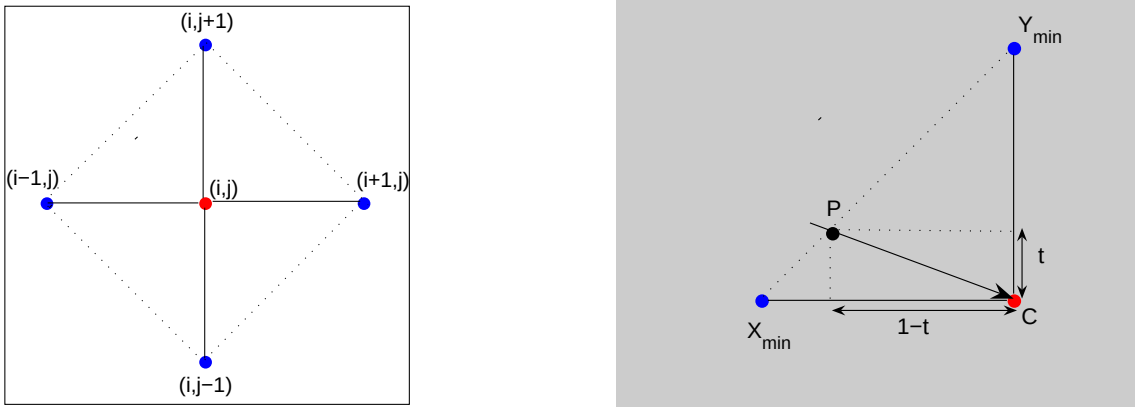
isotropic optimal trajectory problem in control theory. The fundamental difference between the control theory and the front expansion formulation is that the former interprets the starting point p_j as the exit point and tries to find the fastest way to reach p_j from x while the latter one looks for the time it will take to advance the front originating from p_j to pass point x. Both Fast Marching method methods compute the values of U in increasing order consistent with Huygens principle of wavefront propagation. These wavefronts propagate from the source set S with a velocity $(1/\Phi)$. The structure of the algorithm is almost identical to Dijkstra's algorithm for computing shortest paths on graphs. In the course of the algorithm, each grid point is tagged as either Solved (a point for which U has been computed and frozen), Active (a point for which U has been estimated but not frozen) or Unvisited (a point for which U is unknown). The set of Active points form an interface between the set of grid points for which U has been frozen (the Solved points) and the set of other grid points (the Unsolved points). This interface may be regarded as a set of fronts expanding from each source until every grid point has been reached. Given the source points, the Fast Marching algorithm helps to calculate geodesic distance map U and the Euclidean distance map L for every grid point. The improvement made by the Fast Marching algorithm was to introduce order in the selection of the grid points. The Fast Marching algorithm is computationally efficient because a priority queue can be used to quickly find the Solved points at each step. The Solved points are points with the smallest U (geodesic distance map) value among the current Active points . If Active points are ordered in a minimum heap data structure, the computational complexity of the Fast Marching method is $O(NlogN)$, where N is the total number of grid points. Benmansour and Cohen [6] compute the geodesic distance map U and the Euclidean distance map L, according to the discretization scheme for Fast Marching method given in [52] . A number of different modifications have been suggested to improve the accuracy of the Fast Marching method [16, 26]. For

example, an 8-neighbor scheme was proposed by Danielsson and Lin in [16]. These modifications adopt the control theoretic formulation of Tsitsiklis [58]. Therefore, we compute the maps U and L following the control theoretic formulation because these modifications can be used in future to improve the accuracy of L. In the next section, the calculation of the maps U and L for 4-neighbors is given.

Algorithm *FastMarchComputation*
Input: Image Im, potential function Φ and initial source point set S.
Output: Geodesic distance map U and Euclidean distance map L.
1. **for** all source points $p_1, \ldots, p_n \in S$
2. **do** Set $U(p_j) = 0, L(p_j) = 0$ and Status(p_j) = ACTIVE.
3. **for** all other points $x \in \Omega$
4. **do** Set $U(x) = \infty, L(p_j) = \infty$ and Status(p_j) = UNVISITED.
5. **while** Number of ACTIVE Points > 0
6. **do** Take out point w_{min} with minimum U.
7. Set Status(p_j) = UNVISITED.
8. Update neighbors (STATUS $\neq$ SOLVED) of w_{min} with procedure *UpdateNeighbors*.
9. **repeat**

3.2.1 Computation of maps U and L

In the Fast Marching algorithm, we first start with the L and U values at the source points in S to be zero and the status of the source points is set to Active. At all other points, U and L values are set to ∞ (a very high value) and their status is set to Unvisited. All the Active points are put in a minimum heap data structure according to the value of U. For each Fast Marching iteration, the grid point with the lowest U value, which is called w_{min}, in the minimum heap structure is removed from the heap and its status is changed to Solved. Next, the L and U values at the neighbors of the point are updated. This corresponds to 4-neighbors in the 2D case and 6-neighbors in 3D case. The algorithm stops once all points are labeled Solved. The Fast Marching algorithm is listed in *FastMarchComputation*. The *UpdateNeighbor* method is used to update the neighbor values at each iteration. Only neighbors that are Unvisited



(a) (b)

Figure 3.1: (a) Four neighbors of the current point c. (b) Triangle formed by x_{min} and y_{min}

or Active need to be updated. The *UpdateNeighbor* method for the 2D case will be given in this section. It can easily be extended to the 3D case. We show the *UpdateNeighbor* computation for one arbitrary neighbor of the point w_{min}, which we call c. This procedure is carried out similarly for the other three neighbors of w_{min}. Consider point c located at (i, j) and the four triangles formed by the neighbors of c as illustrated in Figure 3.1a. The Fast Marching boundary can propagate from any of the four directions given by the triangles. However, out of the four virtual triangles formed by the four neighbors of w_{min}, only one will give the minimum value at $U(i, j)$. The triangle that should be used for computation of $U(i, j)$ is the one formed by x_{min} and y_{min}. x_{min} is the minimum x-neighbor that corresponds to the x-neighbor $((i + 1, j)$ or $(i - 1, j)))$ that takes a lower U value. Similarly, y_{min} is the minimum y-neighbor that corresponds to the y-neighbor $((i, j - 1)$ or $(i, j + 1))$ that takes a lower U value. The triangle identified by this method is $x_{min}y_{min}c$ where c is the current point (i, j). This triangle is shown in Figure 3.1b. The steps in the *UpdateNeighbor* method are described next.

1. If x_{min} is Unvisited and y_{min} is Active or Solved,

$$U(c) = U(y_{min}) + \Phi(c). \tag{11}$$

$$L(c) = L(y_{min}) + 1. \tag{12}$$

2. If y_{min} is Unvisited and x_{min} is Active or Solved,

$$U(c) = U(x_{min}) + \Phi(c). \tag{13}$$

$$L(c) = L(x_{min}) + 1. \tag{14}$$

3. If x_{min} and y_{min} are not Unvisited, we compute point P, which is given by

$$P = (1-t)x_{min} + ty_{min} \tag{15}$$

where t is the parameter for linear interpolation that lies between 0 and 1. $U(c)$ is given by the solution of the following minimization problem in t:

$$U(c) = \min_{0 \le t \le 1} \{(1-t)U(x_{min}) + tU(y_{min}) + \sqrt{t^2 + (1-t)^2}\Phi(c)\}. \tag{16}$$

The parameter t that minimizes the expression in Equation (16) is called t_{min}. The minimum t_{min} occurs at the boundary points where t is 0 or 1, or it occurs at the local extrema t^*, which is obtained by differentiating the expression in Equation (16).

$$t^* = \frac{1}{2} + \frac{U(x_{min}) - U(y_{min})}{2D_1}. \tag{17}$$

where

$$D_1 = \sqrt{2\Phi^2(c) - (U(x_{min}) - U(y_{min}))^2}. \tag{18}$$

The two scenarios that arise are described below.

Case 1: If the discriminant $D_1 > 0$ and $0 \le t^* \le 1$, the parameter t_{min} is computed first. To identify t_{min}, the values $U_{t^*}(c)$, $U_0(c)$ and $U_1(c)$ are computed, which correspond to the values of the expression in Equation (16) at $t = t^*$,

$t = 0$ and $t = 1$ respectively. $U(c)$ is the minimum of the values $U_{t^*}(c)$, $U_0(c)$ and $U_1(c)$ and t_{min} is the parameter t that gives the value of $U(c)$. The value $U(c)$ is given by

$$U(c) = \min_{t \in \{0,1,t^*\}} \{(1-t)U(x_{min}) + tU(y_{min}) + \sqrt{t^2 + (1-t)^2}\Phi(c)\}. \quad (19)$$

The point that corresponds to the parameter value t_{min} in Equation (15) is called P_{min}. P_{min} is the point from which the Fast Marching front travels to the current point c. We can use the value t_{min} at the point P_{min} to compute the Euclidean distance L(c) value by replacing $\Phi(c)$ with 1 in Equation (19).

$$L(c) = (1 - t_{min})L(x_{min}) + t_{min}L(y_{min}) + \sqrt{t_{min}^2 + (1 - t_{min})^2}. \quad (20)$$

In the computation of L and U given in [6], which is based on the approach of Sethian [52], the computed value of $U(c)$ is the same as in Equation (19). However, the value of $L(c)$ is computed by the following equation:

$$L(c) = \frac{L(x_{min}) + L(y_{min})}{2} + \frac{D_2}{2}, \quad (21)$$

where

$$D_2 = \sqrt{2 - (L(x_{min}) - L(y_{min}))^2}. \quad (22)$$

According to the Tsitsiklis's control theory formulation that we follow, the computation of $L(c)$ shown in Equation (21) corresponds to the value $\bar{t}_{min}$ that minimizes the expression

$$L(c) = \min_{0 \le t \le 1} \{(1-t)L(x_{min}) + tL(y_{min}) + \sqrt{t^2 + (1-t)^2}\}. \quad (23)$$

Clearly, in general the value of $\bar{t}_{min}$ and t_{min} will be different. According to the definition of L given by Equation (9), t_{min} will lead to a more accurate computation of $L(c)$ compared to $\bar{t}_{min}$. The reason for that is we want to use the minimizer t_{min} of $U(c)$ to compute $L(c)$, not the minimizer of $L(c)$ (given

by $\bar{t}_{min}$). In addition, (18) and (22) demonstrate that discriminants $D1$ and $D2$ are formed by completely different terms. Therefore, it may be possible that $D1 > 0$ and $D2 < 0$. If that happens, $L(c)$ cannot be computed using the same neighbors that are used to compute $U(c)$. Therefore, it can lead to errors in the computation of $L(c)$. Our method avoids this problem by finding the location of the point P from which the geodesic front propagates. Hence, the method allows for the computation of L directly instead of solving for a separate quadratic equation. The accurate computation of L is essential for the accuracy of our proposed algorithm to detect curves, so we use this approach.

Case 2: If $D_1 \leq 0$ or $t^* \notin [0, 1]$, then we use the minimum of $U(x_{min})$ and $U(y_{min})$ to compute $U(c)$.

$$U(c) = \begin{cases} U(x_{min}) + \Phi(c) & \text{if } U(y_{min}) > U(x_{min}), \\ U(y_{min}) + \Phi(c) & \text{otherwise.} \end{cases} \tag{24}$$

$L(c)$ is computed using the same neighbors as $U(c)$

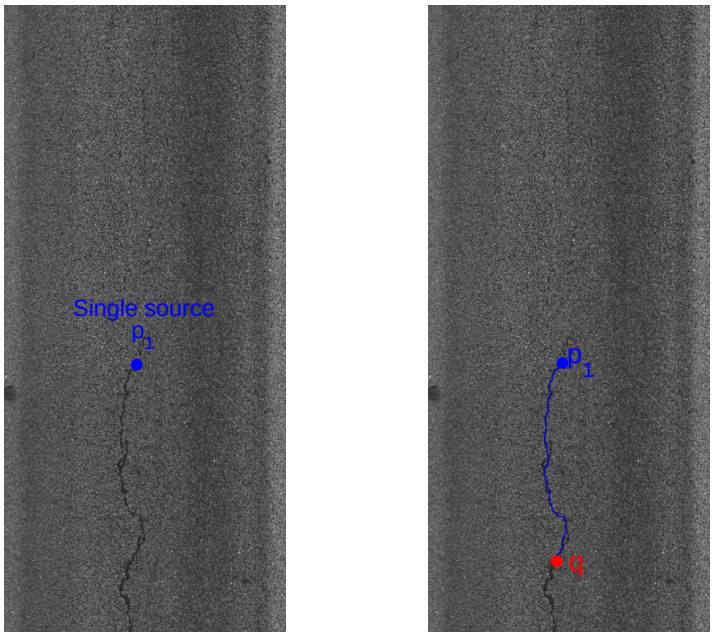
$$L(c) = \begin{cases} L(x_{min}) + 1 & \text{if } U(y_{min}) > U(x_{min}), \\ L(y_{min}) + 1 & \text{otherwise.} \end{cases} \tag{25}$$

3.2.2 Back Propagation

For a source point set S with a single point p_1, the minimum of U is at the front propagation starting point p_1 where $U(p_1) = 0$. The gradient of U is orthogonal to the propagating fronts since these are its level sets. Therefore, the minimal path between any point q and the starting point is found by sliding back along the gradient of the map U until arriving at $U(p_1)$. This back propagation procedure is a simple steepest gradient descent. It is possible to make a straight forward implementation on a rectangular grid: given a point q, the next point in the chain connecting q to p_1 is selected to be the neighbor p' of q for which $U(p')$ is minimum. The tracking

can be made more precise by using the Runge-Kutta method based gradient descent. The back propagation approach can be easily extended to the multiple source points $(p_1, p_2, \ldots, p_n)$ scenario. In the case of multiple source points, the back propagation procedure gives the minimal path between point q and the set of source points $S = \{p_1, p_2, \ldots, p_n\}$. In addition, we can also find the *origin point* $s^* \in S$ from which the minimal path to q originates. The minimal path back propagation procedure will be important for our algorithm and we will denote it by $MinimalPathbk(S, q)$, where S is the set of source points and q is the destination point.

3.3 Preliminary Investigation

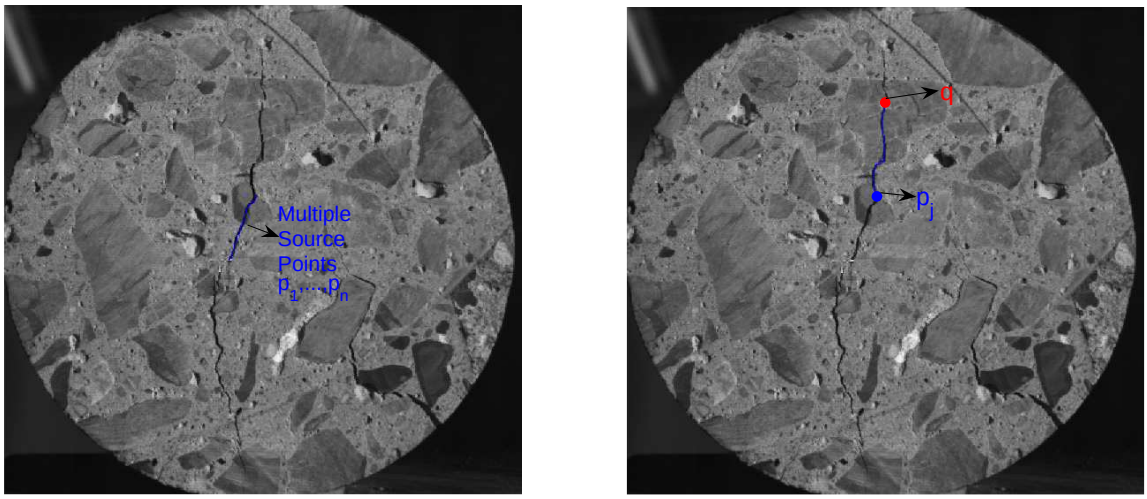


(a) (b)

Figure 3.2: (a) Pavement image with one source point p_1. (b) Back propagation from a point q.

We applied the minimal path algorithm to crack images on pavement and concrete structures. We used a potential function Φ based on intensity information. The potential function is given by

$$\Phi(i, j) = \epsilon + I(i, j), \tag{26}$$



(a) (b)

Figure 3.3: (a) Lab image with multiple source points $p_1, \ldots, p_n$. (b) Back propagation from a point q with origin point p_j (The point closest to q among $p_1, \ldots, p_n$).

where $I(i,j)$ is the image intensity value at each point $x = (i,j)$ and $\epsilon = 10^{-6}$ is a constant. Figures 3.2 and 3.3 show the results of the minimal path back propagation algorithm, $MinimalPathbk(S,q)$, for the single source and multiple source cases respectively. In Figure 3.3b, p_j is the origin point s^* that is closest to q . The grid spacings for the discretization step Δx and Δy were chosen to be 1. The raw image for Figure 3.2 is a pavement image containing cracks that was provided by GDOT. The image in Figure 3.2 was obtained from the experimental fatigue tests in Georgia Tech Structures lab that were conducted to simulate crack propagation in critical structures. The same potential function Φ was used for both images. The results indicate that the same potential function Φ can be used to find cracks in both these images. This potential function works better than functions based on edge detectors like $||\nabla I||$ because the images have high intensity variations in the non-crack regions. However, to find the crack, the two end points of the crack still need to be precisely known. This is a very restrictive condition for finding open curves or crack patterns. Therefore, our research was focused on finding cracks or open curves, using less prior information. A novel algorithm to detect open curves with only one known point

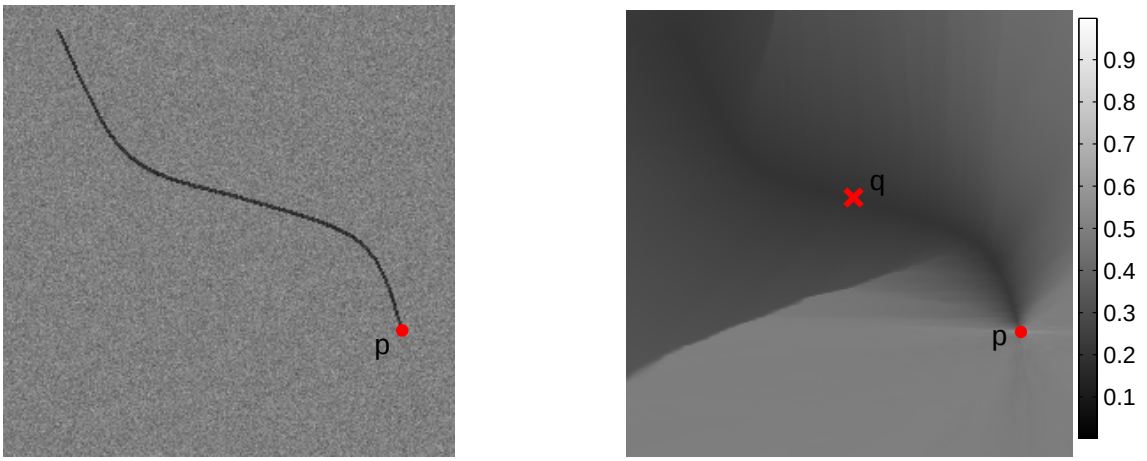
was developed as part of the research. We starting by investigating two parameters : geodesic to Euclidean length ratios and the local gradient around keypoints.

3.3.1 Geodesic to Euclidean length ratio

Given the geodesic and Euclidean distance maps U and L respectively, we can calculate U/L or the Geodesic to Euclidean length ratio. Considering the fact that the curve points or desired feature points have low potential function values, it is expected that the U values at curve points will be smaller than the L values, given that $0 < \Phi \leq 1$. This fact can be helpful in identifying the terminal point of the curve, if one or more points on the crack are already known. If U and L are set to the value 1 at the source points or the known curve locations, any continuous path along the points on the curve should continue to decrease the U/L ratio until it reaches the end of the curve. Therefore, the point at which the minimum value of U/L is reached can be identified as the endpoint of the curve. The computation of U/L ratio is done using the following steps:

- Start with known curve locations as source points and set U and L to 1 at these points.
- Follow the Fast Marching procedure given in Section 4.2 to compute distance maps U and L for all grid points.
- Compute U/L ratio for all grid points.

U/L ratios were computed for the some synthetic images first and the results on one of the synthetic image is shown in Figure 3.4b. The intensity value is chosen as the potential function. The curve in the image is of mean intensity 0.2 and variance 0.05 and the background is of mean intensity 0.5 and variance 0.05. Even on this image that has sufficient contrast, this method does not identify the correct endpoint location. Figure 3.4a shows the synthetic image with the initial source point p. Figure



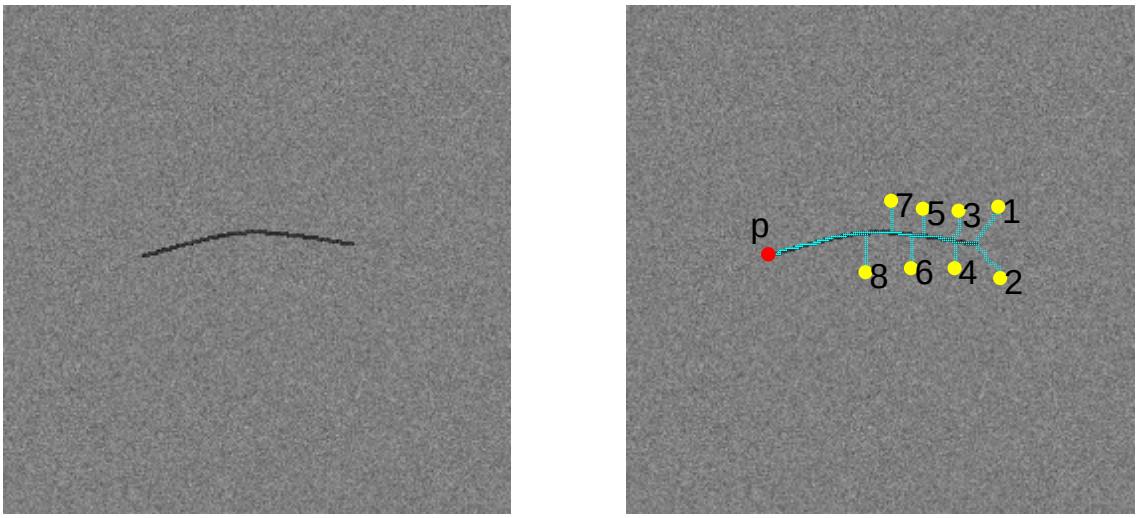
(a) (b)

Figure 3.4: (a) Image with source point p. (b) U/L ratio plot for the all Image pixels.

3.4b shows the U/L ratio plot at all pixel locations. The point with lowest U/L ratio is labeled q. The U/L ratio exhibits low values at curve locations, but the lowest value of U/L does not occur at the endpoint of the curve. The result can be attributed to two factors. In images, a continuous path of uniform decreasing U/L ratio may not be available as some noise or even small intensity changes along the curve may be present. Therefore, it is difficult to identify the correct endpoint of the curve. Secondly, though we assume the curve to be continuous, the U/L is computed using the Fast Marching discretization scheme. Even these small discretization errors can make the task of identifying correct endpoints difficult.

3.3.2 Local Gradient around Keypoints

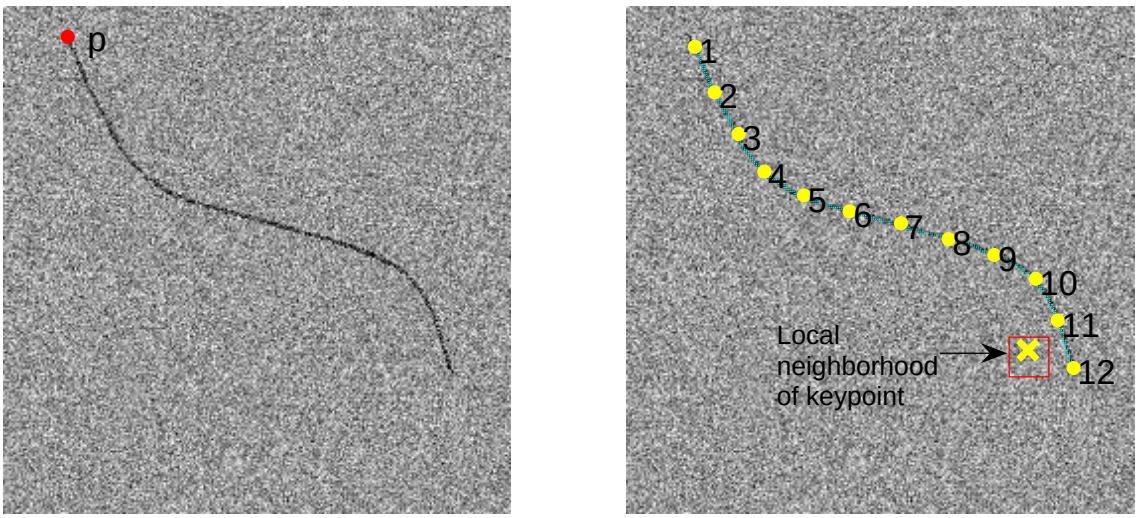
The minimal path procedure described above requires the knowledge of two endpoints. In addition, it also assumes that the potential function is not very noisy and provides enough contrast that can enable the minimal path to track even convoluted, long curves along desired features. In many applications, these conditions are not met and the desired feature points have a lower potential only in a local neighborhood region. To overcome the limitations of this approach, Benhamasour and Cohen [6] introduced



(a) (b)

Figure 3.5: (a) Image with a curve of interest. (b) Minimal paths from endpoint p to multiple points $1 \ldots 8$.

the Minimal Path Method With KeyPoint Detection (MPWKD) approach. This algorithm is based on the idea that among all points on the Fast Marching boundary that have equal geodesic distance U values, the points near the desired features will have the maximum Euclidean distance L. The path along the desired feature acts like an attractor for all minimal paths. In Figure 3.5, the minimal paths that have a common endpoint p on the curve are displayed. We can see that for all locations of the other endpoints labeled $1, \ldots, 8$, the minimal path is attracted to the desired curve. As the potential Φ at the feature points is lower than the neighborhood points, the Fast Marching boundary propagates with high speed $1/\Phi$ and travels the furthest Euclidean distance along the feature points. When the Fast Marching boundary propagates with the potential Φ from a source point set S ($S = p$ initially), the first point for which Euclidean distance L exceeds λ is identified. This point is referred to as a *keypoint*. Keypoints are recursively detected until a known endpoint is reached or until the detected curve exceeds a given Euclidean length. At each iteration, the source point set S is augmented by including the detected keypoint.



(a) (b)

Figure 3.6: (a) Image with a curve of interest and source point p. (b) Image illustrating source point and the keypoints on the curve in sequence. The keypoint on the random background is marked by 'x'

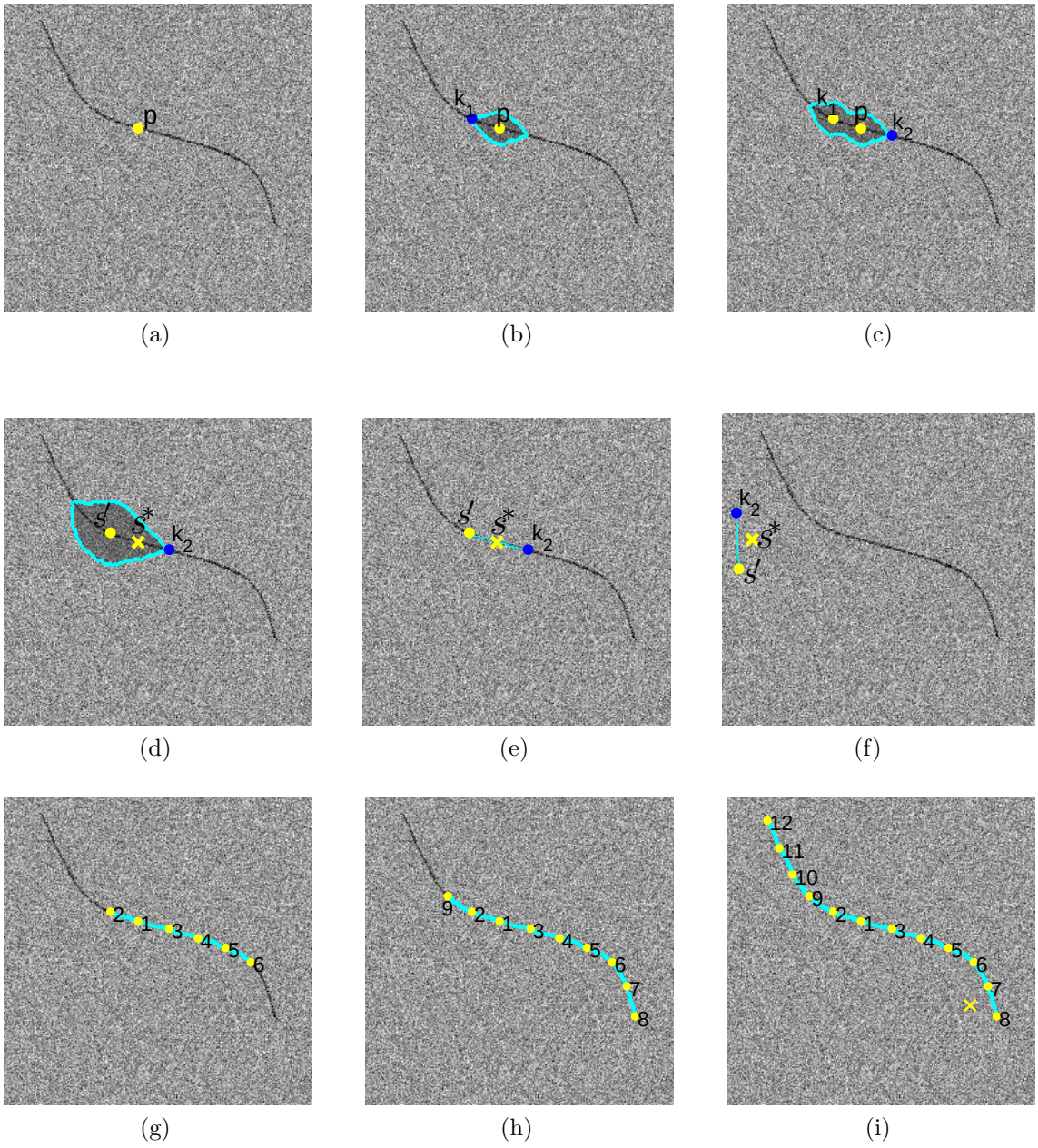
The MPWKD requires prior knowledge of both endpoints or one endpoint plus the total length of the curve. If the curve has multiple branches then the user needs to know all endpoints of the curve. To relax these prior assumptions, we investigated the gradient of L around each keypoint. The local gradient was computed by investigating the minimum L value in a rectangle of size 5×5 around the keypoint and subtracting it from the L value at the keypoint (keypoint has the highest L value). It is expected that when the keypoint is on the curve, the local gradient of L will be high because the local neighborhood will have pixels with substantially lower potential values than the keypoint. For a keypoint that is not on the curve but on a random background, the potential values in the neighborhood should not produce a substantial gradient. Figure 3.6a shows an image with the curve against a random background with an initial starting point. All the points that include the initial point p and keypoints on the curve are labeled in the sequence of their appearance in Figure 3.6b . The last keypoint identified lies on the random background and is marked as 'x'. The rectangular region that represents the local neighborhood of this keypoint is also

shown in Figure 3.6b.

We found out that it is not easy to choose a threshold gradient to identify the termination of the curve. Even if the curve intensity changes slightly keeping the background fixed, a new threshold needs to be set. After this initial investigation, we developed a novel algorithm to detect open curves (even with multiple branches) with the knowledge of an arbitrary known point. This algorithm is described next.

3.4 Algorithm for Detection of Open Curves

The *keypoints* detected in the MPWKD are approximately equally spaced along the curve. We use this fact to find a good stopping criterion for curve endpoint detection. Starting from the initial point p, fronts are propagated with potential Φ and the first point that crosses a Euclidean distance of λ is detected. This point is labeled as the first keypoint k_1. The Fast Marching procedure to find the first keypoint k_1 is referred as $FMM(S, \lambda)$ in this document where S is the source point set. The source point set S includes only the point p initially. We use a synthetic image that has a curve with lower mean intensity value than the random background for illustrating the algorithm. A λ value of 30 was used for this image and the potential function was chosen to be the intensity value. Figure 3.7a shows the image with initial source point p on the curve. Figure 3.7b illustrates the use of $FMM(S, \lambda)$ to find k_1. The points inside the Fast Marching boundary are scaled to a lower intensity value for better illustration. The source point set S is augmented by including the new keypoint k_1 and the set S now becomes $p \cup k_1$. An associative map $m : S \mapsto S$ is generated that defines an *origin point* for every keypoint in S. For a new keypoint, the origin point is the point closest in the set S to the keypoint. Hence, for k_1, the origin point is the initial point p. Though the initial point p does not have an origin point explicitly because it is not a keypoint, we assign k_1 as the origin point of p. The reason is that the minimal path for two points is symmetric and can be found by propagating the



(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

Figure 3.7: (a) Synthetic image with initial point p. (b) Detection of first keypoint k_1. (c) Detection of second keypoint k_2. (d) Fast March propagation from set s'. (e) Minimal path on the curve. (f) Minimal path in background region. (g) Curve and keypoints detected after 5 iterations. (h) Curve and keypoints detected after 8 iterations. (i) Final Image with ordered keypoints and terminating point marked by 'x'.

Fast Marching algorithm from either point (in this case p or k_1). Therefore, if started with k_1 as the initial point in source set S, p would be the keypoint determined by the procedure $FMM(S, \lambda)$. Hence

$$m(p) = k_1, \tag{27}$$

$$m(k_1) = p. \tag{28}$$

Now, the Fast Marching propagation is carried out from the updated set S and the next keypoint k_2 is detected as shown in Figure 3.7c. Using the minimal path backpropagation algorithm $MinimalPathbk(S, k_2)$, the point s^* in S that is a part of the minimal path from k_2 to S is identified. s^* is the origin point of k_2 and $m(k_2)$ equals s^*. In our example, origin point s^* is the initial point p . Next, we find the origin point s' of the point s^* from the map m (s' is the point k_1 for our example). If there exists a continuous path of desired feature points with low potential values between the points s' and k_2, then the minimal path between s' and k_2 should also pass through the vicinity of s^*. Figure 3.7e shows the minimal path from s' and k_2 in the case where the minimal path overlaps with the curve. When there is no portion of a curve on this minimal path, the path does not pass through the neighborhood of s^* as illustrated in Figure 3.7f. We denote the Euclidean distance from a source point set s' to a point k_2 as $L(s', k_2)$. The Fast Marching propagation required to determine $L(s', k_2)$ is denoted by $FMM(s', k_2)$. Figure 3.7d depicts this Fast March propagation. From Figure 3.7e, we conclude that if the minimal path between s' and k_2 lies on the curve, then

$$L(s', k_2) \approx L(s', s^*) + L(s^*, k_2). \tag{29}$$

As all keypoints are detected sequentially with a fixed Euclidean distance parameter λ, the Euclidean distance L between neighboring keypoints is close to λ. The point s^* is the closest point to the keypoint k_2 in the set S. Hence, by definition of the procedure $FMM(S, \lambda)$, the distance between s^* and k_2 is close to λ . Similarly,

The point s' should be at a distance of λ from s^* because s' is the origin point of s^*. This fact is clear from Figure 3.7e. According to Equation (29), the Euclidean distance $L(S', k_2)$ should then be approximately 2λ if the minimal path from S' to k_2 passes through s^*. If Equation (29) does not hold (like in Figure 3.7f), then the curve detection procedure can be terminated and the algorithm outputs that there is no curve detected. Otherwise, this procedure of finding keypoints is recursively carried out and the subsequent keypoints are identified as k_2 in the *OpenCurveDetection* algorithm that is given below. The point s^* determined at each iteration is either the initial source point p or a keypoint. A *tolerance error* value ϵ is specified for the termination. We used an ϵ value of 0.2λ or 10% of the total length 2λ for the algorithm. In Chapter 5, a sensitive analysis of the effect of ϵ and λ on curve detection is presented. The intermediate results of the algorithm are shown in Figure 3.7g and 3.7h. The complete curve with keypoints (including the initial point) is shown in Figure 3.7i. The keypoints are labeled according to the order of their appearance. The last keypoint for which (29) does not hold and the algorithm terminates is called the *terminating point* (represented by 'x' in Figure 3.7i). For a simple curve with no branches, the estimated length of the detected curve should lie within λ distance of the actual length.

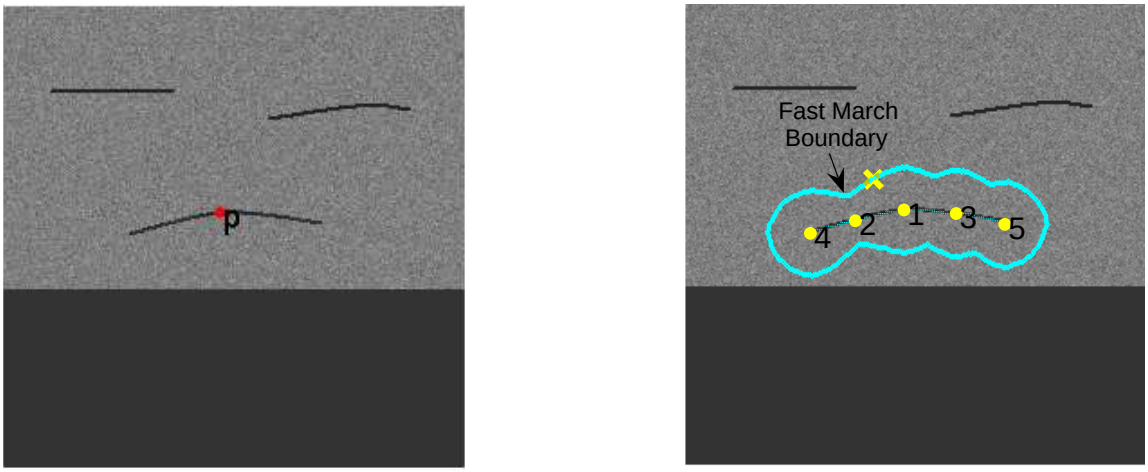
There are several advantages to the above algorithm. Firstly, if the curve representing desired features grows or expands at each time frame, we can use the detected keypoints from the previous time frame as the initial points for the next time frame. Then, the curve already detected will not be recomputed and only new portions of the curve will be added to the existing curve. This process can be very useful in applications of crack growth in structures where the crack grows at each time frame. The applications of the algorithm to the crack growth problem will be discussed in Chapter 5. Secondly, the algorithm based on keypoint detection requires that the potential chosen for the desired curve possesses sufficient contrast only in the local

Algorithm *OpenCurveDetection*
Input: Image Im, potential function Φ and initial source point set S.
Output: Detected Curve C

1. Start with initial source point set S containing an arbitrary point p on the curve.
2. Set $StopDetection = FALSE$.
3. Use $FMM(S, \lambda)$ to find keypoint k_1.
4. Run $MinimalPathbk(S, k_1)$ and initialize curve C to contain the minimal path between S and k_1.
5. Set $S \leftarrow S \cup k_1$.
6. Set $m(k_1) = p$ and $m(p) = k_1$.
7. **while** $StopDetection = FALSE$
8. **do** Run $FMM(S, \lambda)$ to find new keypoint k_2.
9. Run $MinimalPathbk(S, k_2)$ and find the origin point s^* in set S.
10. Set curve C' to contain the minimal path between S and k_2.
11. Compute point $s' = m(s^*)$.
12. Use $FMM(s', k_2)$ to calculate $L(s', k_2)$.
13. **if** $|L(s', k_2) - 2\lambda| < \epsilon$
14. **then** $C \leftarrow C \cup C', S \leftarrow S \cup k_2$ and $m(k_2) = s^*$.
15. **else** $StopDetection = TRUE$.
16. **repeat**

neighborhood of the curve. Even if the potential values are similar to the desired curve or there are undesired features similar to the curve in non-local regions, the algorithm is able to detect the desired curve. The reason is that the computation of L and U in the algorithm is confined to the local neighborhood around detected keypoints. This neighborhood is determined by the parameter λ and the neighborhood is larger for a bigger value of λ. Figure 3.8a shows a synthetic image with the desired curve in the center with a dark background and two undesired curves in the non-local neighborhood. The potential is chosen to be the intensity of the image. Figure 3.8b illustrates that the algorithm is still able to detect the desired curve. The Fast March boundary corresponding to the computation of U and L are also drawn in Figure 3.8b . Finally, the algorithm can be used to determine if there is any curve in the vicinity of the initial point as shown in Figure 3.7f.

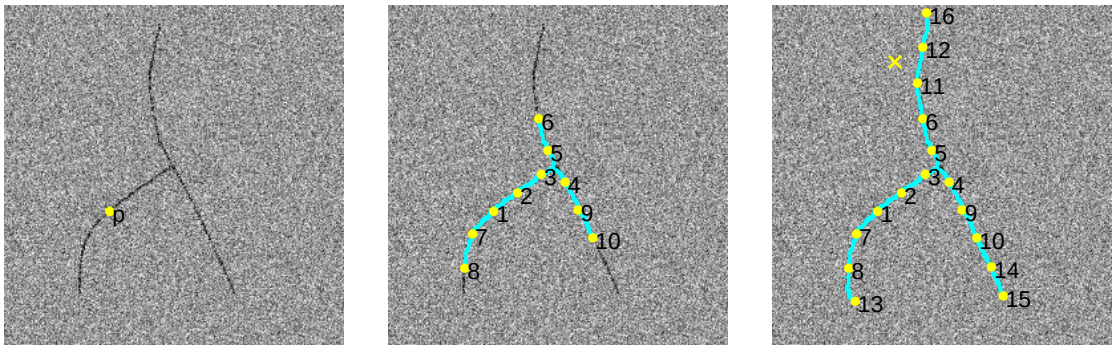
The above algorithm can also be used on more complex curves. Figure 3.9a illustrates a synthetic image that has a curve with branches embedded in a noisy



(a) (b)

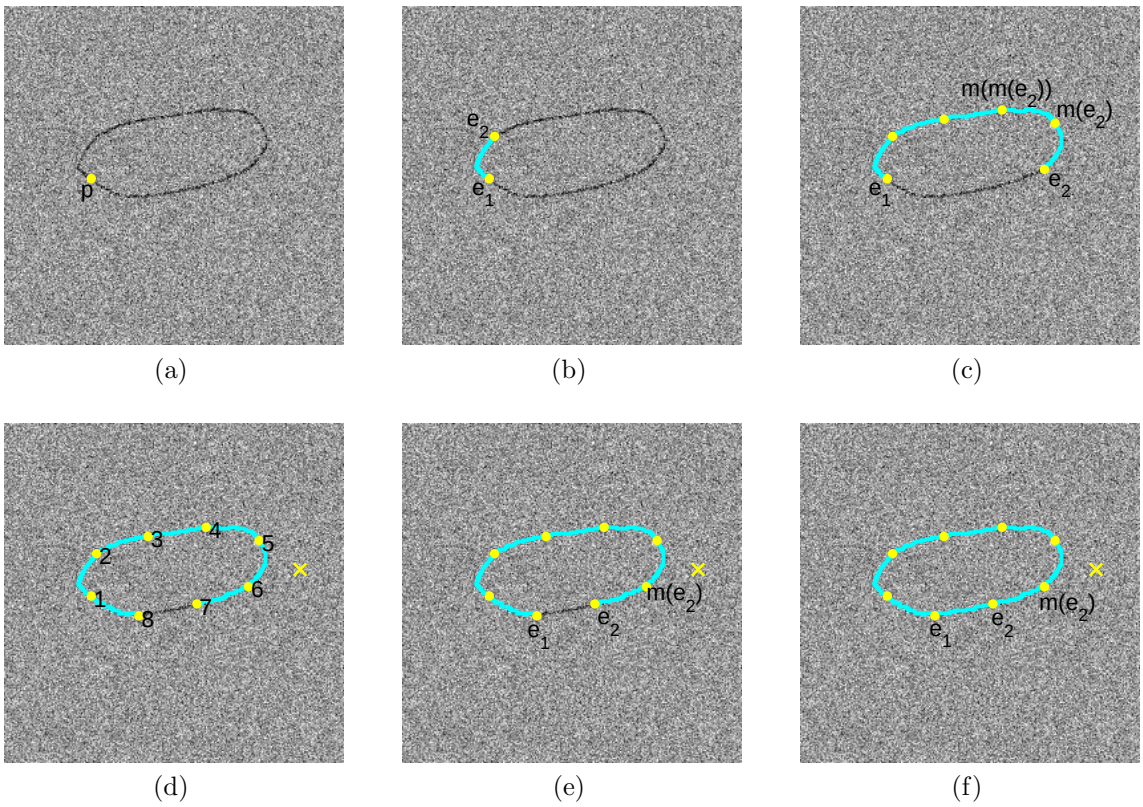
Figure 3.8: (a) Image with a curve of interest, and dark regions and undesired curves in non-local neighborhood. (b) Detected curve and Fast March boundary with ordered keypoints and terminating point labeled by 'x' .

random background of higher mean intensity. An arbitrary source point p on the curve is provided as input to the algorithm. Figure 3.9b demonstrates the intermediate result of the algorithm on the image in which the curve and the keypoints detected are labeled. Figure 3.9c shows the final result with the complete curve, all ordered keypoints and the terminating point. The next section explores the algorithm for more general curves that have cycles.



(a) (b) (c)

Figure 3.9: (a) Curve with branches and initial point p. (b) Intermediate result from the algorithm after 9 iterations. (c) Final curve detection with keypoints and final terminating point labeled by 'x'.



(a) (b) (c)

(d) (e) (f)

Figure 3.10: (a) Image with a closed curve of interest and initial point p. (b) Endpoints e_1 and e_2 after first keypoint detection. (c) Endpoints e_1 and e_2 and the origin point $m(e_2)$ after 5 iterations. (d) Detected curve from *OpenCurveDetection* with ordered keypoints and terminating point labeled by 'x' . (e) Endpoints e_1 and e_2 and the origin point $m(e_2)$. (f) Complete curve with closure between e_1 and e_2 .

3.5 Algorithm for Detection of Curves of General Topology

The self-terminating minimal path algorithm described in Section 3.4 can also be extended to curves of general topology that have cycles. For that we need to keep track of keypoints that are *endpoints*. Endpoints are defined as points that have only one neighbor. Two points are neighbors of each other if one point is the origin point of the other. For every point in the source point set S, which is augmented each time a new keypoint is added, the origin point is given by the map m as explained in Section 3.4. The map m is one-to-one, so every point in S has an origin point. Hence, every point in S has at least one neighbor. For any general point q, $m(q)$

and q are neighbors. We illustrate the algorithm on a synthetic image with a closed curve and a starting point p as shown in Figure 3.10a. When the first keypoint k_1 is detected starting from initial point p, both points are designated as endpoints and added to the endpoint set E as shown in Figure 3.10b. These endpoints are labeled as e_1 (which is same as p) and e_2 (which is same as k_1) in Figure 3.10b. For all subsequently detected keypoints, it is easy to update the endpoint set E. Whenever a new keypoint (identified as k_2 in *OpenCurveDetection* algorithm and e_2 in Figure 3.10c) is detected from the current source point set S and the keypoint satisfies Equation (29) of the *OpenCurveDetection* algorithm, the keypoint is also added to the endpoints E. This is because every new keypoint e_2 has only one neighbor: the origin point of e_2 given by $m(e_2)$. When a new keypoint originating from the point $m(e_2)$ contained in S is added to the set S, the point $m(e_2)$ has a new neighbor e_2. All points in the source point set S have at least one neighbor, which is given by the map m. Hence, the point $m(e_2)$ has at least two neighbors now given by $m(m(e_2))$ and e_2 as shown in Figure 3.10c. If point $m(e_2)$ is in the endpoint set E (that is it had only one neighbor $m(m(e_2))$ previously), $m(e_2)$ should be removed from the set E. For example, in Figure 3.10c, $m(e_2)$ was a part of endpoint set E previously and it has to removed after the detection of the new keypoint e_2. Apart from this additional step of keeping track of endpoints, we follow the *OpenCurveDetection* algorithm exactly as before. When the algorithm *OpenCurveDetection* terminates, we examine all pairs of endpoints in the set E. Figure 3.10d illustrates the results of the *OpenCurveDetection* algorithm with the keypoints and initial point marked in order. Let e_1 and e_2 be any two points contained in the endpoint set E, and $m(e_2)$ be the origin point of e_2 as shown in Figure 3.10e. A continuous curve between e_1 and e_2 will exist only if the distance $L(e_1, e_2)$ is less than 2λ. This is because all keypoints are at a minimum distance of λ from each other and a keypoint cannot be detected on the curve between e_1 and e_2 with the total separation between e_1 and e_2 less than 2λ. If the distance

$L(e_1, e_2)$ is more than 2λ, an additional keypoint that lies on the curve between e_1 and e_2 should be detected. For the curve to exist between e_1 and e_2, an additional condition similar to Equation (29) should be satisfied. The condition is given by the following equation:

$$L(e_1, m(e_2)) \approx L(e_1, e_2) + L(e_2, m(e_2)). \tag{30}$$

By the definition of the origin point $L(e_2, m(e_2))$ is close to λ, so only the other two quantities need to be computed in the above equation. If Equation (30) is satisfied within a certain tolerance range, the detected curve is closed between endpoints e_1 and e_2 as shown in Figure 3.10f. This process is repeated for all pairs of points in E (there are only two endpoints in the set E for the image in Figure 3.10f). The same tolerance value ϵ is used for both Equations (29) and (30) in our algorithm. If the user has prior knowledge that the curve is closed, the keypoint verification given by Equation (29) can be dropped and unconstrained keypoint detection can continue until the two endpoints (there are only two endpoints for a simple closed curve) become closer than 2λ. However, in our algorithm we do not assume this prior knowledge.

3.6 Robust Algorithm for Curve Detection

In our experimental tests, we found that some modifications of the algorithm described in Section 3.5 can be made to improve the robustness of curve detection. During the *GeneralCurveDetection* algorithm, there is a possibility of generating additional keypoints that are not on the curve after all the keypoints on the curve have been identified. Figure 3.11a shows a synthetic image with a closed curve and Figure 3.11b depicts the results of *GeneralCurveDetection* with parameter $\lambda = 30$. Point '14' in Figure 3.11b is the additional keypoint that is generated and it produces spurious curve portions. We observed that these additional points that lead to spurious curve detection have an origin point that has at least 3 neighbors. Such points are called

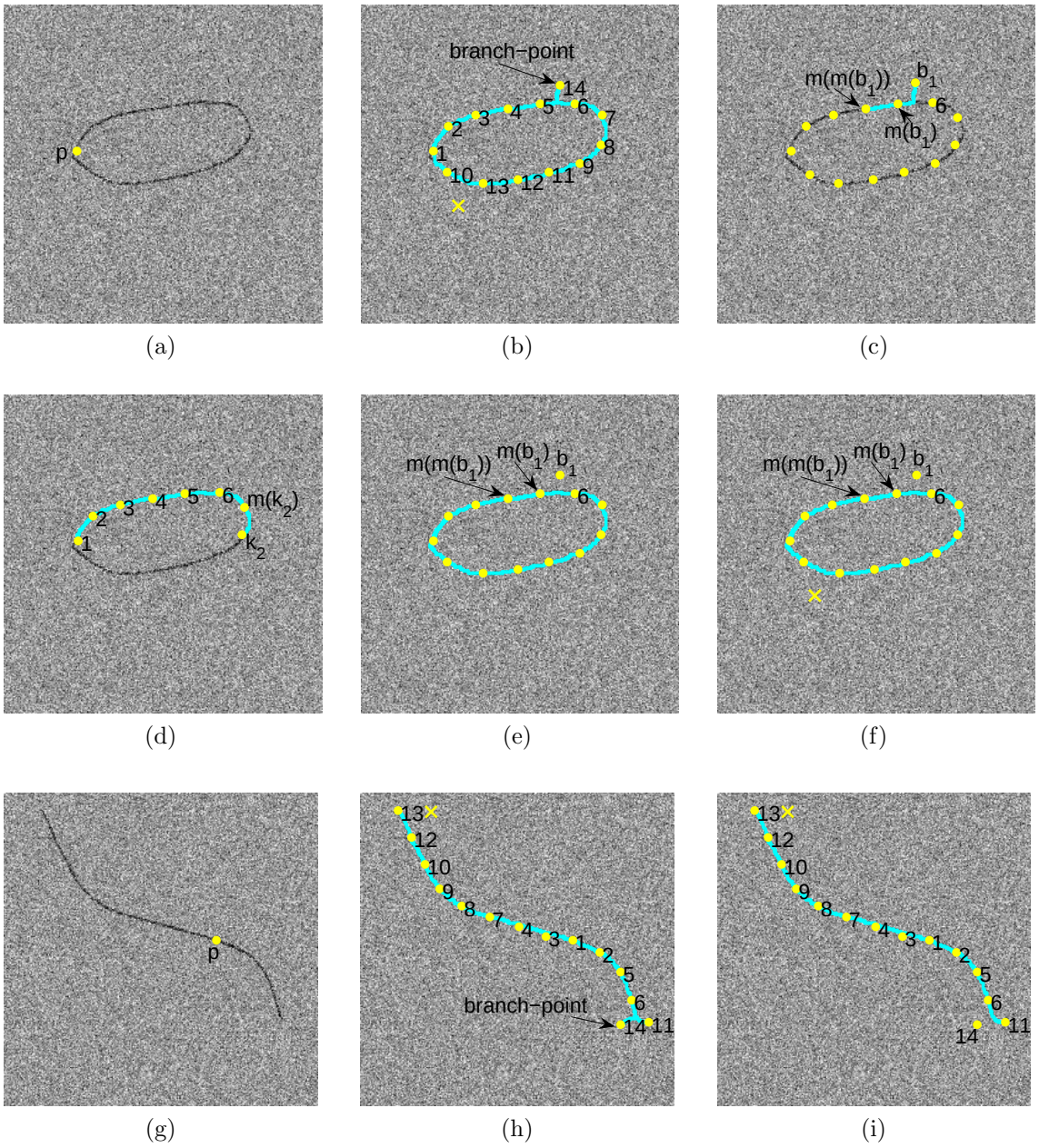


Figure 3.11: (a) Image with a closed curve of interest and initial point p. (b) Curve, ordered keypoints and terminating point (marked by 'x') detected by *GeneralCurveDetection*. (c) Minimal path between branch-point b_1 and $m(m(b_1)$. (d) Keypoint k_2 for which the origin point $m(k_2)$ has only 2 neighbors. (e) Branch-point b_1 for which the origin point $m(b_1)$ has 3 neighbors branch-point b_1, point '6' and the origin point $m(m(b_1))$. (f) Complete curve detected by modified algorithm with disconnected branch-point. (g) Image with open curve of interest and initial point p. (h) Curve, ordered keypoints and terminating point (marked by 'x') detected by *GeneralCurveDetection*. (i) Complete curve detected by modified algorithm with disconnected branch-point.

Algorithm *GeneralCurveDetection*
Input: Image Im, potential function Φ and initial source point set S.
Output: Detected Curve C

1. Start with initial source point set S containing an arbitrary point p on the curve.
2. Set $StopDetection = FALSE$.
3. Use $FMM(S, \lambda)$ to find keypoint k_1.
4. Run $MinimalPathbk(S, k_1)$ and initialize curve C to contain the minimal path between S and k_1.
5. Set $S \leftarrow S \cup k_1$.
6. Set $m(k_1) = p$ and $m(p) = k_1$.
7. Set $E \leftarrow p \cup k_1$.
8. **while** $StopDetection = FALSE$
9. **do** Run $FMM(S, \lambda)$ to find new keypoint k_2.
10. Run $MinimalPathbk(S, k_2)$ and find the origin point s^* in set S.
11. Set curve C' to contain the minimal path between S and k_2.
12. Compute point $s' = m(s^*)$.
13. Use $FMM(s', k_2)$ to calculate $L(s', k_2)$.
14. **if** $|L(s', k_2) - 2\lambda| < \epsilon$
15. **then** $C \leftarrow C \cup C', S \leftarrow S \cup k_2$ and $m(k_2) = s^*$.
16. Set $E \leftarrow E \cup k_2$ and $E \leftarrow E - m(k_2)$.
17. **else** $StopDetection = TRUE$.
18. **for** all pairs of points e_i, e_j in E,
19. **if** $L(e_i, e_j) < 2\lambda$
20. Run $MinimalPathbk(e_i, e_j)$ and initialize C' to contain the minimal path between e_i and e_j.
21. **if** $|L(e_i, m(e_j)) - L(e_i, e_j) - \lambda| < \epsilon$
22. **then** $C \leftarrow C \cup C'$
23. **repeat**

branch-points. In Figure 3.11c, the branch-point is labeled b_1, which has origin point $m(b_1)$. The point $m(b_1)$ has 3 neighbors that includes its origin point $m(m(b_1))$, the branch-point b_1 and the point '6', which is the sixth keypoint detected by the *GeneralCurveDetection* algorithm. For the branch-point b_1, Condition (29) is satisfied and *GeneralCurveDetection* algorithm is not terminated. This is because b_1 is located in a neighborhood of the curve section between $m(b_1)$ and point '6'. Hence, a major portion of the minimal path between b_1 and $m(m(b_1))$ lies on the actual curve and passes through the vicinity of $m(b_1)$ as shown in Figure 3.11c . This ensures that Equation (29) is satisfied for the point b_1. Here points b, $m(b)$ and $m(m(b))$ correspond to k_2,

s^* and s' in (29). To overcome the problem of detection of spurious branch-points, we keep track of the spurious branch-points using a set B in the modified algorithm. A map $m_1 : S \mapsto Z$ is generated for all the points in the source point set S. This map gives the number of neighbors for each point in S. Whenever a new keypoint k_2 is generated using the procedure $FMM(S, \lambda)$, the neighbor count given by $m_1(k_2)$ is set to 1. In addition, the neighbor count for the origin point of the new keypoint is incremented by 1. For a keypoint k_2 that is not a branch-point (its origin point $m(k_2)$ has only two neighbors) like in Figure 3.11d, the earlier algorithm procedure given in *GeneralCurveDetection* is followed without changes. However, if the neighbor count of the origin point $m(k_2)$ is greater than 2 , the new keypoint is identified as a branch-point (which is b_1 in Figure 3.11e) and added to the branch-point set B. It is also added to the source point set S, but not to the endpoint set E. The minimal path from the branch-point b_1 and its origin point $m(b_1)$ is not added to the complete curve so that spurious curve sections are not detected. A map $n : B \mapsto \mathcal{A}_I$ is generated for all branch-points in B where n stores the minimal path between a branch-point b and its origin point $m(b)$ for all $b \in B$. $\mathcal{A}_I$ is the space of continuous curves in the grid generated by the input image I. We also wanted to ensure that if a branch-point b in B corresponds to a genuine branch of the curve, we add the minimal path given by $n(b)$. Therefore, whenever a new keypoint k_2 has an origin point b in set B, both the minimal path from k_2 to b and the minimal path from $m(b)$ to $m(m(b))$ are added to the detected curve. In addition, the origin point b is removed from the set B. Figure 3.11f shows the complete curve detected using the modified algorithm where the branch-point curve section is not detected. Figure 3.11g illustrates another image for which the *GeneralCurveDetection* algorithm detects spurious curve portions. Figure 3.11h shows the results of *GeneralCurveDetection* algorithm and Figure 3.11i shows the corrected curve detection results using the modified algorithm.

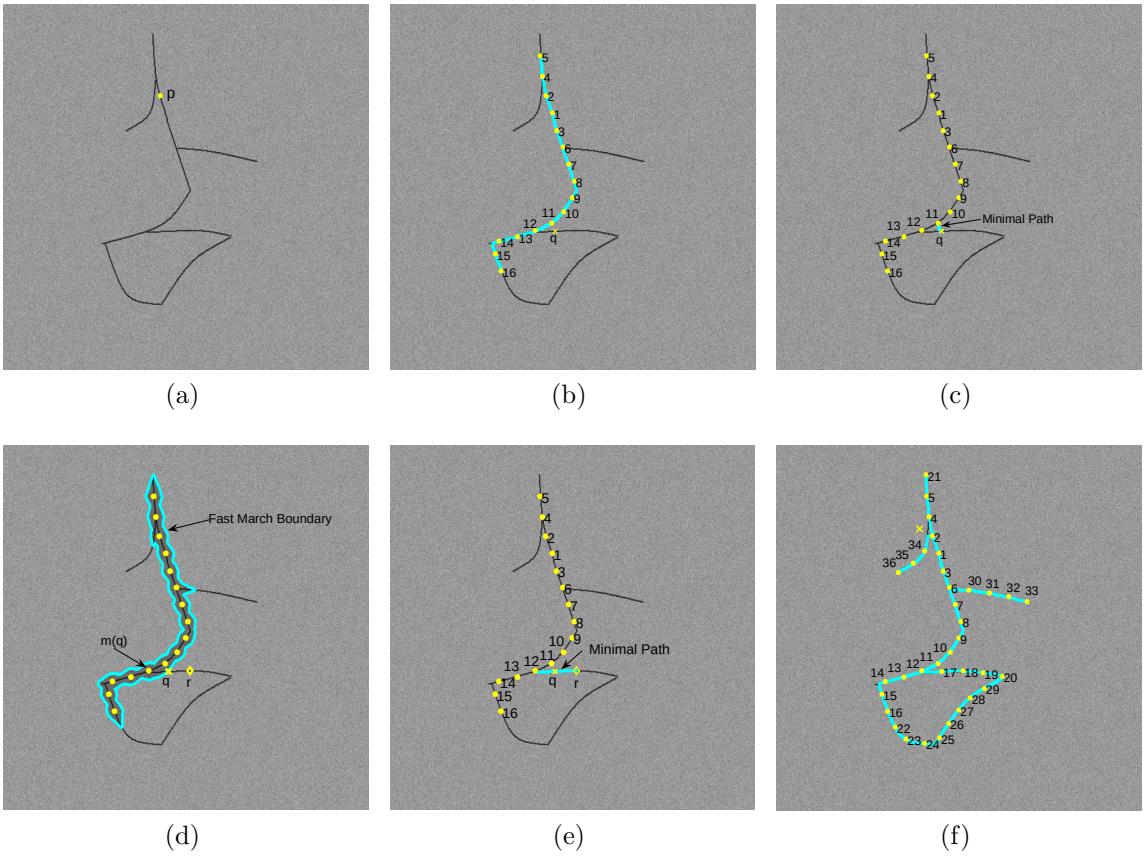


Figure 3.12: (a) Image with complex topological curve and initial point p. (b) Curve, ordered keypoints and terminating point (marked by 'x') detected by *GeneralCurveDetection*. (c) Minimal path between terminating point q and $m(m(q))$. (d) Incremental keypoint r and Fast Marching boundary of $FMM(S, \lambda)$ (e) Minimal path between $m(q)$ and r. (f) Complete curve detected by *RobustCurveDetection* with ordered keypoints and terminating point marked by 'x'

Apart from the modification described above, we identified another area of algorithm improvement. We found that for complex topological curves that have sharp corners at branches the *GeneralCurveDetection* algorithm terminates before the detection of the complete curve. This is illustrated by Figure 3.12a that shows a synthetic image with a complex topological curve with cycles and branches. Figure 3.12b depicts the curve detection result of *GeneralCurveDetection* algorithm with ordered keypoints and the terminating point (marked as 'x'). A λ value of 30 was used for the

algorithm. Figure 3.12b shows that the complete curve is not detected and the algorithm stops after the detection of the terminating point, which is located on a branch with a sharp corner. This is because though the potential image (the image intensity value is used as potential for the image) provides a good contrast between the curve and the background, there exists a shorter path between the terminal point, which is called q, and $m(m(q))$ that does not pass through $m(q)$. This fact is shown in Figure 3.12c where $m(m(q))$ is point '11' and $m(q)$ is point '12'. Hence, $L(m(m(q)), q)$ is not close to 2λ and Condition (29) is violated and the *GeneralCurveDetection* algorithm is terminated. q, $m(q)$ and $m(m(q))$ correspond to k_2, s^* and s' in (29). To fix this problem, we introduced an extra condition for algorithm termination. We compute an additional keypoint r and call it the *incremental keypoint*. This point r is computed using the Fast Marching procedure with the terminal point q as the source point. The incremental keypoint r is the first point for which the Euclidean distance L is greater than λ. However, this keypoint r should also lie outside the Fast Marching boundary derived from the procedure $FMM(S, \lambda)$. Recall that $FMM(S, \lambda)$ is the Fast Marching procedure used to compute the terminal keypoint q in the *GeneralCurveDetection* algorithm where S is the source point set containing all keypoints apart from q. The incremental point r and the Fast Marching boundary for the procedure $FMM(S, \lambda)$ is shown in Figure 3.12d. In the figure, the region inside the Fast Marching boundary is scaled to a lower intensity value for better illustration. The status of all the grid points in an image after the execution of the $FMM(S, \lambda)$ Fast Marching procedure is stored in an array called *STATUS*. Points inside the Fast Marching boundary of $FMM(S, \lambda)$ correspond to the Solved points in the Fast Marching algorithm described in Section 3.2. Hence, the incremental point r should not be a Solved point in the *STATUS* array. This condition ensures that the new keypoint r originates from the terminal point q and does not correspond to other keypoints on the curve that are already computed. The Fast Marching procedure to compute the incremental

keypoint r is called $FMM(q, \lambda, \text{STATUS})$. If a condition similar to (29) given by

$$L(m(q), r) \approx L(m(q), q) + L(q, r), \tag{31}$$

is satisfied then the curve detection algorithm is not terminated. This condition ensures that if the incremental keypoint r lies along a branch of the curve as shown in Figure 3.12e, the curve detection procedure continues. Since the new point r is restricted to lie outside the Fast Marching boundary of $FMM(S, \lambda)$, we found that a lower tolerance value should be used for the validating Condition (31) compared to Condition (29). According to the results of the quantitative validation study that is described in detail in Section 5.3, we chose the tolerance value to be $\epsilon/2 = 0.1\lambda$, half of the value of Condition (29). In Figure 3.12e, the incremental keypoint r satisfies Condition (31) because the minimal path between r and $m(q)$ passes through the point q. The algorithm can therefore proceed till Condition (31) is not satisfied for a terminating point q. The complete curve detected using the modified algorithm is shown in Figure 3.12f. Figure 3.12f also shows all the ordered keypoints and the terminating point (point for which Condition (31) is not satisfied). The robust curve detection algorithm that was designed with the two modifications discussed in this section is described in *RobustCurveDetection*.

3.7 Summary

We have presented a novel algorithm for detecting curves with unknown endpoints based on minimal path techniques. We showed that this algorithm can detect curves using any arbitrary point on the desired curve as the sole user input. Synthetic image data was used to illustrate our algorithm. The algorithm can also be generalized to detect closed curves and detect curves with complex topologies that have both closed cycles and open sections. The algorithm is also easy and straight-forward to extend to volumetric data sets for the extraction of 3D (and even higher dimensional) curves.

Algorithm *RobustCurveDetection*
Input: Image Im, potential function Φ and initial source point set S.
Output: Detected Curve C

1. Start with initial source point set S containing an arbitrary point p on the curve.
2. Set $StopDetection = FALSE$.
3. Use $FMM(S, \lambda)$ to find keypoint k_1.
4. Run $MinimalPathbk(S, k_1)$ and initialize curve C to contain the minimal path between S and k_1.
5. Set $S \leftarrow S \cup k_1$.
6. Set $m(k_1) = p$ and $m(p) = k_1$.
7. Set $m_1(k_1) = 1$ (no. of neighbors is given by the map m_1.) and $m_1(p) = 1$.
8. Set $E \leftarrow p \cup k_1$.
9. **while** $StopDetection = FALSE$
10. **do** Run $FMM(S, \lambda)$ to find new keypoint k_2.
11. Store $STATUS$ of all points after $FMM(S, \lambda)$.
12. Run $MinimalPathbk(S, k_2)$ and find the origin point s^* in set S.
13. Set curve C' to contain the minimal path between S and k_2.
14. Compute point $s' = m(s^*)$.
15. Use $FMM(s', k_2)$ to calculate $L(s', k_2)$.
16. **if** $|L(s', k_2) - 2\lambda| < \epsilon$
17. **then** $S \leftarrow S \cup k_2$ and $m(k_2) = s^*$.
18. Set $m_1(k_2) = 1$ and $m_1(s^*) \leftarrow m_1(s^*) + 1$.
19. **if** $s^* \in B$ (branch-points set)
20. **then** $C \leftarrow C \cup n(s^*)$, $B = B - s^*$.
21. **if** $m_1(s^*) > 2$
22. **then** $n(k_2) = C'$
23. **else** $C \leftarrow C \cup C'$, $E \leftarrow E \cup k_2$, $E \leftarrow E - s^*$.
24. **else** $C'' = C'$, $q = k_2$ and $m(q) = s^*$.
25. Use $FMM(q, \lambda, \text{STATUS})$ to compute incremental keypoint r.
26. Set curve C' to contain the minimal path between q and r.
27. Use $FMM(m(q), r)$ to calculate $L(m(q), r)$.
28. **if** $|L(m(q), r) - 2\lambda| < \epsilon$
29. **then** $S \leftarrow S \cup q \cup r$, $C \leftarrow C \cup C'' \cup C'$ and $m(r) = q$.
30. Set $m_1(r) = 1$, $m_1(q) = 2$ and $m_1(m(q)) \leftarrow m_1(m(q))+1$.
31. $E \leftarrow E \cup r$, $E \leftarrow E - m(q)$.
32. **else** $StopDetection = TRUE$.
33. **for** all pairs of points e_i, e_j in E,
34. **if** $L(e_i, e_j) < 2\lambda$
35. Run $MinimalPathbk(e_i, e_j)$ and initialize C' to contain the minimal path between e_i and e_j.
36. **if** $|L(e_i, m(e_j)) - L(e_i, e_j) - \lambda| < \epsilon/2$
37. **then** $C \leftarrow C \cup C'$
38. **repeat**

CHAPTER IV

DEVELOPMENT OF QUANTIFICATION MEASURE FOR EVALUATING CRACK MAP ACCURACY

The Transportation Research Board Pavement Management Systems Committee [13] of United States identified a need for establishing fundamental pavement crack elements using automatic image processing, and that need can be addressed by a robust crack map generating or crack segmentation algorithm. As discussed in Chapter 2, a robust segmentation algorithm that accurately differentiates pavement crack pixels from the pavement pixels without crack can help to establish fundamental crack elements that can be standardized and compared among different transportation agencies. These segmented distress elements can be then classified into the distress types according to the definition of a particular transportation agency. Many crack segmentation algorithms have been developed in the past decade, but there is no good method to quantitatively evaluate the performance of these segmentation algorithms. A simple quantitative measure that is generally used by transportation agencies compares the number of crack pixels in the ground truth image to the number of crack pixels in the automatically segmented image. This is illustrated in Figure 4.1 where the number of crack pixels in a 5ft $\times$ ft area are compared. Even though qualitatively Algorithm 1 is much better than Algorithm 2 in detecting cracks, the quantitative measure reflects 100% detection for both algorithms. Unfortunately, qualitative assessment of image segmentation results by engineers is time consuming and inefficient; hence, there is a critical need to develop an objective quantitative evaluation criteria that accurately reflects the assessment of a trained visual inspector. This will help the research community focus on the development of better and faster algorithms.

Figure 4.1: Region based quantification measure.

We address this issue by surveying different quantification methods and developing a new, novel quantification method that accurately reflects the assessment of trained engineers. Most of this chapter is based on [37].

Nazef [46] conducted a qualitative evaluation of pavement distress detection algorithms under different lighting conditions, but the performance of segmentation algorithms was not analyzed quantitatively. Others, like Koutsopoulos [42] and Wang [59, 61] did a comparison of segmentation algorithms to show the superiority of their particular algorithms, but the evaluations are again qualitative. Huang and Xu [30], and Zhou [76] measured the performance of their algorithms by devising a scoring criterion based on statistical correlation. Mean square error is another metric, which is used extensively in image comparison studies. Another metric, called the Mahalanobis distance [24], has also been described in literature. All the above evaluation methods use the entire image data for image comparison and do not target the crack regions specifically. This can obscure the results, considering the fact that crack pixels are typically only a small percentage of the total image pixels, and these scoring

measures are not specifically sensitive to crack information. In addition, information about crack locations is not used in these evaluation methods. This may lead to the error that two segmented crack images having different crack locations (one is correct and the other is wrong) are considered the same simply because they have the same number of total crack pixels in an image. Different quantification methods are also used in medical imaging and machine vision. They are briefly discussed to evaluate the possibility of applying them to pavement images. These methods include Receiver Operator Characteristic (ROC) [38, 55, 68] and Hausdorff distance [5, 67]. Hausdorff distance enables the user to measure the distance between objects of different sizes. This can be helpful in the case of cracks, as the number of crack pixels in the ground truth image can be different from the crack pixels in the segmented images. A new quantification method based on the buffered Hausdorff distance metric is proposed. The buffered Hausdorff distance metric incorporates the merits of both mean square error and Hausdorff distance metric. The proposed method is compared with four other possible quantification methods (mean square error, statistical correlation, ROC, and Hausdorff distance) to demonstrate its superior capability in distinguishing the performance of different segmentation algorithms. This paper is organized as follows. The need for developing a method to quantitatively evaluate the performance of different segmentation algorithms is identified in this section. The proposed quantification method and other possible quantification methods that were used for comparison are presented in the subsequent section. Experimental tests that were performed to compare the capability of different quantification methods using both real and synthetic images are described next. Finally, conclusions and recommendations are made.

4.1 The Quantification Methodology

This section presents the proposed quantification method that is based on buffered Hausdorff distance. Four other possible quantification methods are also briefly introduced in this section. The scores of different quantification methods are normalized to a common scale of 0 to 100 to facilitate our evaluation. Zero and one hundred represent the worst and the best performance possible for an algorithm, respectively, according to this scoring criterion. The proposed quantification method is presented below.

4.1.1 The Proposed Buffered Hausdorff Distance Measure

Our proposed buffered Hausdorff distance measure tries to incorporate the strengths of both mean square error and Hausdorff distance by modifying the Hausdorff distance metric. The Hausdorff distance is among the most popular distance measures that measures the distance between two curves and is a metric. It has been extensively used in literature [5, 67]. For any two sets of points $A = a_1, a_2, \ldots, a_n$ and $B = b_1, b_2, \ldots, b_n$

$$H(A, B) = \max(h(A, B), h(B, A)), \tag{32}$$

where

$$h(A, B) = \max_{a \in A} \min_{b \in B} \|a - b\|$$

is the greatest of all the small distances from points of A to B and $h(B, A)$ is the greatest of all the small distances from points of B to A. Figure 4.2 illustrates this distance measure effectively.

The value of Hausdorff distance is large, even if one crack pixel in the automatically segmented image is far from ground truth image crack pixels. Seeing this limitation of the Hausdorff distance metric, a new metric was developed that does not suffer from the defects of the Hausdorff distance. The intuitive development of this measure is described next. A better distance measure than the Hausdorff distance is the modified

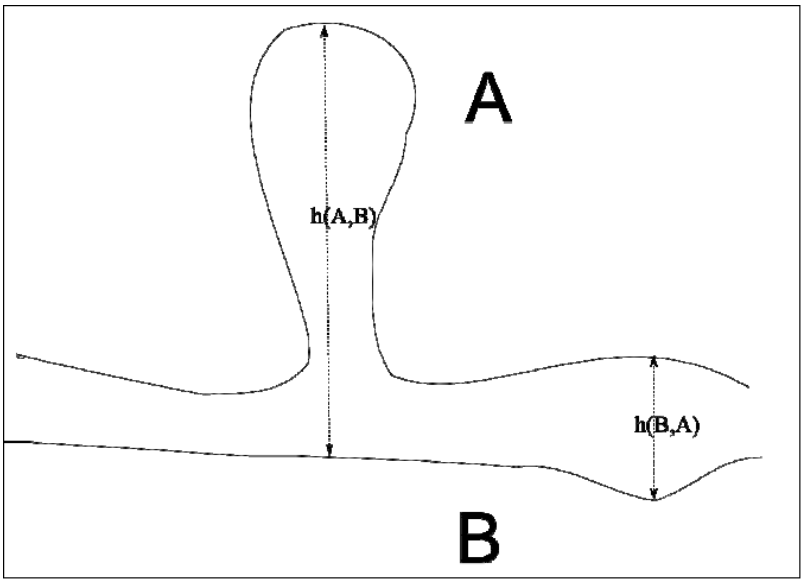


Figure 4.2: Hausdorff distance illustration.

Hausdorff distance given by $MH(A, B)$:

$$MH(A, B) = \max(h_1(A, B), h_1(B, A)), \tag{33}$$

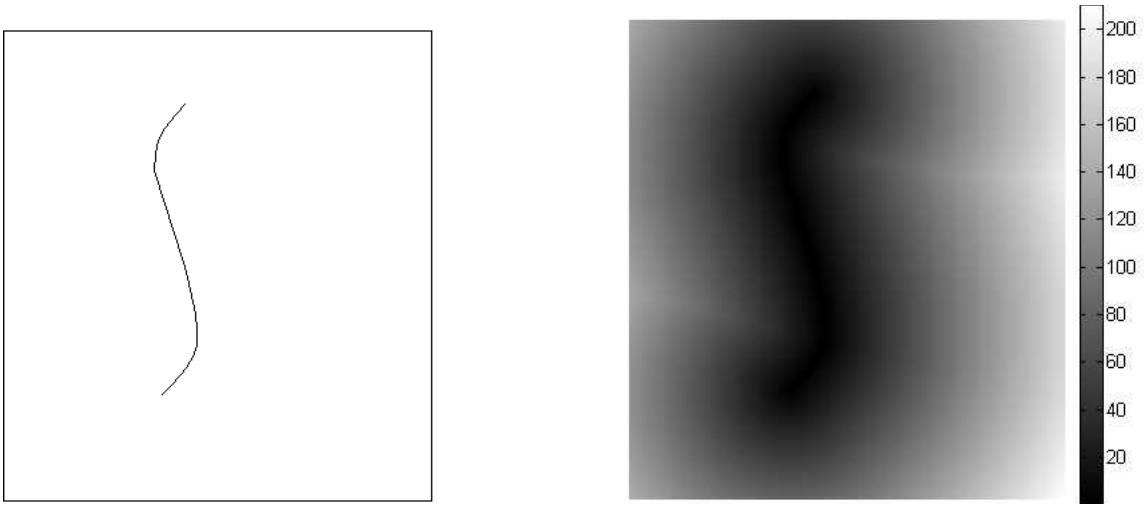
where

$$h_1(A, B) = \frac{1}{m} \sum_{a \in A} \min_{b \in B} \|a - b\|$$

In order to apply this measure to our set of images, we employ the procedure that follows. Let M be the manually segmented ground truth image and P be the segmented image after automatic detection. Let $M1$ be the matrix of co-ordinate locations derived from M where the crack is supposedly located, and let $P1$ be the matrix of co-ordinate locations derived from P. $M1$ and $P1$ locations correspond to sets A and B respectively. Then we have

$$M1 = \begin{pmatrix} m_{11} & m_{12} \\ \vdots & \vdots \\ m_{N1} & m_{N2} \end{pmatrix}_{N \times 2}, P1 = \begin{pmatrix} p_{11} & p_{12} \\ \vdots & \vdots \\ p_{N1} & p_{N2} \end{pmatrix}_{S \times 2}$$

One can calculate the modified Hausdorff distance $MH(M1, P1)$ which is given by Equation (33). After using the modified Hausdorff distance measure initially for our



(a) (b)

Figure 4.3: (a) Image with Ground truth crack to compute $M1$. (b) Modified distance variation for pixel points in the Image

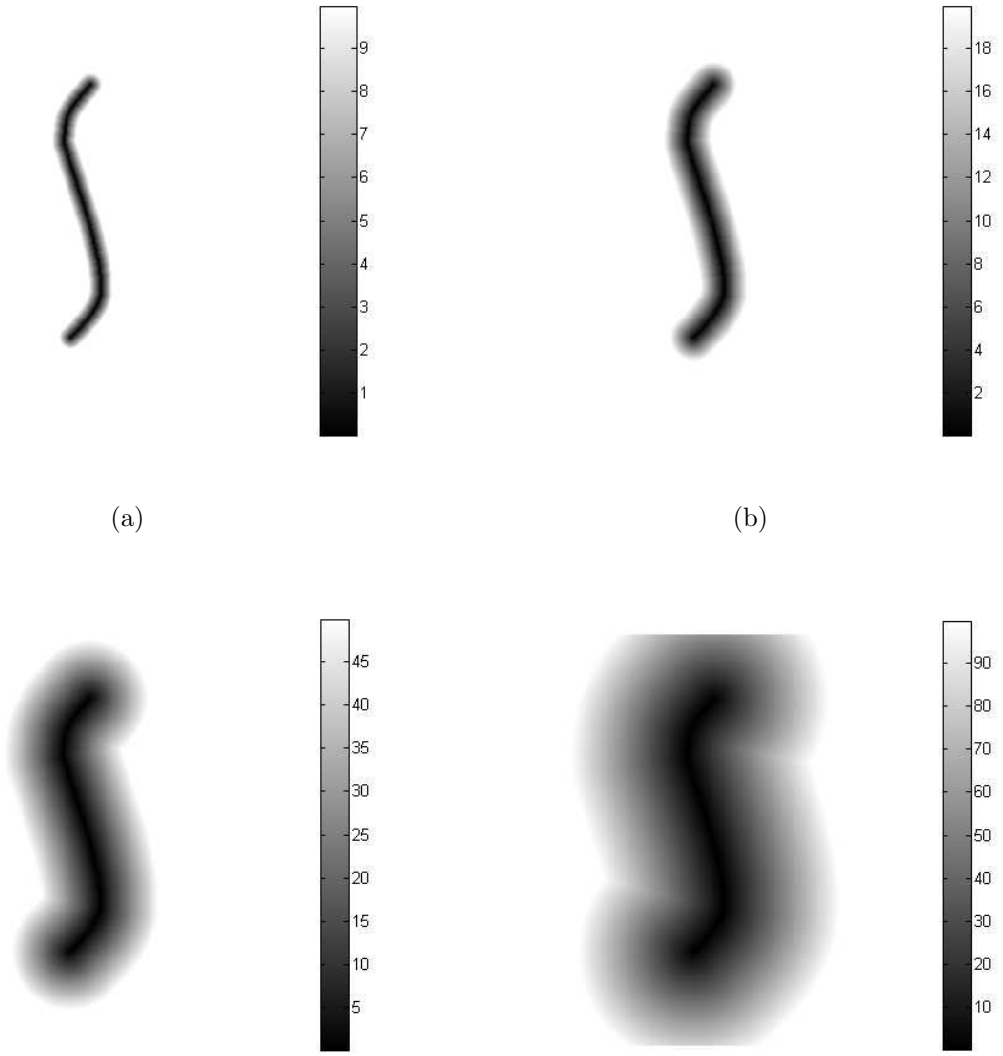
image comparison, we felt that there was one more possible improvement. If we have $M1$ locations determined by the curve(or the crack) in Figure 4.3a , the distance h_1 for each pixel location in the image increases uniformly as the pixel moves away from the ground truth crack locations. The cracks are represented by a curve in Figure 4.3b. However, when a crack pixel in the automatically segmented image falls substantially away from the closest pixel in the ground truth image, it no longer makes sense to heavily penalize this distance. Wrong detections beyond a certain distance should be penalized equally. This led to a new distance measure, the buffered Hausdorff distance measure given by $BD(A, B)$ where A and B are co-ordinate locations of the curve in the ground truth image and automatically segmented image respectively.

$$BD(A, B) = \max(h_2(A, B), h_2(B, A)), \tag{34}$$

where

$$h_2(A, B) = \frac{1}{m} \sum_{a \in A} sat_L \min_{b \in B} ||a - b||. \tag{35}$$

Here sat_L indicates that when the distance of the crack pixel to the closest crack pixel in the other image exceeds a saturation value L, we use a constant value of



(a) (b)

(c) (d)

Figure 4.4: (a)Buffered Hausdorff distance variation for $L = 10$. (b) Buffered Hausdorff distance variation for $L = 20$. (c)Buffered Hausdorff distance variation for $L = 50$. (d)Buffered Hausdorff distance variation for $L = 10$.

L for the distance. The buffered Hausdorff distance algorithm is given in *Buffered Hausdorff distance*. It is applied to the same matrices $M1$ and $P1$, which were derived before. Figure 4.4 shows the variations of buffered Hausdorff distance h_2 computed at each pixel location for different L values . As before, ground crack locations in Figure 4.4 are used to construct the $M1$ matrix. The value of L can be decreased if the user desires more accuracy in detection. For instance, if the L value is changed from 20 to 10, all the automatically segmented crack pixels beyond a distance of 10 from the ground truth will be given the worst performance measure. When the L value is 20, only automatically segmented pixels beyond a distance of 20 will be given the worst performance measure. The buffer L was heuristically chosen to be 50 for our comparison experiments. Figure 4.5 illustrates the buffered Hausdorff distance measure for two curves that represent crack locations of the ground truth and the automatically segmented image. The sample values of the buffered Hausdorff distance have a very intuitive meaning. The buffered Hausdorff distance can be interpreted as the average Euclidean distance between the crack pixels in the ground truth image and the segmented images. The measure can also be used if 3D crack location information is available by computing the corresponding Euclidean distance. In addition, this quantification method can be used to validate algorithms for detection of features like optical nerves, vessels etc that are similar to cracks.

To compare other scoring methods with this buffered Hausdorff distance, a scaled scoring measure was derived as given below:

$$\text{Buffered distance score} = 100 - \frac{BD(A,B)}{L} \times 100. \tag{36}$$

The buffered Hausdorff distance effectively measures the performance of the segmentation methods and generates a score that corresponds with the qualitative performance of the particular method. In order to establish the merits of the buffered scoring distance, four other scoring measures were used in our experiments. Among the scoring measures, the buffered Hausdorff distance gives the best performance, as

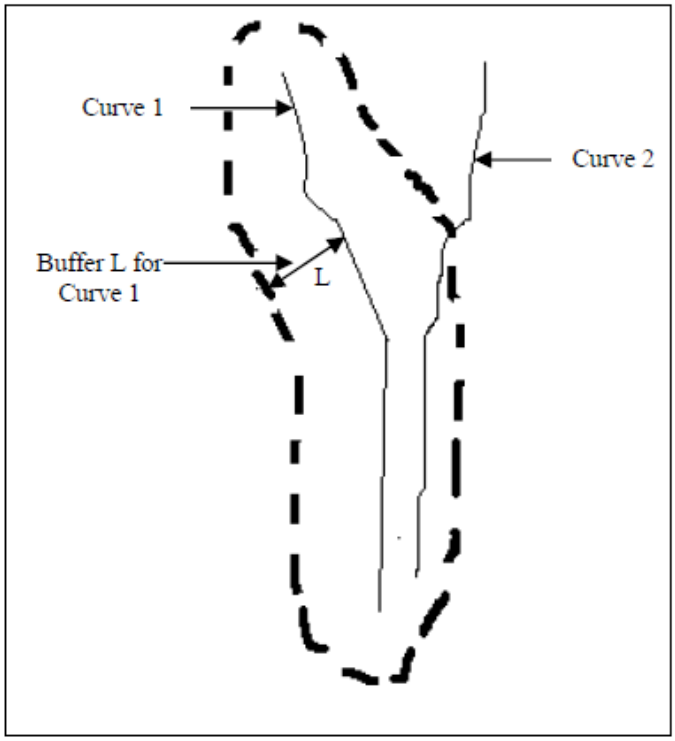


Figure 4.5: Buffered Hausdorff distance illustration.

Algorithm *Buffered Hausdorff distance*
Input: Crack location matrix $M1$, crack location matrix $P1$ and buffer value L.
Output: buffered Hausdorff distance $BD(M1, P1)$.

1. **for** $i \leftarrow 1$ **to** S
2. **do** Min Dist = $\| \min_{j=1,\ldots,S}(P1(\text{i}^{\text{th}}\text{row}) - M1(\text{j}^{\text{th}}\text{row}))\|$.
3. **if** Min Dist $> L$
4. **then** Min Dist $= L$.
5. Set Distance1 = Distance1 + Min Dist.
6. $h_2(P1, M1) = \frac{\text{Distance1}}{S}$.
7. **for** $i \leftarrow 1$ **to** N
8. **do** Min Dist = $\| \min_{j=1,\ldots,S}(M1(\text{i}^{\text{th}}\text{row}) - P1(\text{j}^{\text{th}}\text{row}))\|$.
9. **if** Min Dist $> L$
10. **then** Min Dist $= L$.
11. Set Distance2 = Distance2 + Min Dist.
12. $h_2(M1, P1) = \frac{\text{Distance2}}{N}$.
13. $BD(M1, P1) = \max(h_2(P1, M1), h_2(M1, P1)$.

will be clear from the results.

To compare the performance of the proposed scoring method, four other scoring methods were used: mean square error, statistical correlation, receiver operator characteristic, and Hausdorff distance. These scoring methods are described briefly below.

4.1.2 Mean Square Error Method

Mean Square Error (MSE) is one of the most commonly used performance metrics in the image processing literature, especially in image compression. MSE is the cumulative squared error between the two images, indicated by I_1 and I_2 respectively. In our case the two images used are the ground truth image and the automatically segmented image.

$$\text{MSE}(I_1, I_2) = \frac{1}{MN} \sum_{y=1}^{M} \sum_{x=1}^{N} (I_1(x, y) - I_2(x, y))^2 \tag{37}$$

Here, M and N are row and column lengths of the images respectively and (x, y) indicate the co-ordinates of the pixel location in each image. In order to compare MSE performance with other scoring measures, a scaled scoring measure is devised that gives values between 0 and 100. As the ground truth image and the automatically segmented images are binary images, the error for each pixel is either 1 or 0. Considering that cracks comprise a small portion of an image, we analyzed 150 different pavement images to compute the maximum crack pixels as a percentage of total pixels. Observations showed that this percentage never exceeded 5%; hence, more than 5% error between two images was considered the worst segmentation performance. Using this fact, a reasonable scaled scoring measure based on the MSE was derived for comparison purposes. It is given by the following equation:

$$\text{MSE score} = 100 - \frac{\text{MSE}(I_1, I_2)}{0.05} \times 100 \tag{38}$$

.

4.1.3 Statistical Correlation Method

Statistical correlation is another measure used to evaluate performance in the existing literature. It is given by the correlation coefficient $\text{Corr}(I_1, I_2)$ between two images,

$$\text{Corr}(I_1, I_2) = \frac{Cov(I_1, I_2)}{\sqrt{Var(I_1)}\sqrt{Var(I_2)}} \tag{39}$$

where $Cov(I_1, I_2)$ is the covariance between the two images, and $Var(I_1)$ and $Var(I_2)$ are the variances of the two images. As the correlation lies between -1 and 1, the scoring measure for correlation is derived using this information.

$$\text{Corr Score} = \frac{\text{Corr}(I_1, I_2) + 1}{2} \times 100 \tag{40}$$

4.1.4 Receiver Operator Characteristic Method

The Receiver Operator Characteristic (ROC) is a highly effective tool for image classification evaluation, and it has been extensively used in medical literature and machine learning [38, 55, 68]. In [3], it was used to evaluate the performance of the segmentation algorithm. The ROC represents the dependence of the rate of correct detections and the rate of false detections. The correct detection (CD) rate is defined as the ratio between the number of areas or pixels correctly labeled as defective $\text{N}_{\text{defective}}$ to the number of truly defective areas N_{true} on the image:

$$\text{CD} = \frac{\text{N}_{\text{defective}}}{\text{N}_{\text{true}}} \tag{41}$$

. The false alarm (FA) rate is defined as the ratio between the number of false detections N_{false} to the number of truly defect-free areas ($\text{N}_{\text{all}} - \text{N}_{\text{true}}$) (where N_{all} is total number of areas on the image):

$$\text{FA} = \frac{\text{N}_{\text{false}}}{\text{N}_{\text{all}} - \text{N}_{\text{true}}} \tag{42}$$

The ROC value is defined based on the ratio between FA and CD:

$$\text{ROC} = \frac{\text{FA}}{\text{CD}} \tag{43}$$

. For our experiments, correctly detected pixels in the segmented image needed to lie within a square of 5-pixel width centered on a true crack pixel in the ground truth image. A scaled scoring measure based on ROC was derived for comparison purposes by taking the maximum value of the ROC to be 0.1. A ROC value above 0.1 would indicate an extremely poorly segmented image, which is not useful for further investigation.

$$\text{ROC score} = 100 - \frac{\text{ROC}}{0.1} \times 100 \tag{44}$$

4.1.5 Hausdorff Distance Method

The Hausdorff distance $H(A, B)$ between two sets A and B was described in Section 4.1.1 and is represented by Equation (32) . Using the width of the ground truth image as a scaling factor, a scoring measure for the Hausdorff distance was calculated.

$$\text{Hausdorff distance score} = 100 - \frac{H(A, B)}{\text{Column Width}} \times 100 \tag{45}$$

4.2 Experimental Results and Discussion

For comparing the capability of the proposed quantification method to other methods, both real data and synthetic data, simulating different pavement distress conditions, were used. For the actual pavement distress images, GDOT pavement engineers visually marked the cracks (their actual locations) to establish the ground truth. Synthetic images were generated in order to demonstrate the comparative strength of the buffered Hausdorff distance method over other quantitative scoring methods. Figure 4.6 gives an overview of the experimental data presented in this section. Four segmentation algorithms were used to evaluate the capability of different quantification methods for being able to separate the performance of different segmentation methods. These segmentation algorithms include Canny edge detection [8], crack seed verification [30], the iterated clipping method [47], and the dynamic optimization-based method [3]. Although a large data set of 100 images has been analyzed in

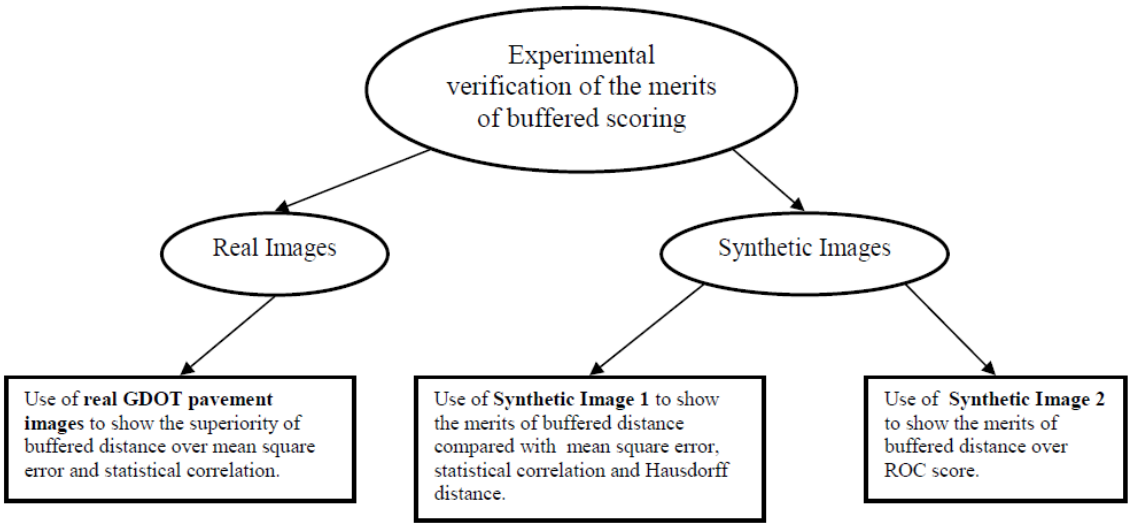


Figure 4.6: Overview of experimental data.

our experiments, here we present the results of the four segmentation methods on 4 GDOT images. The results obtained from these segmentation algorithms were visually inspected by trained GDOT pavement engineers to assess the comparative performance of the four algorithms. This known performance for each segmentation algorithm was then used to evaluate the capability of different quantification methods. For example, the engineers and authors observed that dynamic optimization-based method gives substantially better performance than other methods qualitatively for all the test images [57]. The underlined mean square error score and correlation score values in Table 1 indicate that the dynamic optimization-based method is not the best method for some images. Hence, the results of these two scoring measures do not match with the assessment of GDOT engineers. It is also found that the buffered Hausdorff distance score, ROC score and Hausdorff distance scores are consistent with the visual assessment, but the buffered Hausdorff distance score achieves the best score separation to distinguish the performance of different algorithms.

Two synthetic image data sets were generated for illustrating the better performance of buffered Hausdorff distance compared to Hausdorff distance and ROC, and to demonstrate a better score separation achieved by our proposed method. The

Table 1: Scoring measures for GDOT images

Scoring Measure	Dynamic optimization	Canny edge detection	Seed verification	Iterated clipping
Image 1				
Mean Square Error	95.9941	95.1833	95.4276	96.6716
Correlation	60.3986	49.8636	55.8749	58.7725
ROC measure	99.6927	0	99.614	95.9945
Hausdorff distance	97.6623	80.9665	39.3898	50.9328
Buffered Hausdorff distance	91.6468	24.4089	72.4155	38.8489
Image 2				
Mean Square Error	95.6805	95.0156	95.0714	95.9808
Correlation	58.551	49.846	54.24	59.6205
ROC measure	99.5189	0	99.716	98.4453
Hausdorff distance	96.9521	41.9934	55.3874	42.2313
Buffered Hausdorff distance	91.8955	14.0205	17.5782	66.1937
Image 3				
Mean Square Error	95.5108	94.7141	94.6804	95.5614
Correlation	59.6935	49.9575	50.8001	58.3103
ROC measure	99.6954	2.2489	99.7577	98.2338
Hausdorff distance	98.0013	41.3101	0	40.6776
Buffered Hausdorff distance	92.4423	14.9668	3.1376	64.0048
Image 4				
Mean Square Error	95.4859	94.4584	94.5122	94.5391
Correlation	61.5723	49.9428	53.6835	52.1224
ROC measure	99.7451	0	99.7203	91.3818
Hausdorff distance	97.432	49.3804	31.0443	50.6515
Buffered Hausdorff distance	93.0536	16.1591	19.1754	24.4211

first synthetic ground truth image, which is shown in Figure 4.7b, was generated by marking isolated noise pixels apart from true crack pixels, which are located far from the crack pixels, on top of a GDOT raw image as shown in Figure 4.7a. This synthetic ground truth image was compared to four test images generated by applying the four automatic segmentation algorithms to the raw pavement image. Images in Figure 4.7 clearly show that though Test Image1 has the best performance, but table 2 shows that the Hausdorff distance score and the mean square error actually list it as the worst-performing image. The problem is that Hausdorff distance is very sensitive to isolated noise outliers and does not reflect the overall performance of the segmentation method. The buffered Hausdorff distance measure accurately reflects the performance of test images and also achieves good score separation to distinguish their performance behavior.

The second synthetic ground truth image is illustrated in Figure 4.8. This synthetic image has a curve that runs through the middle of the image. The Test images 1, 2, and 3 are horizontal translations of the ground truth image by 1, 2 and 3 pixels,

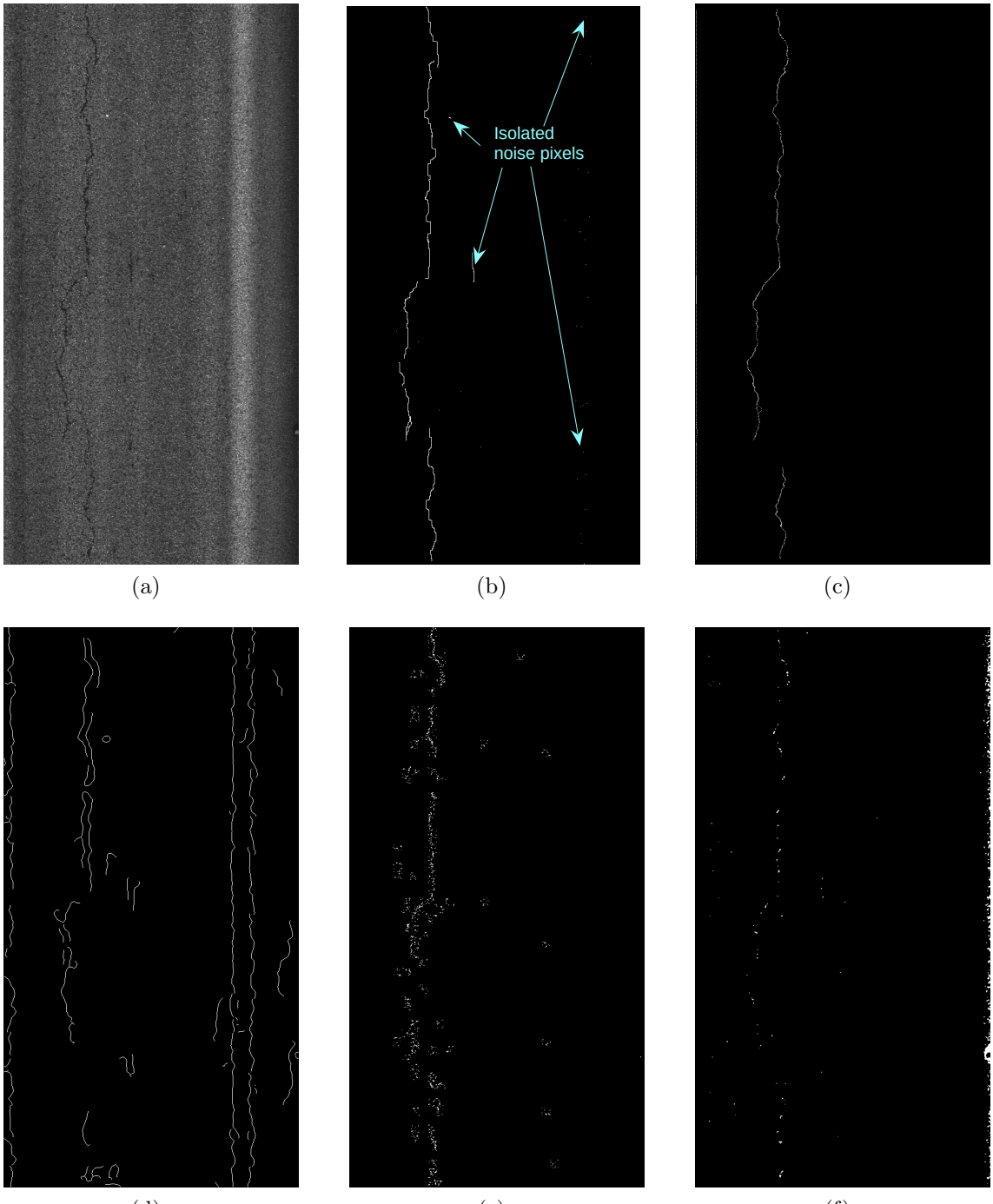


Figure 4.7: (a) Raw image. (b) Synthetic ground truth image with added noise pixels. (c) Test image 1. (d) Test image 2. (e) Test image 3. (f) Test image 4.

Table 2: Scoring measures for synthetic Image 1

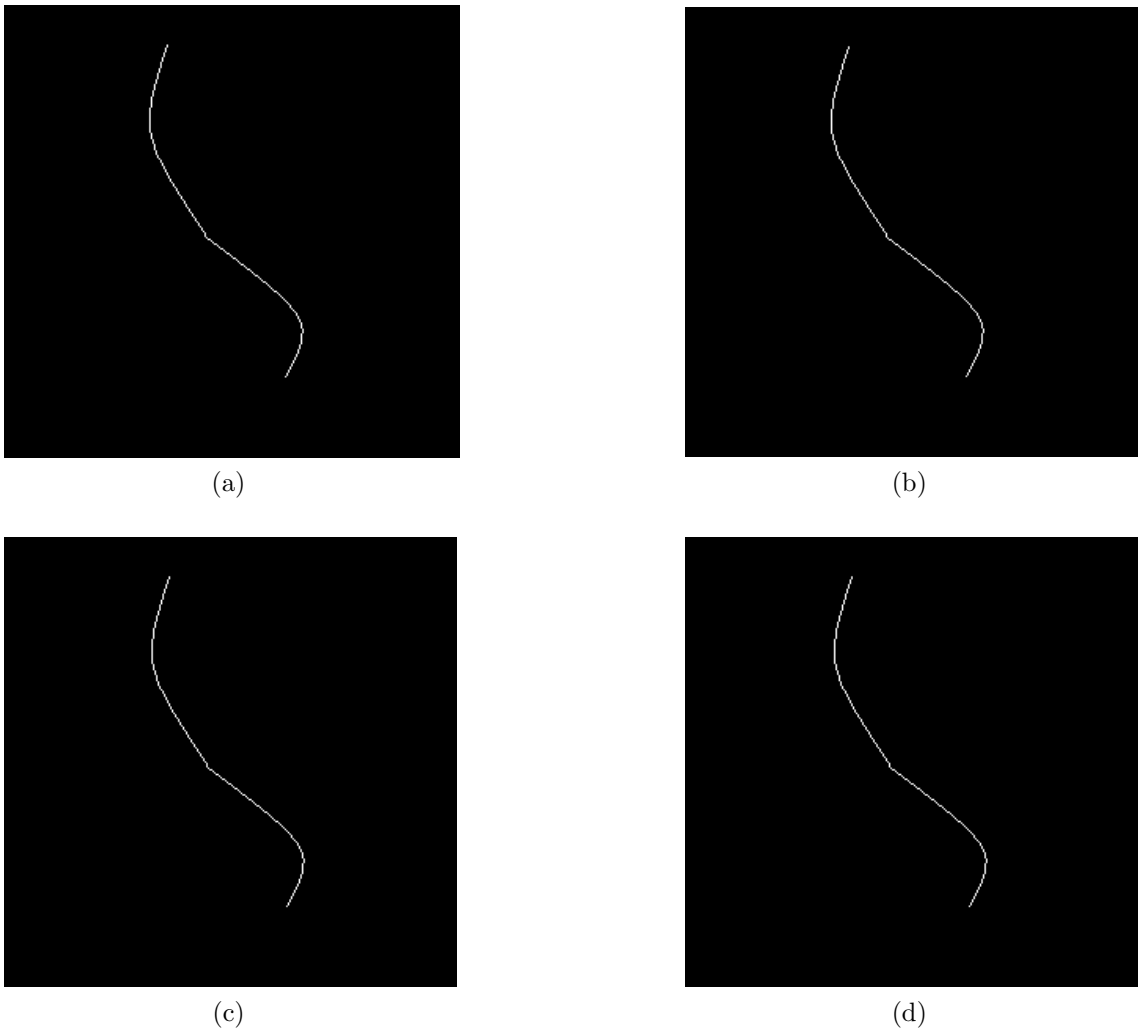
Scoring Measure	**Test Image 1**	**Test Image 2**	**Test Image 3**	**Test Image 4**
Mean Square Error	95.2962	94.6596	95.4198	94.8768
Correlation	58.7137	50.0709	56.244	54.9956
ROC measure	99.5161	36.5221	95.5748	98.683
Hausdorff distance	44.7479	64.2952	69.6907	66.9031
Buffered Hausdorff distance	80.9604	22.6031	57.0306	64.022

Table 3: Scoring measure for synthetic Image 2

Scoring Measure	**Test Image 1**	**Test Image 2**	**Test Image 3**
ROC measure	100	100	0
Buffered Hausdorff distance	98	96	94

respectively. Table 3 shows the scoring measure results for the 3 images. It is seen that the ROC measure allots the same score of 100, reflecting the best performance, to both Test Image 1 and Test Image 2 and a score of 0, reflecting the worst performance, to Test Image 3. This observation illustrates that ROC can show very different score value for two similar images. When the horizontal translation exceeds 2 pixels, then all crack pixels in the test image are classified as wrongly detected. This problem happens around the boundary points where the classification decision changes from true detection to false detection. The buffered Hausdorff distance score still accurately reflects the performance of the test images because it is not based on a hard decision rule like ROC. The combined results indicate that the buffered Hausdorff distance score is better than the other four scoring methods used in experiments.

Based on the results, we examined the strengths and weaknesses of each scoring method. Mean Square Error (MSE) takes into account the error in the whole image and not just the error in location of crack pixels. Moreover, in its computation, MSE does not take into account the relative proximity of the crack pixels in the ground truth image to the crack pixels in the segmented image. Unless there is an exact overlap between the crack pixels in the ground truth image and the segmented image, the MSE score will be the same for the two different automatically segmented



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

Figure 4.8: (a) Synthetic ground truth Image. (b)Test image 1 (translation = 1 pixel). (c) Test image 2 (translation = 2 pixels. (d) Test image 3 (translation = 3 pixels.

images. In other words, the MSE score stays the same irrespective of the relative proximity of the crack pixels in the automatically segmented image to the cracks in the ground truth image. This exact overlap is highly unlikely, considering the fact that marking the ground truth image to such a degree of accuracy can be a tedious exercise. MSE can only be used on images of similar sizes and cannot selectively focus on the crack pixels in the image. Thus, it is not very useful for comparison of pavement segmentation algorithms. Like MSE, the correlation coefficient also requires exact overlap between crack pixels of the two compared images to get a high performance evaluation score. This coefficient also suffers from the limitation that it can be applied only to similarly sized objects. Hence, the non-crack pixels, which are larger in extent in all pavements, affect the correlation severely. Results of ROC are better compared to MSE and correlation scoring measure, but it is seen through results of the second synthetic image data set that qualitatively very similar segmented images can have very different ROC values. This happens due to the fact that ROC uses hard decision rules to classify pixels as false alarms or true defects. Hence, on the boundary of these decision rules, the ROC value can change abruptly, and very similar segmented images can have drastically different ROC values. The Hausdorff distance has the ability to measure the distance between only crack pixels in both the ground truth and segmented image, and, therefore, it can capture the local effectiveness of the segmentation algorithm. However, it is very sensitive to outliers or noise pixels, and Hausdorff score values change drastically even in the presence of one outlier. The buffered Hausdorff distance measure captures the local effectiveness of the segmentation algorithm and is not sensitive to outliers or noise pixels. It also achieves good score separation in the values for different segmentation algorithms, and it is the best out of all the methods. In this method, the buffer value L needs to be chosen by the user according to the accuracy requirements of the system. The drawback of the method is that the selection of this value L is still heuristic in nature.

4.3 Summary

Researchers have developed many pavement crack segmentation algorithms in the past, but it is difficult to compare the performance of different algorithms efficiently without an accurate quantitative method. This hinders the focused development of better segmentation algorithms. Our research is motivated by this need to develop a method to quantitatively evaluate the performance of different pavement distress segmentation algorithms. In this chapter, we introduced a novel quantification method based on the buffered Hausdorff distance. In addition, the capability of the proposed method was compared with four other possible quantification methods (mean square error, statistical correlation, receiver operator characteristic (ROC) and Hausdorff distance). Both real and synthetic data were used to validate the capability of our proposed quantification method. The real data sets consisted of raw GDOT images and the resultant images of four segmentation algorithms. These data sets were visually inspected to assess the performance of different algorithms. It is found that the mean square error and statistical correlation do not reflect the assessed performance of different segmentation algorithms. Further, two sets of synthetic images were generated to show the better performance of the buffered Hausdorff distance method compared to the Hausdorff distance and the ROC method and to illustrate the good score separation achieved by the buffered Hausdorff distance method.The experimental results indicate that the buffered Hausdorff distance scoring measure accurately reflects the observed performance of the segmentation techniques and outperforms the other four quantification methods. It also achieves good score separation to distinguish between the performance of different methods.

CHAPTER V

EXPERIMENTAL RESULTS AND ANALYSIS

In order to analyze the novel self-terminating minimal path algorithm explained in Chapter 3 (*RobustCurveDetection*), the algorithm was tested extensively on both real and synthetic experimental data sets. This chapter presents a detailed analysis of these experiments. We used the scaled buffered distance (SBD), true positive rates, true negative rates and false negative rates to evaluate the accuracy of our algorithm. The SBD is a scaled version of the buffered Hausdorff distance measure described in Chapter 4. Recall that the buffered Hausdorff distance is given by given by $BD(A, B)$ where A and B are coordinate locations of the curve in the ground truth image and the processed image respectively.

$$BD(A, B) = \max(h_2(A, B), h_2(B, A)), \tag{46}$$

where

$$h_2(A, B) = \frac{1}{m} \sum_{a \in A} sat_L \min_{b \in B} ||a - b||. \tag{47}$$

$BD(A, B)$ lies between 0 and L and we normalize this value to lie between 0 and 1 to get scaled buffered distance $SBD(A, B)$. The value of SBD approaches 0 if the curve detected by the algorithm and the curve in the ground truth image are close to each other.

$$SBD(A, B) = \frac{BD(A, B)}{L}. \tag{48}$$

The value of L was chosen to be 50 for our experiments.

The true positive (TP) rate is defined as the ratio between the number of curve pixels in the ground truth image correctly detected in the processed image ($N_{correct}$)

to the total curve pixels in the ground truth image (N_{ground}):

$$TP = \frac{N_{correct}}{N_{ground}}. \tag{49}$$

For our experiments, correctly detected pixels ($N_{correct}$) are pixels in the ground truth image that lie within a square of 5-pixel width centered on a detected curve pixel in the processed image. The false positive (FP) rate is defined as the ratio between the number of pixels in the processed image that are wrongly identified as curve pixels (N_{false}) to the total curve pixels in the ground truth image:

$$FP = \frac{N_{false}}{N_{ground}}. \tag{50}$$

The false negative (FN) rate is the ratio between the the number of true curve pixels that are undetected ($N_{ground} - N_{correct}$) to the total curve pixels in the ground truth image:

$$FN = \frac{N_{ground} - N_{correct}}{N_{ground}}. \tag{51}$$

In this chapter, we first present the experimental results of our algorithm on crack images in pavements and concrete structures. These cover the three applications of semi-automatic crack detection, crack sealing in pavements and tracking crack growth in concrete structures. The consolidated quantitative validation of several crack images is also presented. Next, the algorithm results on a two medical images with thin elongated features are shown. We then present the results of a detailed sensitivity study on the effect of algorithm parameters λ and ϵ on curve detection. Synthetic images with varying background intensities were used to conduct the sensitivity analysis. Finally, the extension of the robust algorithm is described where user input can be utilized at successive stages of the algorithm for more accurate curve detection results.

5.1 Results on Crack Images

We present the experimental results of the algorithm for three applications: semi-automatic crack detection, detection of continuous cracks for crack sealing in pavements and tracking crack growth in critical structures.

5.1.1 Semi-automatic Crack Detection

The objective of the experimental tests was to detect cracks of different types using an arbitrary user defined point on the crack. We used 80 images provided by Georgia Department of Transportation (GDOT) to conduct our study. Figure 5.1 presents the results on 4 pavement images that have different crack types: longitudinal, transverse, diagonal and compound. Figure 5.2 shows the algorithm results on a complex pavements crack with branches and closed cycles. The *RobustCurveDetection* algorithm was used on all the crack images. Some of the processed images have branch-points that were discussed as part of the *RobustCurveDetection* algorithm. The robust algorithm avoids false positives associated with these branch-points. Figure 5.1f, 5.1g and 5.2b contain branch-points that have no crack detection associated with them. The qualitative analysis of the algorithm on 80 GDOT pavement crack images gave very good results for crack detection. A λ value of 50 was used for all the pavement images. A more rigorous, quantitative validation using FN, FP, TP and SBD measures was conducted on 25 selected images. The consolidated quantitative validation for crack images, including those used for detecting continuous cracks and tracking crack growth is given in Section 5.1.4.

5.1.2 Crack Sealing Application

Many efforts have been made to automate the process of crack sealing. According to the recent study by Kim et al. [39], complete automation of crack mapping and sealing operations is not feasible and a desirable balance has to be found in the human and machine functions. Specifically, it was concluded that complete automation of crack

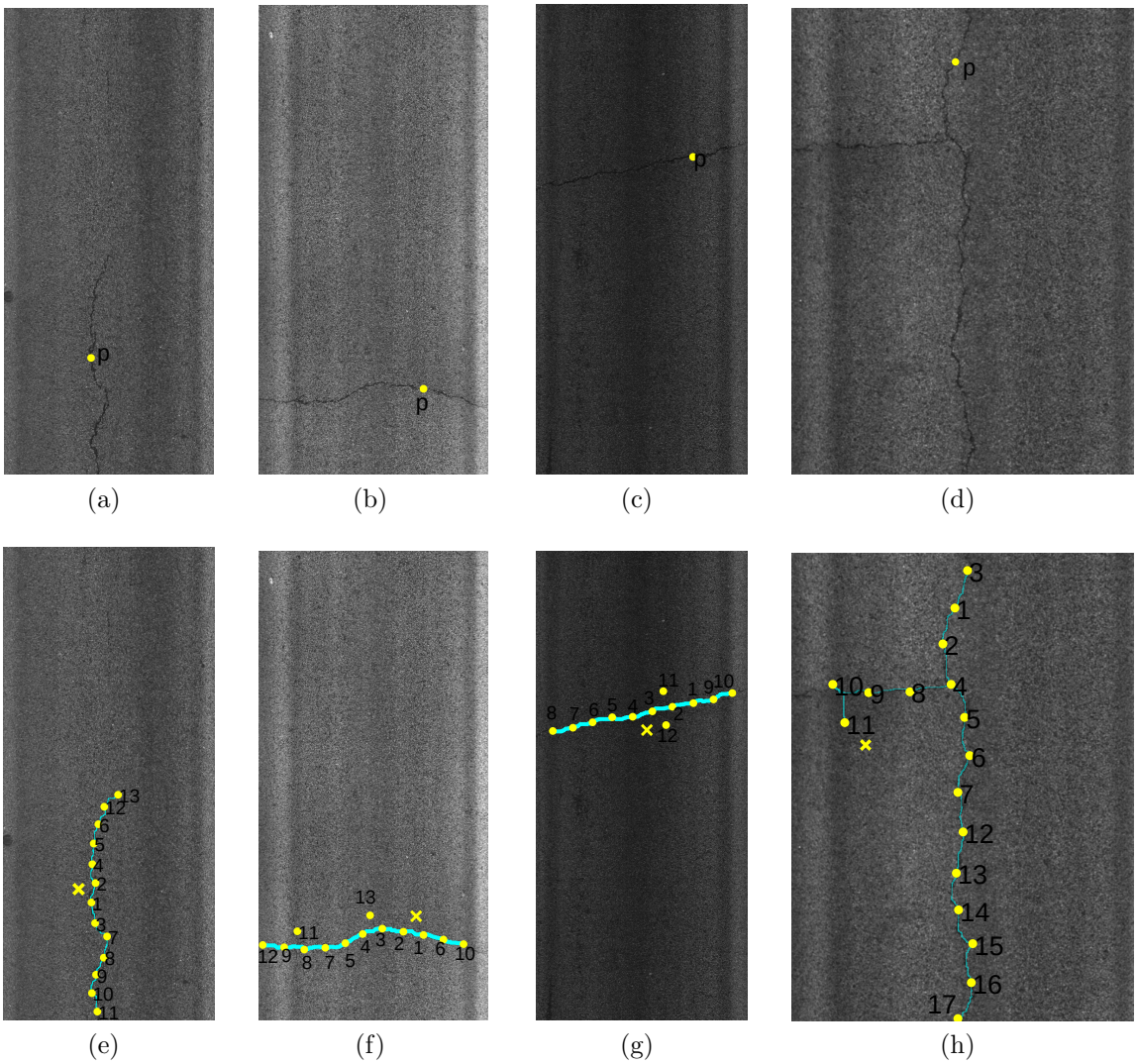
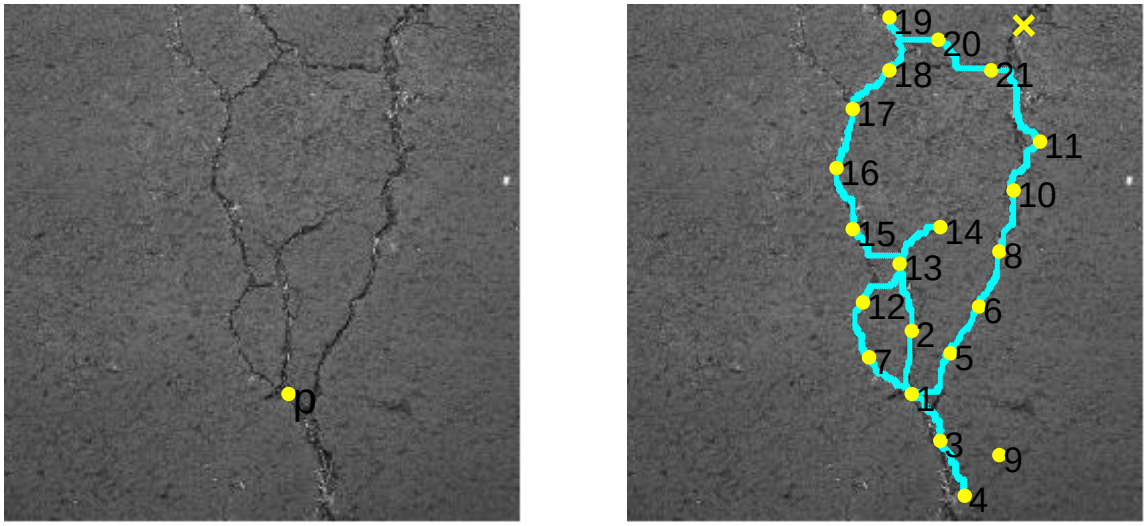


Figure 5.1: (a) Longitudinal crack. (b) Transverse crack. (c) Diagonal crack. (d) Complex crack with multiple branches. (e) Longitudinal crack detected with ordered keypoints and terminating point (marked by 'x'). (f) Transverse crack detected with ordered keypoints (including branch-points) and terminating point (marked by 'x'). (g) Diagonal crack detected with ordered keypoints (including branch-points) and terminating point (marked by 'x'). (h) Compound crack detected with ordered keypoints and terminating point (marked by 'x').

mapping is not desirable because pavement images have varying lighting conditions, poor contrast, oil stains and shadows that make accurate automatic crack detection very difficult. In addition, crack mapping for crack sealing has different requirements

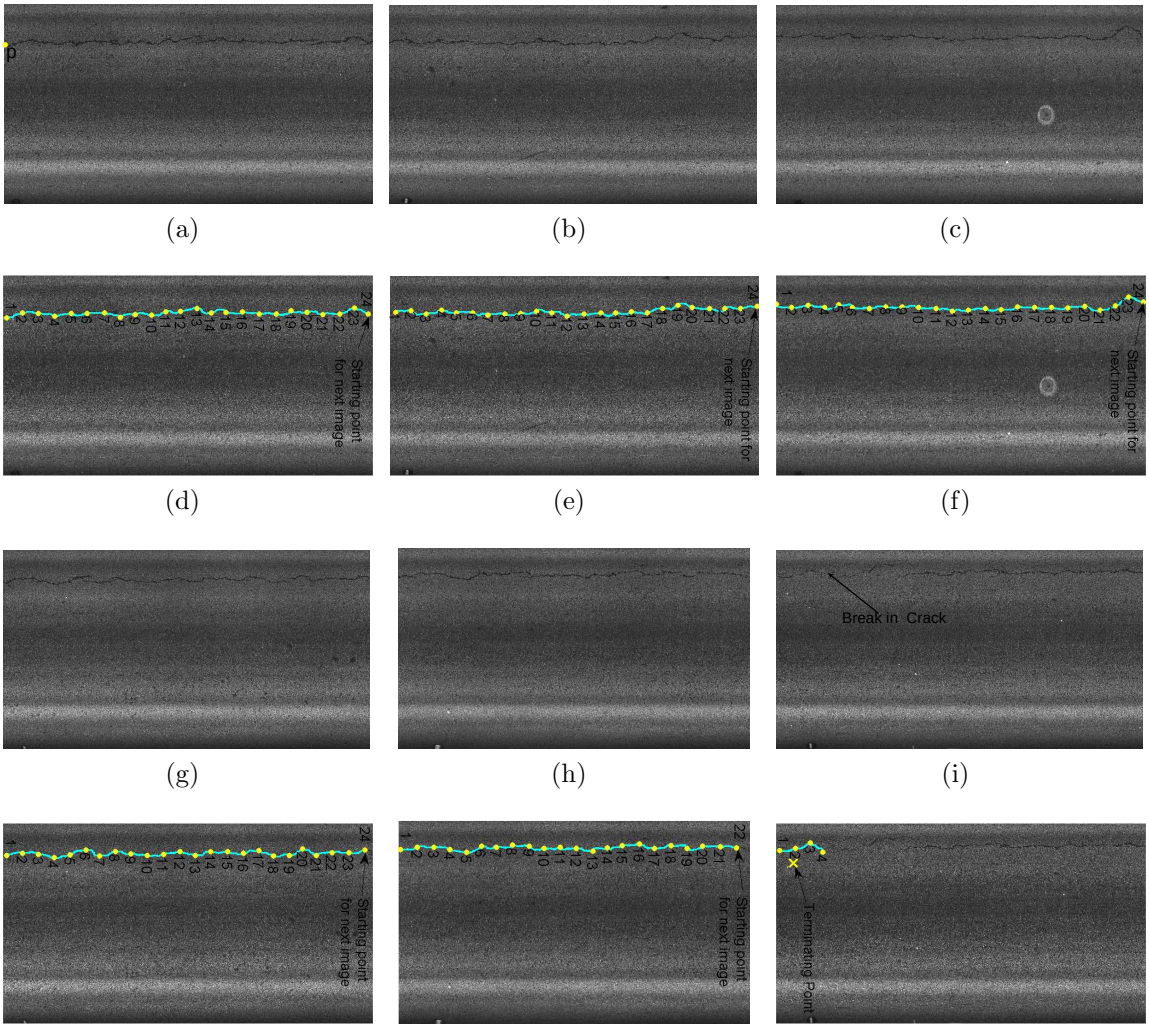


(a) (b)

Figure 5.2: (a) Complex crack with multiple branches and closed cycles. (b) Complex crack containing branches and closed cycles detected with ordered keypoints (including branch-points) and terminating point (marked by 'x').

from the pavement distress survey. Firstly, there is a higher need for location and direction accuracy in the crack map detection for the automatic crack sealer. Secondly, the crack map that is used as input for robotic arm path planning needs to have continuous components without too many discontinuities and isolated pixels. This is due to the fact that the automatic crack sealer needs to be efficient and seal large continuous crack segments without too many breaks. However, most machine vision algorithms do not ensure that they detect continuous crack segments. Therefore, we use our algorithm to detect cracks with minimum user interaction. The algorithm can detect continuous longitudinal cracks that extend over several miles with the specification of just the starting point of the crack by the user.

To demonstrate this concept, we selected 3 sets of pavement images with continuous longitudinal cracks. Each of these sets contained continuous cracks extending over 6-10 images. Figure 5.3 shows one of these data sets. We specified an initial crack point p in the first image shown in Figure 5.3a. Using our *RobustCurveDetection* algorithm, we kept on detecting keypoints until a new keypoint was less than a



(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

(j) (k) (l)

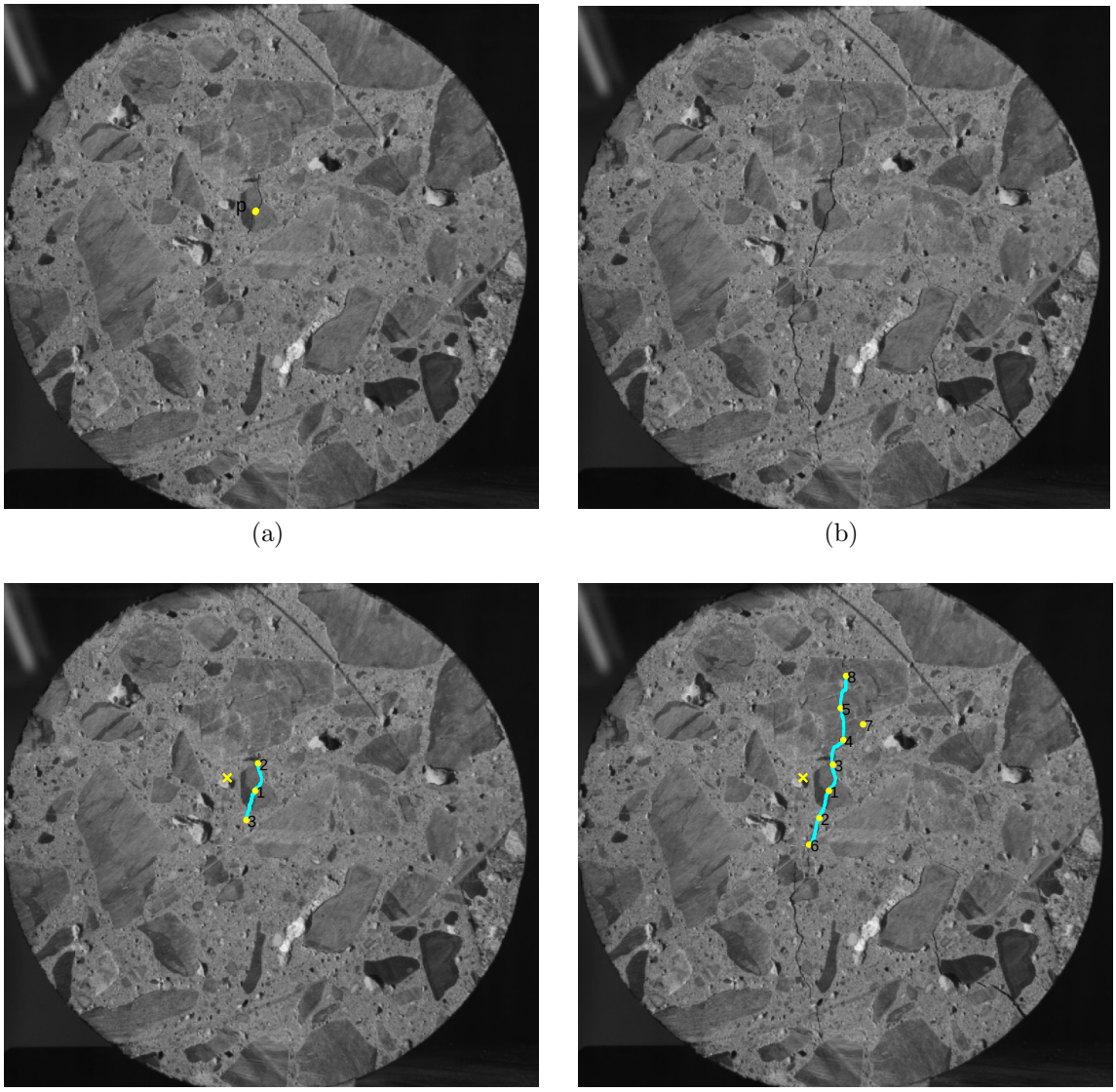
Figure 5.3: (a) First image and initial point p. (b) Second consecutive image. (c) Third consecutive image. (d) Detected crack map in first image with ordered keypoints and last keypoint that is used as starting point for next image. (e) Detected crack map in second image with ordered keypoints and last keypoint that is used as starting point for next image. (f) Detected crack map in third image with ordered keypoints and last keypoint that is used as starting point for next image. (g) Fourth consecutive image. (h) Fifth consecutive image. (i) Sixth consecutive image with end of continuous crack. (j) Detected crack map in fourth image with ordered keypoints and last keypoint that is used as starting point for next image. (k) Detected crack map in fifth image with ordered keypoints and last keypoint that is used as starting point for next image. (l) Detected crack map in first image with ordered keypoints and terminating point (marked by 'x').

distance λ from the end of the pavement image or the algorithm terminated on its own. If a new keypoint close to the end of a pavement image was detected, it was

used as the initial point for the next pavement image. This is possible because in real crack sealing applications overlapping images are acquired, so there is always some portion of the previous pavement image that is common to the next pavement image. Our experimental data did not have overlapping images, but we demonstrate this idea through six consecutive images. In the sixth image, which is shown in Figure 5.3l, the continuous crack ends as there is a break of length greater than λ between the two crack portions. The crack detection results were satisfactory in all the three data sets. Hence, our algorithm provides a unique application in the area of crack sealing.

5.1.3 Tracking Crack Growth

To simulate crack growth in critical structures like bridges, 3 concrete structure samples were subject to indirect tensile tests. Images with size 2448 $\times$ 2040 pixels were captured with the aid of the 5 Mega Pixel camera at the rate of 7 frames/second. However, indirect tensile tests were unsuitable for crack growth simulation because the sample structures ruptured under the application of tensile load. In future, strain tests will be conducted to get better sample images. We could find two images by which we could demonstrate the application of our algorithm. Figure 5.4a shows the concrete structure sample with a small crack. This is identified as the crack initiation step. An arbitrary point p is chosen on the crack in the image and the crack detection algorithm is applied to this image. Figure 5.4c shows the *RobustCurveDetection* algorithm results. In the image at the next time frame, which is shown in Figure 5.4b, the initiated crack has grown. We registered the two images (Figure 5.4a and 5.4b) so that the coordinates of the two images are aligned to one another. Next, the keypoints identified in the crack initiation step are used as initial points for the image at the next time frame shown in Figure 5.4b. Using these initial points (marked 1,2 and 3 in Figure 5.4a), we ran the *RobustCurveDetection* algorithm to detect the



(a) (b)

(c) (d)

Figure 5.4: (a) Image at crack initiation step (Stage 1) with initial point p. (b) Image with crack propagation (Stage 2). (c) Detected crack map in first image with ordered keypoints (including branch-points) and terminating point marked by 'x'. (d) Detected crack map in second image with ordered keypoints and terminating point marked by 'x'. (Keypoints from Stage 1 are used as initial conditions for Stage 2)

crack growth. The complete crack after the crack initiation and the crack growth stage with all keypoints in shown in 5.4d. In the future, more experimental images that are acquired from strain testing can be used to test the algorithm.

Table 4: Quantitative validation of crack images.

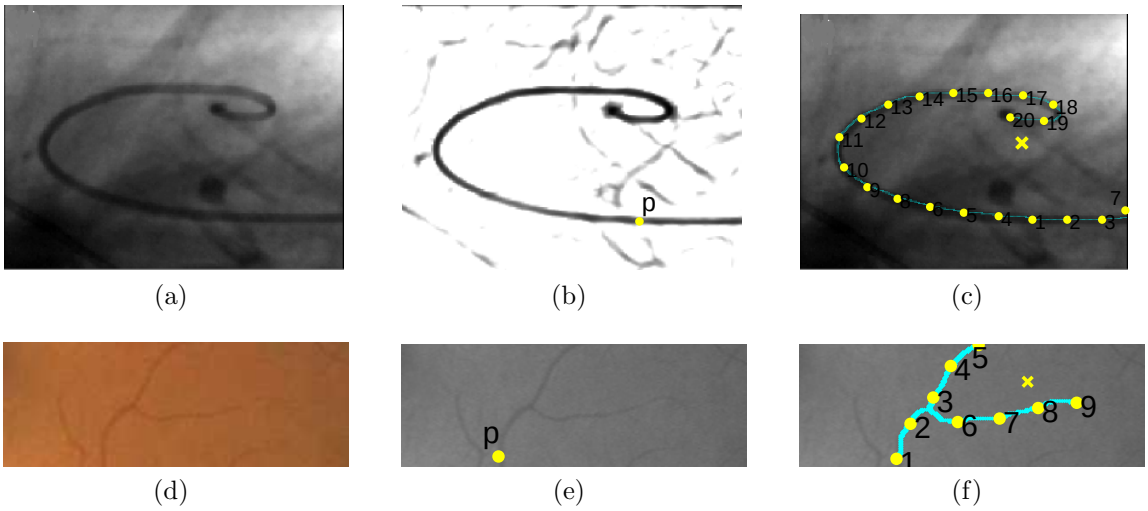
Image	TP(%)	FP(%)	FN(%)	SBD
Figure 5.1a	91.37	15.31	8.73	0.094
Figure 5.1b	91.21	3.1	8.79	0.0364
Figure 5.1c	91.38	13.52	8.62	0.1121
Figure 5.1d	92.35	18.1	7.65	0.121
Figure 5.2a	81.16	10.80	18.84	0.0693
Figure 5.3a	87.67	17.98	12.23	0.0319
Figure 5.3b	90.95	15.28	9.05	0.0236
Figure 5.3c	95.60	10.51	4.40	0.0238
Figure 5.3g	88.21	19.81	11.79	0.0311
Figure 5.3h	85.21	15.62	14.79	0.0895
Figure 5.3i	80.27	33.74	19.73	0.0773
Figure 5.4a	85.71	11.61	14.29	0.0353
Figure 5.4b	88.55	5.59	11.45	0.092
Average (25 images)	89.46	14.93	10.64	0.0721

5.1.4 Quantitative Validation

The ground truth for crack data was determined by GDOT engineers for 25 pavement images. They were asked to represent the crack with zero-width curves in the local neighborhood of the initial source point p. Table 4 presents the consolidated validation results of the crack images shown in this chapter. The validation results include the TP, FP, FN and SBD values. The average validation measure values for 25 pavement images, which were evaluated, is also provided in Table 4. The results show that our algorithm achieves consistently high TP rates. The FP rates and FN rates arise because of the fact that the algorithm adds a minimal path of length λ at every iteration step and this tends to slightly overestimate or underestimate the crack. In addition, the ground truth itself is approximate for crack images because it is manually drawn. The FP rate for Figure 5.3i is high because the total number of crack pixels in the ground truth is not high. Therefore, false detections form a large percentage of the total crack pixel detections. The SBD value, which was established in Chapter 4 as an excellent measure for quantitative validation of cracks, shows consistently good results for all images. The total average value of SBD is 0.0721, which corresponds to an absolute buffered distance (BD) value of 3.60 out of the maximum of 50. As ground truth accuracy of crack image is also suspect because it is derived from manual

hand markings, the SBD value indicates the good performance of our algorithm. In the future, more crack images can be tested to validate our algorithm.

5.2 Medical Applications



(a) (b) (c)

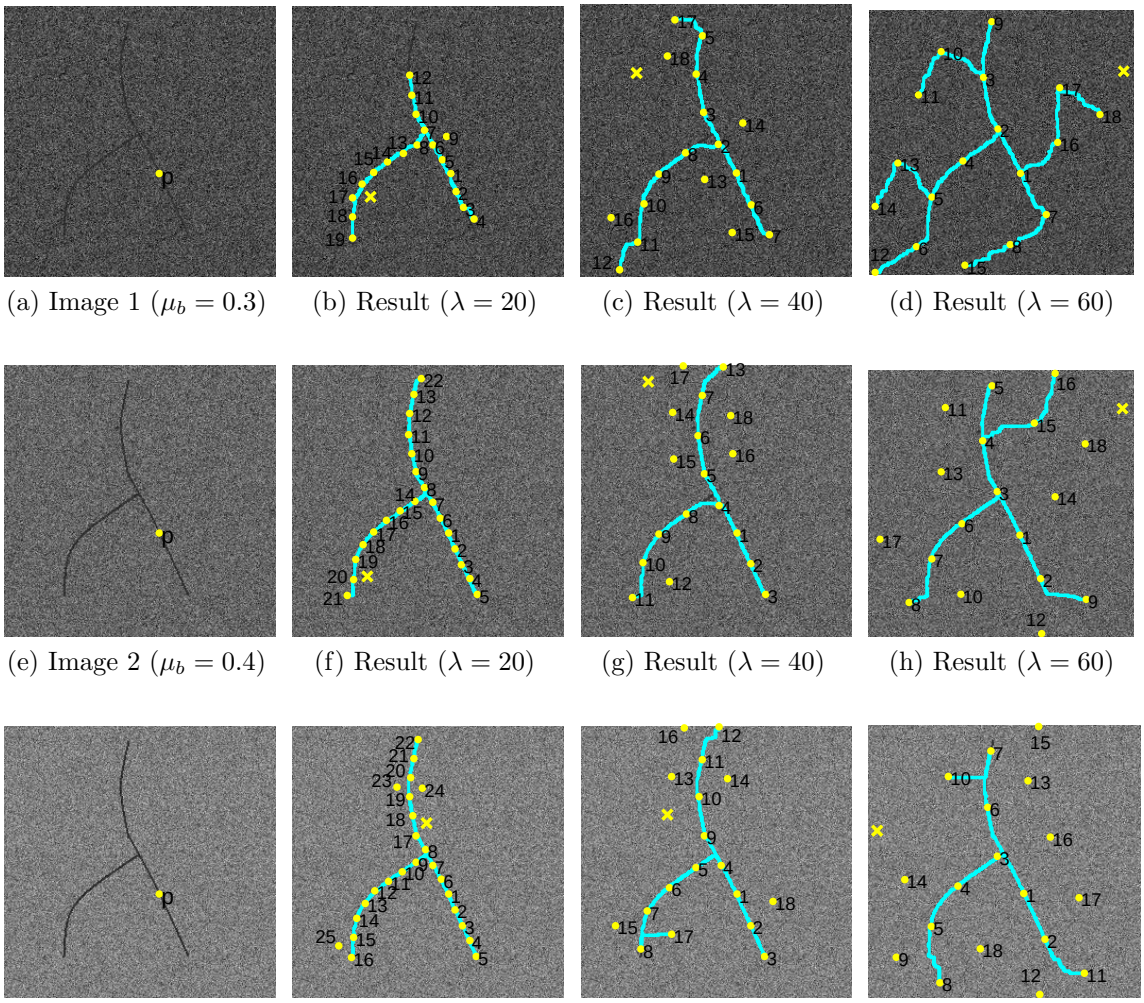
(d) (e) (f)

Figure 5.5: (a) Catheter tube image. (b) Edge based potential image. (c) Final catheter tube detection with ordered keypoints and terminating point (marked by 'x'). (e) Color Retinal image. (f) Grayscale image. (g) Final optical nerve detection with ordered keypoints and terminating point (marked by 'x').

We also used our algorithm to detect features or objects in medical images that can be modeled as thin curves. Figure 5.5a shows a medical image containing a catheter tube. A potential function based on the Laplacian is used to provide contrast between the desired feature and the background. Figure 5.5b illustrates the potential function image. Figure 5.5c shows the detected curve (catheter tube), keypoints and terminating point for the catheter tube image. We also applied our algorithm to a retinal image containing optical nerves shown in Figure 5.5d. The color image is converted to grayscale and shown in Figure 5.5e. An intensity based potential function was used as input for the *RobustCurveDetection* algorithm. Figure 5.5f shows the detected curve (optical nerve), keypoints and terminating point for the retinal image. These examples demonstrate the wide applicability of our algorithm.

In future, the algorithm can be tested on more medical images.

5.3 Sensitivity Analysis of Parameters λ and ϵ



(a) Image 1 ($\mu_b = 0.3$) (b) Result ($\lambda = 20$) (c) Result ($\lambda = 40$) (d) Result ($\lambda = 60$)

(e) Image 2 ($\mu_b = 0.4$) (f) Result ($\lambda = 20$) (g) Result ($\lambda = 40$) (h) Result ($\lambda = 60$)

(i) Image 3 ($\mu_b = 0.5$) (j) Result ($\lambda = 20$) (k) Result ($\lambda = 40$) (l) Result ($\lambda = 60$)

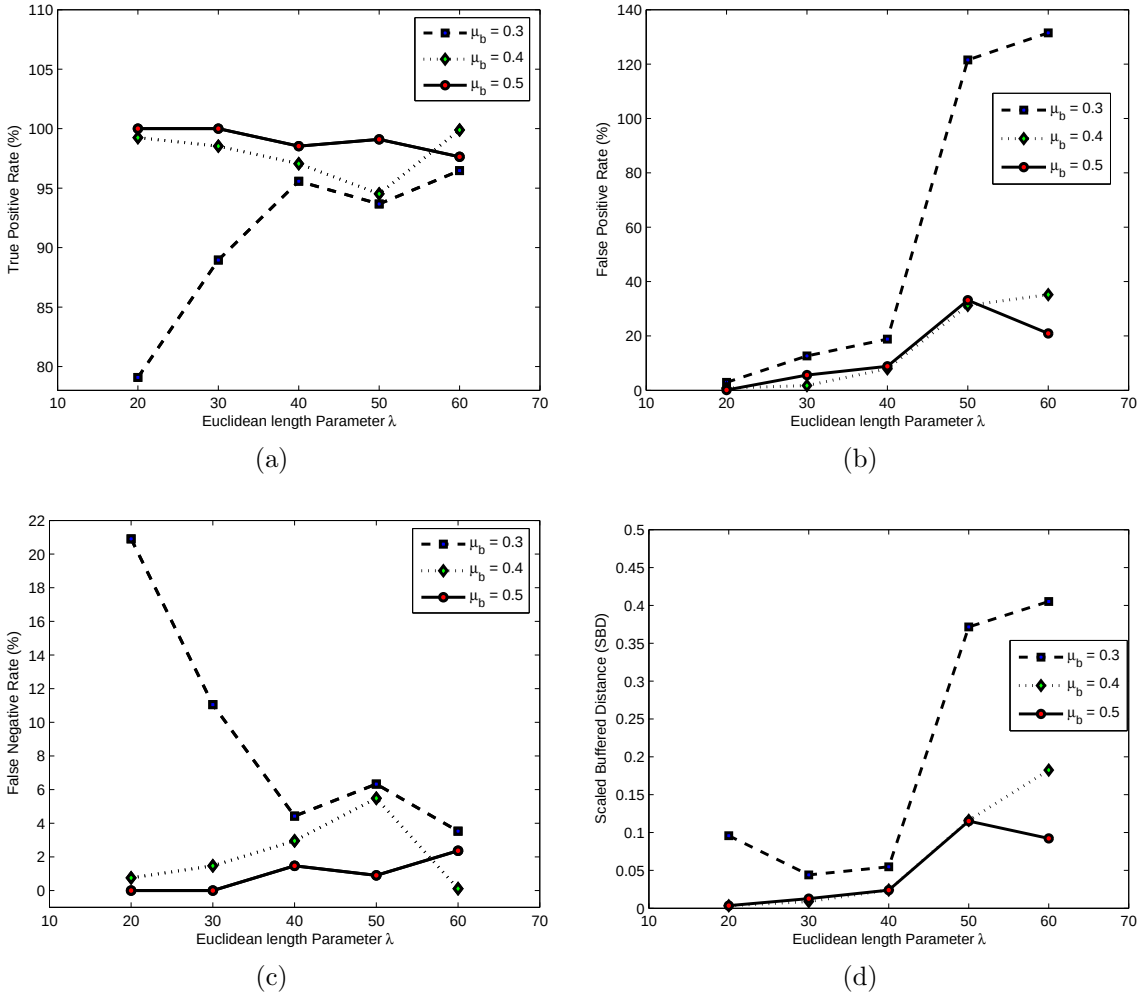
Figure 5.6: Synthetic Images corresponding to different background mean intensity $\mu_b = 0.3, 0.4, 0.5$ and algorithm results for $\lambda = 20, 40, 60$

We conducted an extensive quantitative study on sensitivity of parameter λ using synthetic images and kept parameter ϵ constant at 0.4λ. Four basic synthetic images that contain a simple curve, closed curve, curve with multiple branches and a complex topology curve (similar to Figures 3.7a, 3.9a, 3.10a and 3.12a used for algorithm illustration) were used for this validation study. The curves were chosen to be of mean

Table 5: Comparison of synthetic curve detection results obtaining by changing parameters λ and μ_b.

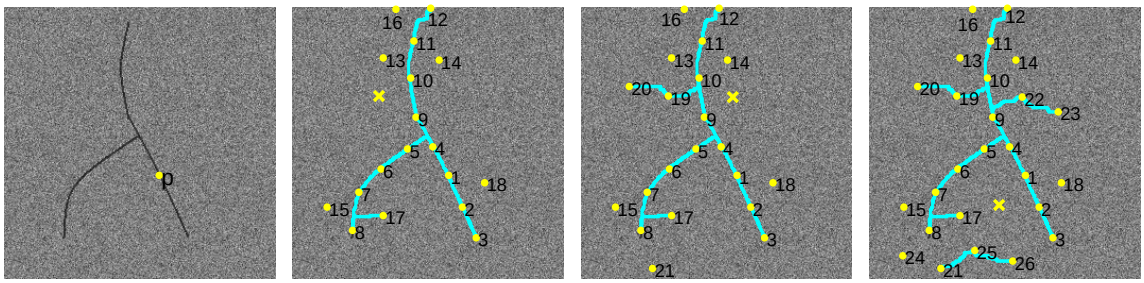
Measure	$\lambda = \mathbf{20}$	$\lambda = \mathbf{30}$	$\lambda = \mathbf{40}$	$\lambda = \mathbf{50}$	$\lambda = \mathbf{60}$
$\mu_b = 0.5$					
TP(%)	100	100	98.53	99.1	97.64
FP(%)	0.1	5.6	8.84	33.15	20.91
FN(%)	0	0	1.47	0.9	2.36
SBD	0.0035	0.0127	0.0239	0.1151	0.0923
$\mu_b = 0.4$					
TP(%)	99.25	98.53	97.05	94.52	99.89
FP(%)	0.86	1.77	7.95	31.24	35.2
FN(%)	0.75	1.47	2.95	5.48	0.11
SBD	0.0034	0.0089	0.0241	0.1161	0.1825
$\mu_b = 0.3$					
TP(%)	79.09	88.95	95.58	93.67	96.47
FP(%)	2.95	12.67	18.85	112.52	131.5
FN(%)	20.91	11.05	4.42	6.33	3.53
SBD	0.0958	0.0441	0.0547	0.3716	0.4051

intensity $\mu_c = 0.2$ and variance $\sigma_c = 0.05$. As we used a potential function based on intensity for our algorithm, we varied the mean intensity μ_b of the background to see its impact on the curve detection rate. The background variance σ_b was chosen to be 0.1. We found that when $\mu_b > 0.5$, the curve detection was very accurate and there was no impact of varying μ_b beyond a value of 0.5. Therefore, we chose μ_b to be 30,40 and 50 for our study. To determine the impact of the Euclidean length parameter λ on the curve detection, we varied λ between 20 to 60. μ_b and λ were varied for each of the four synthetic images and the average TP, FP, TN and SBD values for the detected curves were calculated. The results are tabulated in Table 5 and the plots for TP, FP, TN and SBD values are shown in Figure 5.7. Figure 5.6 shows one of the four synthetic images that contains a curve with multiple branches. The curve detection results for λ values 20,40 and 60 are illustrated for each of the three images generated using different μ_b. The plots and Figure 5.6 indicate that for higher μ_b values (0.4 and 0.5), a smaller λ value gives better curve detection results as indicated by the TP, FP, TN and SBD values. This is because the potential provides enough contrast



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

Figure 5.7: Plots indicating variation of true positive, false positive and false negative rates, and scaled buffered distance with changes in parameter λ and background intensity μ_d



(a) Synthetic Image (b) Result ($\epsilon = 0.2\lambda$) (c) Result ($\epsilon = 0.4\lambda$) (d) Result ($\epsilon = 0.6\lambda$)

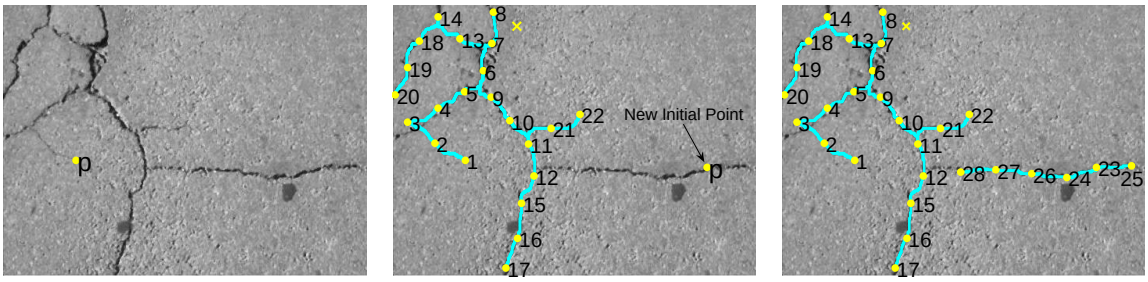
Figure 5.8: Synthetic image and algorithm results for $\epsilon = 0.2, 0.4, 0.6$

between the curve and the background and a smaller λ can help to terminate the *RobustCurveDetection* algorithm near the endpoints of the curve. Recall that the Euclidean length λ is the Euclidean distance L between detected keypoints and the curve is detected in minimum increments of λ. However, when μ_b is 0.3, a smaller λ value of 20 leads to a high FN rate. The reason for this is that a lower μ_b value provides poor contrast between the curve and the background, and a small interval λ is not sufficient for the keypoints to lie on the desired curve. Hence, the Equations (29) and (31) are not satisfied and the algorithm is terminated before the entire curve is detected. On the other hand as we increase the value of λ, the FN rate reduces and the FP rate increases. For $\mu_b = 0.3$ and λ values 50 and 60, the curve detection algorithm shows very high FP rates. The SBD plot in Figure 5.7d indicates that a value of λ between 30 and 40 gives the lowest value of SBD for $\mu_b = 0.3$. Therefore, for images with a poor potential function that does not provide enough contrast between the curve and the background, an optimal λ that minimizes SBD can be picked. In general, the results indicate that a good potential function that provides enough local contrast between the curve and the background is essential for the accuracy of our algorithm. This is true for all minimal path based techniques.

We also assessed the impact of the tolerance value ϵ on the curve detection algorithm. Recall that we used a tolerance ϵ for Condition (29) and (30) and tolerance $\epsilon/2$ for the incremental keypoint Condition (31). Our sensitivity study showed that ϵ values between 0.2λ and 0.6λ had very little impact on Conditions (29) and (30). On the other hand, ϵ value had an impact on Condition (31) because the incremental keypoint r in the (*RobustCurveDetection*) algorithm was restricted to originate only from the previous keypoint q. Hence, a higher ϵ value led to more false positives because many points that were not on the actual curve were able to meet a relaxed tolerance value for Condition (31). This fact is illustrated in Figure 5.8, which shows the curve detection results for ϵ values 0.2λ, 0.4λ and 0.6λ on a synthetic image.

Based on our study, we chose $\epsilon = 0.2\lambda$ for all experimental tests on real images. The high FP rates for λ values 50 and 60 seen in Table 5 can also be reduced substantially by chosing $\epsilon = 0.2\lambda$ instead of 0.4λ. The high FP rates for λ values 50 and 60 seen in Table 5 can also be reduced substantially by chosing $\epsilon = 0.2\lambda$ instead of 0.4λ. This concludes the sensitivity study results.

5.4 Extended User Interaction



(a) (b) (c)

Figure 5.9: (a) Complex crack and initial point p. (b) Crack detection results, ordered keypoints, terminating point (labeled by 'x') and new initial point p on undetected curve portion (c) Final crack detection with ordered keypoints and final terminating point labeled by 'x'.

Instead of fine tuning and adjusting the parameters λ and ϵ for different image data sets, the self-terminating algorithm also provides the user an option of interactively providing additional inputs. The algorithm can use the curve detection results from the first run of the algorithm and an additional user point in the undetected portion of the curve as starting conditions for next run of the algorithm. Figure 5.9 illustrates this principle. Figure 5.9a shows a complex crack and an initial point p provided by the user. Figure 5.9b shows the crack detection results, ordered keypoints and terminating point (labeled by 'x') after applying the *RobustCurveDetection* algorithm with parameters $\lambda = 45$ and $\epsilon = 0.2$. In additional, a new initial point p is selected in the undetected portion of the crack. Like before, a keypoint k_1 is detected from this new initial point p. However, once the keypoint k_1 is detected, the curve information is

integrated into the detected curve points set C, source point set S and endpoints set E that is obtained from the previous run of the algorithm. The *RobustCurveDetection* algorithm is then followed exactly like before. Figure 5.9b shows the new crack detection results after the second run of the algorithm. The sets C, S and E are empty for the first run of the algorithm. The details of this extended user interaction procedure are described in *ExtendedUserInteraction*.

Algorithm *ExtendedUserInteraction*
Input: Image Im, potential function Φ, previous source point set S, map $m(S)$, endpoints set E, curve detected so far C
Output: New Detected Curve C
1. Start with an arbitrary point p on the undetected portion of the curve.
2. Set $StopDetection = FALSE$.
3. Use $FMM(p, \lambda)$ to find keypoint k_1.
4. Run $MinimalPathbk(p, k_1)$ and set curve C' to contain the minimal path between p and k_1.
5. Set $S \leftarrow S \cup p \cup k_1$.
6. Set $C \leftarrow C \cup C'$.
7. Set $m(k_1) = p$ and $m(p) = k_1$.
8. Set $m_1(k_1) = 1$ (no. of neighbors is given by the map m_1.) and $m_1(p) = 1$.
9. Set $E \leftarrow E \cup p \cup k_1$.
10. Follow Step 9-38 of RobustCurveDetection algorithm.

5.5 Summary

We demonstrate the use of our *RobustCurveDetection* algorithm for three applications in identification of cracks in critical infrastructures like pavements and bridges: semi-automatic crack detection, detection of continuous cracks for crack sealing and tracking of crack growth in time. The consolidated quantitative validation of several crack images is also presented. Next, the algorithm results on a two medical images with thin elongated objects are shown. Finally, a detailed sensitivity study on the effect of algorithm parameter λ and ϵ. on the curve detection was conducted. Synthetic images with varying background intensities were used to conduct the analysis. Finally, the extension of the robust algorithm is described where user input can be

utilized at successive stages of the algorithm for more accurate curve detection results. The experimental results demonstrate that our algorithm can be used effectively for a wide range of applications.

CHAPTER VI

CONCLUSION

This thesis has successfully accomplished the intended research objectives. Initially, existing crack map detection methods were tested on experimental data sets and the limitations of these methods were highlighted . This motivated our use of minimal path techniques for three related problems of crack map detection in pavements and critical structures like bridges: tracking crack growth in critical structures like bridges after crack initiation, finding continuous cracks extending over several miles in pavements using the starting point of the crack, and semi-automatic detection of cracks in pavements. Existing minimal path techniques requires additional user input to detect features like cracks. To reduce this user input, several research directions were explored. This led to the development of a novel self-terminating minimal path algorithm that can detect curves of complex topology (including branches and closed cycles) from a single point on the curve (not necessarily an endpoint). This algorithm was tested and validated on an extensive data set containing both synthetic and real images. Apart from crack images in pavements and critical structures, this algorithm was also tested on medical images that contain objects of interest like catheter tubes and optical nerves, which can be modeled as curves.

The fundamental contribution of the thesis is the development of a novel algorithm to detect complex curves of arbitrary topology from a single point. It is a significant step in reducing the user interaction required for detecting curves using minimal path techniques. Earlier, both endpoints for a simple curve with no branches and all endpoints for curves with multiple branches were required to detect such curves using minimal path methods. However, the novel self-terminating algorithm is able

to detect complex curves using just one arbitrary point. The power of the new algorithm was demonstrated by using it for multiple applications in crack detection and in detecting features in medical images. Another contribution of the thesis was the development of the buffered distance measure for quantifying the performance of crack map detection algorithms. Crack map detection is the crucial step in analyzing the condition of pavements and critical structures like bridges and a quantification measure is needed to effectively measure the performance of crack map detection algorithms. However, due to the unique nature of the crack map detection problem, traditional quantification measures like mean square error and statistical correlation are not able to distinguish between the performance of different algorithms effectively. This led to the development of the buffered distance measure that is based on modifications of the Hausdorff distance. Using both real and synthetic images, it was demonstrated that the buffered distance is superior to four existing scoring measures in distinguishing between the performance of different crack detection algorithms.

There are several future research directions that can be pursued. First, the self-terminating minimal path algorithm can be extended to detect higher dimensional curves. In particular, it will be useful to detect tubular structures like vessels where these structures are modeled as 3D and 4D curves [29] (2D structures as 3D curves and 3D structures as 4D curves). An extra dimension is introduced to model the width of tubular structure. Even wide cracks can be modeled as higher dimensional curves to capture crack width information. Currently, further user interaction is needed to detect these structures using the method formulated by Hua and Yezzi [29]. Our novel algorithm has the potential to substantially reduce this user interaction. Second, work can be done on complete automation of the curve detection process where the user does not have to supply any point on the curve. More extensive testing on image that have complex structural cracks and features like thin capillaries and bone cracks in medical images can be conducted. Third, the algorithm can

be tested on pavement depth data acquired using a continuous laser profiler. It is expected that depth information can be used to construct a potential that will be better than intensity information because depth information is not affected by shadows, oil stains and lighting conditions. Finally, code optimization can be done to improve the computational speed of the current algorithm. The current algorithm implementation does not exploit redundant information between different iterations. This redundant information can be used to make the algorithm faster and applicable to real-time computation scenarios.

We have explored an important problem of crack detection, proposed a novel algorithm based on a principled theoretical framework of minimal path methods and extended this algorithm to other applications. We expect that this thesis will be very beneficial to researchers exploring the problem of detection of objects or features that can be modeled as curves.

REFERENCES

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

[2] Adalsteinsson, D. and Sethian, J. A., "A fast level set method for propagating interfaces," *Journal of Computational Physics*, vol. 118, no. 2, pp. 269 – 269, 1995.

[3] Alekseychuk, O., "Detection of crack-like indications in digital radiography by global optimisation of a probabilistic estimation function," *BAM-Dissertationsreihe*, 2006.

[4] Ardon, R., Cohen, L. D., and Yezzi, A., "Fast surface segmentation guided by user input using implicit extension of minimal paths," *Journal of Mathematical Imaging and Vision*, vol. 25, no. 3, pp. 289–305, 2006.

[5] Beauchemin, M., Thomson, K. P. B., and Edwards, G., "On the hausdorff distance used for the evaluation of segmentation results," *Canadian Journal of Remote Sensing*, vol. 24, no. 1, pp. 3–8, 1998.

[6] Benmansour, F. and Cohen, L. D., "Fast object segmentation by growing minimal paths from a single point on 2D or 3D images," *Journal of Mathematical Imaging and Vision*, vol. 33, no. 2, pp. 209–221, 2009.

[7] Bonneau, S., Dahan, M., and Cohen, L. D., "Single quantum dot tracking based on perceptual grouping using minimal paths in a spatiotemporal volume," *IEEE Transactions on Image Processing*, vol. 14, no. 9, pp. 1384–95, 2005.

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

[9] Chen, Z. and Hutchinson, T. C., "Application of pde methods for image-based concrete surface damage detection," in *Nondestructive Characterization for Composite Materials, Aerospace Engineering, Civil Infrastructure, and Homeland Security*, vol. 6531, (San Diego, California, USA), pp. 65310N1–12, 2007.

[10] Cheng, H.-D., Chen, J.-R., and Glazier, C., "Novel fuzzy logic approach to pavement distress detection," in *Nondestructive Evaluation of Bridges and Highways*, vol. 2946, (Scottsdale, AZ, USA), pp. 97–108, SPIE, 1996.

[11] Cohen, L. D., "Multiple contour finding and perceptual grouping using minimal paths," *Journal of Mathematical Imaging and Vision*, vol. 14, no. 3, pp. 225–236, 2001.

[12] Cohen, L. D. and Kimmel, R., "Global minimum for active contour models: a minimal path approach," *International Journal of Computer Vision*, vol. 24, no. 1, pp. 57–78, 1997.

[13] Committee, P. M. S. A., "Delivery of pavement distress data from automated systems," tech. rep., Transportation Research Board, 2007.

[14] Committee, T. R. B. E., "NCHRP synthesis 334 automated pavement distress collection techniques," tech. rep., Transportation Research Board of the National Academies, 2004.

[15] Cuhadar, A., Shalaby, K., and Tasdoken, S., "Automatic segmentation of pavement condition data using wavelet transform," in *IEEE Canadian Conference on Electrical and Computer Engineering*, vol. 2, pp. 1009–1014, 2002.

[16] Danielsson, P.-E. and Lin, Q., *Image Analysis*, ch. A Modified Fast Marching Method, pp. 631–644. Lecture Notes in Computer Science, Springer Berlin/Heidelberg, 2003.

[17] Davis, E. R., *Machine Vision: theory, algorithms, practicalities*. Morgan Hanfmann Publishers, CA, USA, 2005.

[18] De Rivaz, P. and Kingsbury, N., "Complex wavelet features for fast texture image retrieval," in *IEEE International Conference on Image Processing*, vol. 1, pp. 109–113, 1999.

[19] Deschamps, T., Letang, J. M., Verdonck, B., and Cohen, L. D., "Automatic construction of minimal paths in 3D images: an application to virtual endoscopy," in *Computer Assisted Radiology and Surgery. Proceedings of the 13th International Congress and Exhibition*, (Amsterdam, Netherlands), pp. 151–5, Elsevier Science, 1999.

[20] Diaz, F. V., Armas, A. F., Kaufmann, G. H., and Galizzi, G. E., "Nondestructive evaluation of the fatigue damage accumulation process around a notch using a digital image measurement system," *Optics and Lasers in Engineering*, vol. 41, no. 3, pp. 477–487, 2004.

[21] EI-Korchi, T., Gennert, M., Ward, M., and Wittels, N., "Lighting design for automated pavement surface distress evaluation," *Transportation Research Record:Journal of the Transportation Research Board*, vol. 1311, pp. 144–148, 1991.

[22] Gangone, M. V., Whelan, M. J., and Janoyan, K. D., "Wireless sensing system for bridge condition assessment and health monitoring," in *Smart Sensor Phenomena, Technology, Networks, and Systems*, vol. 7293, (San Diego, CA, USA), pp. 72930M1–12, 2009.

[23] Gonzalez, J. and Knauss, W. G., "Strain inhomogeneity and discontinuous crack growth in a particulate composite," *Journal of the Mechanics and Physics of Solids*, vol. 46, no. 10, pp. 1981–1995, 1998.

[24] Gonzalez, R. C. and Woods, R. E., *Digital Image Processing*. Prentice Hall, 2nd ed., 2002.

[25] Guo, S. and Fei, B., "A minimal path searching approach for active shape model (ASM)-based segmentation of the lung," *Proceedings of the SPIE - The International Society for Optical Engineering*, vol. 7259, pp. 72594B1–8, 2009.

[26] Hassouna, M. S. and Farag, A. A., "Multistencils fast marching methods: A highly accurate solution to the eikonal equation on cartesian domains," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, pp. 1563–1574, 2007.

[27] Hong-Gyoo, S., Yun-Mook, Lim, K.-H., and Yun, G.-H. K., "Monitoring crack changes in concrete structures," *Computer-Aided Civil and Infrastructure Engineering*, vol. 20, no. 1, pp. 52–61, 2005.

[28] Hou, Z., Wang, K. C. P., and Gong, W., "Experimentation of 3d pavement imaging through stereovision," in *International Conference on Transportation Engineering*, (Chengdu, China), pp. 376–381, 2007.

[29] Hua, L. and Yezzi, A., "Vessels as 4-D curves: global minimal 4-D paths to extract 3-D tubular surfaces and centerlines," *IEEE Transactions on Medical Imaging*, vol. 26, no. 9, pp. 1213–23, 2007.

[30] Huang, Y. and Xu, B., "Automatic inspection of pavement cracking distress," *Journal of Electronic Imaging*, vol. 15, no. 1, pp. 0130171–6, 2006.

[31] Huo, X., Chen, J., and Donoho, D., "Multiscale detection of filamentary features in image data," *SPIE Wavelet-X*, 2003.

[32] Hutchinson, T. C. and Chen, Z., "Improved image analysis for evaluating concrete damage," *Journal of Computing in Civil Engineering*, vol. 20, no. 3, pp. 210–216, 2006.

[33] Iyer, S. and Sinha, S. K., "A robust approach for automatic detection and segmentation of cracks in underground pipeline images," *Image and Vision Computing*, vol. 23, no. 10, pp. 921–933, 2005.

[34] Iyer, S. and Sinha, S. K., "Segmentation of pipe images for crack detection in buried sewers," *Computer-Aided Civil and Infrastructure Engineering*, vol. 21, no. 6, pp. 395–410, 2006.

[35] James, W. F., Steve, E. W., Jagannathan, S., and Maciej, Z., "Embeddable sensor mote for structural monitoring," in *Sensors and Smart Structures Technologies for Civil, Mechanical, and Aerospace Systems*, vol. V1-8, p. 6932, 2008.

[36] Kass, M., Witkin, A., and Terzopoulos, D., "Snakes: active contour models," *International Journal of Computer Vision*, vol. 1, no. 4, pp. 321 – 31, 1987.

[37] Kaul, V., Tsai, Y., and Mersereau, R., "A quantitative performance evaluation of pavement distress segmentation algorithms," *Transportation Research Record: Journal of Transportation Research Board*, 2010.

[38] Kerekes, J., "Receiver operating characteristic curve confidence intervals and regions," *IEEE Geoscience and Remote Sensing Letters*, vol. 5, no. 2, pp. 251–5, 2008.

[39] Kim, Y. S., Yoo, H. S., Lee, J. H., and Han, S. W., "Chronological development history of x-y table based pavement crack sealers and research findings for practical use in the field," *Automation in Construction*, vol. 18, pp. 513 – 524, 2009.

[40] Kirschke, K. R. and Velinsky, S. A., "Histogram-based approach for automated pavement-crack sensing," *Journal of Transportation Engineering*, vol. 118, no. 5, pp. 700–710, 1992.

[41] Koutsopoulos, H. N. and Downey, A. B., "Primitive-based classification of pavement cracking images," *Journal of Transportation Engineering*, vol. 119, no. 3, pp. 402–418, 1993.

[42] Koutsopoulos, H. N., El Sanhouri, I., and Downey, A. B., "Analysis of segmentation algorithms for pavement distress images," *Journal of Transportation Engineering*, vol. 119, no. 6, pp. 868–888, 1993.

[43] Kumar, A. and Pang, G. K. H., "Defect detection in textured materials using gabor filters," *Industry Applications, IEEE Transactions on*, vol. 38, no. 2, pp. 425–440, 2002.

[44] Li, K. and Fei, B., "A new 3D model-based minimal path segmentation method for kidney MR images," in *2nd International Conference on Bioinformatics and Biomedical Engineering*, (Shanghai, China), pp. 2342–2344, IEEE, 2008.

[45] Mallat, S. and Zhong, S., "Characterization of signals from multiscale edges," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 14, no. 7, pp. 710–732, 1992.

[46] Nazef, A., Mraz, A., Gunaratne, M., and Choubane, B., "Experimental evaluation of a pavement imaging system: Florida department of transportation's multipurpose survey vehicle," *Transportation Research Record:Journal of the Transportation Research Board National Research Council*, vol. 1974, pp. 97–106, 2006.

[47] Oh, H., Garrick, N. W., and Achenie, L. E., "Segmentation algorithm using iterated clipping for processing noisy pavement images," in *Imaging Technologies: Techniques and Applications in Civil Engineering: Proceedings of the 2nd International Conference*, pp. 138–147, 1997.

[48] Patil, P. M., Biradar, M., and Jadhav, S., "Orientated texture segmentation for detecting defects," in *IEEE International Joint Conference on Neural Networks*, vol. 4, pp. 2001–2005, 2005.

[49] Randen, T. and Husoy, J. H., "Filtering for texture classification: a comparative study," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 21, no. 4, pp. 291–310, 1999. 0162-8828.

[50] Rethore, J. and Roux, F. H. S., "Extended digital image correlation with crack shape optimization," *International Journal for Numerical Methods in Engineering*, vol. 73, no. 2, pp. 248–272, 2008.

[51] Ryu, D. H. and Nahm, S. H., "Image processing techniques applied to automatic measurement of the fatigue-crack," *Key Engineering Materials*, vol. I, pp. 34–39, 2005.

[52] Sethian, J. A., "Fast marching methods," *SIAM Review*, vol. 41, no. 2, pp. 199–235, 1999.

[53] Shah, S., "Automatic cell image segmentation using a shape-classification model," in *IAPR Conference on Machine Vision Applications*, (Tokyo, Japan), 2007.

[54] Sun, B.-C. and Qiu, Y.-J., "Automatic identification of pavement cracks using mathematic morphology," in *International Conference on Transportation Engineering 2007, ICTE 2007*, 2007.

[55] Tagashira, H., Arakawa, K., Yoshimoto, M., Mochizuki, T., Murase, K., and Yoshida, H., "Improvement of lung abnormality detection in computed radiography using multi-objective frequency processing: Evaluation by Receiver Operating Characteristics (ROC) analysis," *European Journal of Radiology*, vol. 65, pp. 473–477, 2008.

[56] Tomczak, L. and Mosorov, V., "Singular value decomposition for texture defect detection in visual inspection systems," in *Proceedings of the 2nd International Conference on Perspective Technologies and Methods in MEMS Design*, pp. 131–133, 2006.

[57] Tsai, Y., Kaul, V., and Mersereau, R., "Critical assessment of pavement distress segmentation methods," *ASCE Journal of Transportation Engineering*, vol. 136, pp. 11–19, 2010.

[58] Tsitsiklis, J. N., "Efficient algorithms for globally optimal trajectories," *IEEE Transactions on Automatic Control*, vol. 40, no. 9, pp. 1528–1538, 1995.

[59] Wang, H., Zhu, N., and Wang, Q., "Segmentation of pavement cracks using differential box-counting approach," *Journal of the Harbin Institute of Technology*, vol. 39, no. 1, pp. 142–4, 2007.

[60] Wang, K. C. P., "Challenges and feasibility for comprehensive automated survey of pavement conditions," in *Proceedings of the International Conference on Applications of Advanced Technologies in Transportation Engineering*, pp. 531–536, 2004.

[61] Wang, K. C., Li, Q., and Gong, W., "Wavelet-based pavement distress image edge detection with trous algorithm," in *Transportation Research Board Meeting*, (Washington D.C, USA), 2007.

[62] Wang, K. C. P., "Designs and implementations of automated systems for pavement surface distress survey," *Journal of Infrastructure Systems*, vol. 6, no. 1, pp. 24–32, 2000.

[63] Wang, K. C. P. and Gong, W., "Real-time automated survey of pavement surface distress," in *Proceedings of the International Conference on Applications of Advanced Technologies in Transportation Engineering*, pp. 465–472, 2002.

[64] Wang, K. C. P. and Gong, W., "Real-time automated survey system of pavement cracking in parallel environment," *Journal of Infrastructure Systems*, vol. 11, no. 3, pp. 154–164, 2005.

[65] Wang, K. C. P. and Tee, W., "Understanding of pavement profiling and validation of an implementation," in *Traffic and Transportation Studies Proceedings of ICTTS 2002*, vol. 2, (Guilin, China), pp. 817–824, 2002.

[66] Wang, K. C., Gong, W., and Hou, Z., "Automated cracking survey," in *RILEM International Conference on Cracking in Pavements*, (Chicago, Illinois), pp. 881–889, 2008.

[67] Wang, Y., "Image matching based on robust hausdorff distance," *Journal of Computer Aided Design & Computer Graphics*, vol. 14, no. 3, pp. 238–41, 2002.

[68] Xiaomei, S., Pogue, B. W., Dehghani, H., Shudong, J., Paulsen, K. D., and Tosteson, T. D., "Receiver operating characteristic and location analysis of simulated near-infrared tomography images," *Journal of Biomedical Optics*, vol. 12, no. 5, pp. 0540131–10, 2007.

[69] Xu, B., "Artificial lighting for the automated pavement distress rating system," tech. rep., Center for Transportation Research, University of Texas at Austin, 2005.

[70] Yan, P. and Kassim, A. A., "Medical image segmentation using minimal path deformable models with implicit shape priors," *IEEE Transactions on Information Technology in Biomedicine*, vol. 10, no. 4, pp. 677–84, 2006.

[71] Yu, L., Giurgiutiu, V., Ziehl, P., and Ozevin, D., "Piezoelectric based sensing in wireless steel bridge health monitoring," in *Nondestructive Characterization for Composite Materials, Aerospace Engineering, Civil Infrastructure, and Homeland Security*, vol. 7294, (San Diego, CA, USA), pp. 72941D1–10, 2009.

[72] Zhang, D., Nazari, A., Soappman, M., Bajaj, D., and Arola, D., "Methods for examining the fatigue and fracture behavior of hard tissues," *Experimental Mechanics*, vol. 47, no. 3, pp. 325–336, 2007.

[73] Zhang, D. and Arola, D. D., "Applications of digital image correlation to biological tissues," *Journal of Biomedical Optics*, vol. 9, no. 4, pp. 691–699, 2004.

[74] Zhou, J., Huang, P., and Chiang, F.-P., "Wavelet-based pavement distress classification," *Transportation Research Record*, no. 1940, pp. 89–98, 2005.

[75] Zhou, J., Huang, P. S., and Chiang, F.-P., "Wavelet-aided pavement distress image processing," in *Wavelets: Applications in Signal and Image Processing X*, vol. 5207, (San Diego, CA, USA), pp. 728–739, 2003.

[76] Zhou, J., Huang, P. S., and Chiang, F.-P., "Wavelet-based pavement distress detection and evaluation," *Optical Engineering*, vol. 45, no. 2, pp. 027007–10, 2006.

[77] Zhu, D., Qi, Q., Wang, Y., Lee, K.-M., and Foong, S., "A prototype mobile sensor network for structural health monitoring," in *Nondestructive Characterization for Composite Materials, Aerospace Engineering, Civil Infrastructure, and Homeland Security*, vol. 7294, (San Diego, CA, USA), pp. 72941A1–10, 2009.