VISUALIZATION AND INTELLIGENT SOLUTIONS FOR BIG PAVEMENT DATA

By

YUE FEI

Bachelor of Science in Computer and Communication Engineering
Southwest Jiaotong University
Chengdu, China
2011

Master of Science in Civil Engineering
Oklahoma State University
Stillwater, OK, USA
2015

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
DOCTOR OF PHILOSOPHY
July, 2018

VISUALIZATION AND INTELLIGENT SOLUTIONS

FOR BIG PAVEMENT DATA

Dissertation Approved:

Dr. Kelvin C.P. Wang

Dissertation Adviser

Dr. Qiang (Joshua) Li

Dr. Stephen A. Cross

Dr. Keith A. Teague

ACKNOWLEDGEMENTS

I would like to express the deepest appreciation to my advisor, Dr. Kelvin C.P. Wang, for his professional guidance and persistent inspirations in my doctoral work. Dr. Wang's enthusiasm, innovative view on research and rigorous academic standard have tremendously influenced my study. Dr. Wang gave me the freedoms and encouragements to study whatever I am interested in, so that I dared to independently explore novel knowledge areas such as advanced computer programming and algorithms related to Automated Pavement Survey and artificial intelligence. I am grateful to have Dr. Wang as my Ph.D. advisor.

My special words of appreciation should also give to Dr. Joshua (Qiang) Li, Dr. Stephen A. Cross and Dr. Keith A. Teague. I thank Dr. Li for his continuous academic guidance and support throughout my Ph.D. research work. In addition to our academic collaboration, I greatly value his personal suggestions that make me grow and become more mature. I am grateful to Dr. Stephen A. Cross for his valuable and systematic lectures that significantly enhanced my understanding of pavement engineering, guided a "computer guy" to civil engineering. My heartfelt thanks to Dr. Keith A. Teague for his technical advice in his lecture of Digital Signal Processing, which inspired me to develop better solutions for accuracy improvements on pavement groove detection.

I sincerely thank Dr. Cheng Chen for his constant efforts to help me in computer programming techniques. I am very much privileged to closely work with him in past years.

Special thanks to my lab teammates: Allen Zhang, Guangwei Yang, Baoxian Li, You Zhan, Yang Liu, Weiguo Gong, Justin Thweatt, Ran Ji, Shi Qiu, Wenjian Wang, Lin Li, Wenting Luo and many others for their assistance and inspirations.

I must thanks to my personal friends in my Ph.D. study career, Qingang Hu, Zihe Chen, Chuanyi Sui, Shaoqing Huang, Jian Huang, Zhikui Chen and many more for giving me happiness during my graduate life.

Appreciation also goes to faculty and staffs at Oklahoma State University for all of their assistance throughout my Ph.D. program.

Last but not least, I am deeply thankful to my parents, relatives, teachers and friends in China for their love, mentoring, and support. I am very much indebted to my aunt, Dr. Suli Fei (UC Berkeley), who lives and works in Silicon Valley. In past years, she always shared her experience throughout my work and life, and supported me to complete my doctoral study as my surrogate parent.

Name: YUE FEI

Date of Degree: July, 2018

Title of Study: VISUALIZATION AND INTELLIGENT SOLUTIONS FOR BIG PAVEMENT DATA

Major Field: Civil Engineering

ABSTRACT: Pavement visualization and crack detection are two important components supporting modern pavement condition survey. In this dissertation, two major goals are accomplished based on implementable algorithms. 1) The long-distance pavement 3D visualization is developed based on an effective Level-of-Details (LOD) algorithm named "Geometry Clipmap". The Geometry Clipmap is able to render large-scale 3D scene by caching pavement height data in a set of nested grids. During 3D rendering process based on Geometry Clipmap, the data size uploaded to video memory is reduced significantly, while the detailed high-resolution pavement surface data are still retained for further detailed inspection. As a consequence, Geometry Clipmap provides long-distance pavement display with excellent visual continuity and stable frames per second (FPS). In addition, the size of large-area surface distresses can be directly measured under long-distance pavement 3D environment. This is the first study that applies LOD algorithm to long-distance 3D pavement visualization and large-scale distress measurement. 2) The pixel-level automated crack detection is achieved through the modifications of Cell-based Convolutional Neural Network (CNN) on 3D pavement surface. CNN is a deep learning algorithm that targets at recognizing Cell-based imaging objects. Based on combination of specific pre-process layers and new gradient computational strategy, the network for pixel-level detection is developed in the study for effective feature learning. In addition, consecutive convolutional layers and a new activation unit are introduced to improve the detection performance at fine and shallow cracks. Furthermore, GPU parallel computing techniques are utilized to speed up the proposed pixel-level detection network. As a result, the CNN based pixel-level crack detection outperforms traditional imaging algorithms in terms of F-Measure. Lastly, the highly-efficient Deep-Learning network for speedy processing fits well for big pavement datasets processing.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1 INTRODUCTION

1.1 Background

Roads are the fundamental infrastructure contributing to national productivity and employment. Among all the transportation systems (including railway, air, waterway, etc.), road transportation system can handle 87.6% of passenger-miles and 40.9% of freight ton-miles (Bureau of Transportation Statistics, 2012). Every $1 billion invested in highway roads would support 13,000 jobs in construction and maintenance sectors, in supplier industries, and throughout the economy (FHWA, 2017). In addition, road network system plays an important role in daily traveling, commodity delivery, military equipment and personnel transfer, evacuation of natural disaster victims, etc.

To provide and sustain the road system in a condition that is acceptable to the traveling public at the least life cycle cost, the American Association of State Highway and Transportation Officials (AASHTO) defines a series of pavement management activities related to pavement planning, budgeting and programming, design, construction, monitoring, maintenance and rehabilitation, etc. (AASHTO, 1985). All of these management activities are treated in a systematic and coordinated manner according to Pavement Management System (PMS). In fact, the effective decisions made by PMS on management activities heavily relies on accurate and timely pavement condition data. In other words, the accuracy and efficiency of data collection and analysis are critical to pavement management activities.

Pavement condition survey is one essential PMS component for pavement data measurement and collection (Timm and McQueen, 2004; Attoh-Okine and Adarkwa, 2013; Wang et al., 2015). The condition survey data includes surface distress data, structural deformation, ride quality, skid resistance, etc. (CDOT, 2013; Abdelaty et al., 2015; National Research Council, 1970; FHWA, 2006; Carvalho et al., 2012; Sayers et al., 1986; Sayers et al., 1986; Highway Research Board, 1972). Traditionally, most of pavement condition is investigated in a manual manner: either walking or windshield survey on the roads. By this manual approach, pavement condition data is identified and measured by well-trained inspectors or engineers in the field. However, conventional manual approaches based on visual inspection and field measurement are found to be subjective, time-consuming, expensive and even hazardous for engineers (Sivaneswaran et al., 2004). Besides, there are almost 4.1 million miles of roads in the United States, with over 45,000 miles composing the Interstate Highway System (Sullivan, 2006). Such a large road network is impossible to be completely investigated accurately and efficiently only by human resources.

In recent years, automated pavement condition survey has gained increasingly acceptance by most transportation agencies (Pierce et al., 2013). The automated pavement survey involves two major tasks: automated data collection and interpretation (Wang and Smadi, 2011). The automated pavement data collection utilizes customized vehicles fitting with advanced sensing equipment to collect pavement images or videos by traveling on the road at or near highway speeds. Due to technological innovations in hardware equipment such as cameras or lasers, automated data collection tends to become increasingly accurate and popular. Multiple agencies have been utilizing customized optical devices for years to implement automated data collection in both 2D and 3D pavement images (Koch and Brilakis, 2011; Mancini et al., 2013; Laurent et al., 2008; Laurent et al., 2012; Moreno et al., 2013). Most notably, the Digital Highway Data Vehicle (DHDV) (**Fig. 1.1**) equipped with PaveVision 3D Ultra System, which is developed by WayLink Systrems Corp. at Oklahoma State University (OSU), is able to acquire full-lane-scale pavement 2D and 3D data in 1-mm resolution quality at a highway speed up to 60 mph during night- or day-time (Wang et al., 2011).

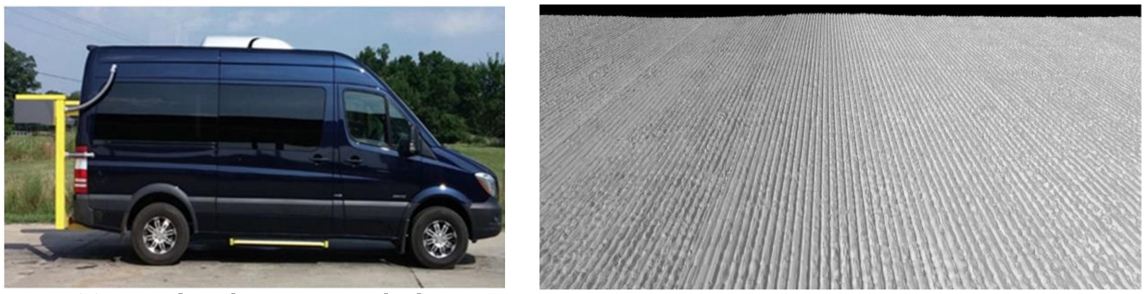

(a) Digital Highway Data Vehicle **(b)** 3D Pavement Data Collected by DHDV

Figure 1.1. DHDV (WayLink) & 1-mm Resolution 3D Pavement Data

1.2 Problem Statement

1.2.1 Emergence of Big Pavement Data

Data collection is no longer a challenge by using laser imaging in recent years for pavement condition survey. The remaining problem is the pavement data processing or interpretation that can provide valuable information for pavement management activities. However, pavement data interpretation is an extremely difficult task due to the big data size and complexity:

- Pavement data size is undergoing an explosion driven by automated data collection universalization. During the past few decades, pavement engineers only needed to process data in the size of MegaBytes for pavement management activities. At the present time, however, pavement data size skyrockets from MegaBytes to TeraBytes because of automation in data collection and high resolution data. In other words, such a huge increase of data volume makes efficient data interpretation more difficult.
- Pavement data complexity and diversity continue to increase due to the use of higher resolution sensors, which poses great difficulties to develop effective algorithms for data interpretation. In the past, many automatic analysis algorithms are developed based on

observations and assumptions of specific small datasets. These algorithms tend to lose their effectiveness due to the increasingly complexity and diversity in modern big pavement data. In other words, effective and highly-generalized algorithms are strongly desired in the complex and diverse big data interpretation.

1.2.2 Primary Challenges

Visualization and automated crack detection are two major tasks of pavement data interpretation. Despite tremendous efforts in the past, pavement visualization and crack detection still face challenges:

- Visual inspection is inevitable in pavement condition survey in the form of automation or manual processing. For instance, one of the most widely used indices in pavement engineering is the Pavement Condition Index (PCI) that rates pavement condition between value 0 and 100 (U.S. Army, 1982). Essentially, the PCI is calculated based on field visual survey of identifying the number and types of distresses on pavement surface. At present, field pavement observation tends to be replaced by the image-based pavement observation in the office with rapid development of computer technology. As an important component for the office image-based observation, 3D virtual pavement is usually generated by either stereo imaging or laser triangulation. Nevertheless, due to the limitations on computer memory and processing capability, 3D virtual pavement visualization usually uses partial pavement data, resulting in only a short-distance pavement display. The short-distance 3D virtual pavement is not able to offer a whole pavement viewing for both visual inspection and surface distress measurement.
- Pavement crack information is valuable for pavement management and planning. Pavement cracking is a typical type of pavement surface distresses caused by various

factors such as repeated heavy traffic loads, temperature cycles, poor pavement mix design, etc. (FHWA, 2006). The continuing development of pavement cracks will significantly reduce pavement load-bearing capacity, resulting in bad ride quality, high cost for travelling, or even potential safety hazard to traveling publics. Thus, accurate and timely surface crack information is required by transportation agencies to plan pavement maintenance activities. Over the past decades, many efforts have been dedicated to automating pavement crack detection. However, such automations still face tremendous challenge in terms of accuracy and efficiency due to difficulties in computerizing human cognition for crack detection.

1.3 Objectives

The objectives of this dissertation are to develop automated algorithms for both large-scale pavement 3D visualization and advanced pixel-level detection of surface cracks based on 1-mm 3D laser pavement images. The sub-objectives are summarized as following:

- Comprehensive literature reviews are conducted to assess traditional approaches for pavement visualization and surface crack detection
- LOD based algorithm is implemented to achieve long-distance pavement 3D visualization and large-area surface distress measurement
- Convolutional Neural Network is utilized to perform crack detection on grid-like image cells, which is a preliminary research on deep learning
- Convolutional Neural Network is modified to have capacity for training at each pixel of pavement images, resulting in the detection accuracy can be improved from the Cell-based to pixel-level

- Efficient convolutional strategy and new activation unit are proposed to improve pixel-level detection performance at fine and shallow cracks
- General-purpose computing on graphics processing units (GPGPU) are utilized to accelerate network computation for high-efficient processing based on large pavement datasets.

According to the objectives aforementioned, the structure of this dissertation is outlined as follows:

- Chapter 1: Introduction
- Chapter 2: Literature Review
- Chapter 3: Three-D (3D) Visualization for Long Pavement Surfaces
- Chapter 4: A Deep Learning System for Cell-based Automated Crack Detection
- Chapter 5: A Deep Learning Methodology for Pixel-Level Detection of Pavement Cracks
- Chapter 6: Conclusions and Future Work

Chapter 2 LITERATURE REVIEW

2.1 Visual Pavement Distress Inspection

Accurate, consistent, and repeatable pavement condition survey plays a vital role in the pavement management activities. The major target of pavement condition survey is to identify and quantify the amount and severity of surface distress in a given segment of pavement (ODOT, 2010). At present, automatic distress analysis based on imaging data tends to become popular in pavement condition survey (SD DOT, 2017). However, it is still necessary to interconnect the visual inspection into surface distress analysis because of the following two reasons. In the first place, the accuracy and consistency of automatic distress assessment based on imaging data are sometimes under a certain of skepticism due to data complexity and diversity in pavement texture (Timm and McQueen, 2004). In this case, human visual inspection can be regarded as an alternative method to provide reliable surface distress information (Metro Nashville, 2018). Furthermore, not all pavement distresses can be detected automatically. For example, polished aggregate (**Fig. 2.1**) is a type of pavement distress defined as surface binder worn away to expose coarse aggregate (FHWA, 2003). There are only limited research activities so far in automatic polished aggregate detection due to the lack of high-resolution data (Coenen and Golroo, 2017). Under this circumstance when automatic analysis is not available, the visual inspection is the unique approach to providing distress data.

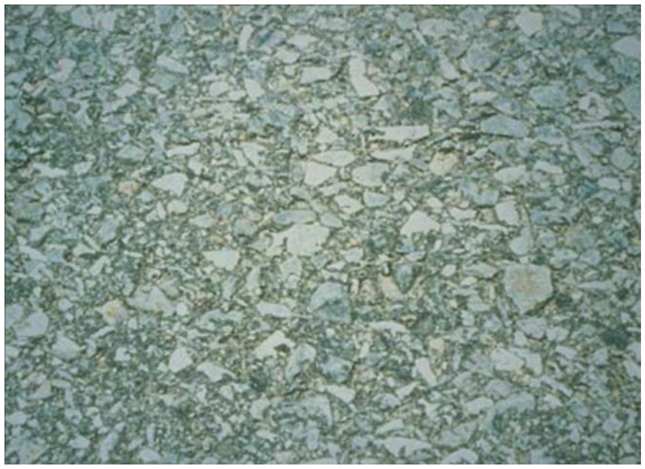
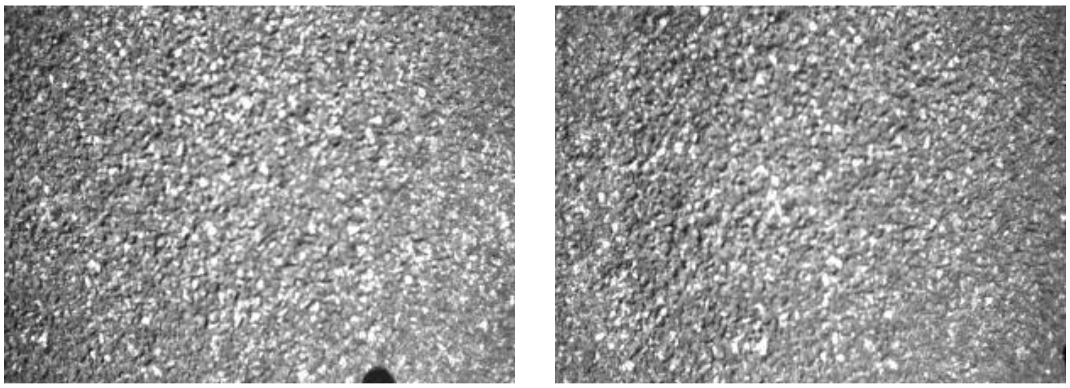


Figure 2.1. Pavement Distress Example – Polished Aggregate

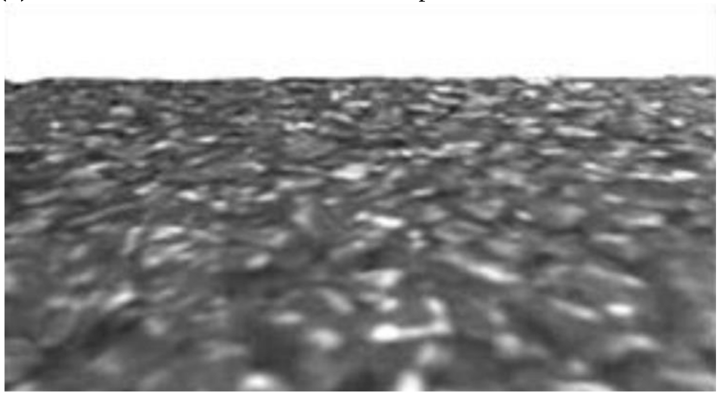
Although visual inspection is inevitable, it does not mean the return to traditional field survey. In fact, the office image-based visual inspection is gradually replacing the old-style fieldwork with the rapid development of automated data collection and computer graphics. Comparing to traditional field inspection, the office survey is considered as a safer and more efficient manner. Through digital computers in the office, surface distresses in automated imaging data can be identified and classified elaborately by well-trained personnel following evaluation criteria of either the “PCI ASTM Standard”, the “Distress Identification Manual for the Long-Term Pavement Performance Program”, or other protocols.

Currently, there are two common types of pavement imaging data: 2D and 3D data. The early office survey is usually based on 2D images. However, the 2D data encounters problem of losing information on realistic pavement surface characteristics with height characteristics (e.g. cracks, rutting, shoving, potholes, etc.), which poses difficulties to accurately identify surface distresses. On the other hand, the surface characteristics lost in 2D data can still be captured and stored in the 3D data, which is helpful to improve accuracy for both visual inspection and automatic analysis (Wang et al., 2004; Wang et al., 2011; Wang et al., 2015). In recent years, 3D data analysis become increasingly popular due to the technological innovations in data collection equipment. In the meanwhile, visual inspection in the office as well is based more on 3D imaging visualization.

Stereo 3D pavement surface visualization are commonly implemented by following two techniques. The first technique is called “Stereo Imaging” that reconstructs virtual 3D pavement images by using couples of 2D images at different viewpoints. According to stereo imaging principle, 3D information is extracted by analyzing and matching feature points in different 2D images (**Fig. 2.2**) (Saleh et al., 2016). However, this technique lacks in accuracy as well as requires a lot of computational power due to the fact that it is a conversional process between 2D and 3D (Zhang and Elaksher, 2012; Lei et al., 2014; Mathavan et al. 2015; Coenen and Golroo, 2017). Another technique is to utilize 3D Graphic API for stereo pavement visualization based on 3D imaging data collected by laser sensors. By using 3D graphic API, pavement geometrical characteristics can be visualized directly in stereo virtual pavement surface (**Fig. 2.3**). The effects of virtual pavement visualization are directly determined by the actual quality of automated 3D imaging data (i.e. the height maps or 3D models). However, due to the big pavement data size and the limited resources of computer hardware, 3D pavement surface is usually visualized as a short-distance pavement section based on single image instead of a long adequate part for representing the state of the pavement as a whole. Besides, some types of large surface distresses (e.g. cracks, patches, etc.) may be sectioned and distributed inside two or more consecutive images, which causes inconvenience for their size measurement if pavement surface is visualized by single image.



(a) Stereo Pair of Pavement Surface Captured with a Stereo Camera



(b) 3D Pavement Surface Constructed from Stereo Images Shown Above

Figure 2.2. Three-D (3D) Pavement Surface Reconstruction based on Stereo Matching

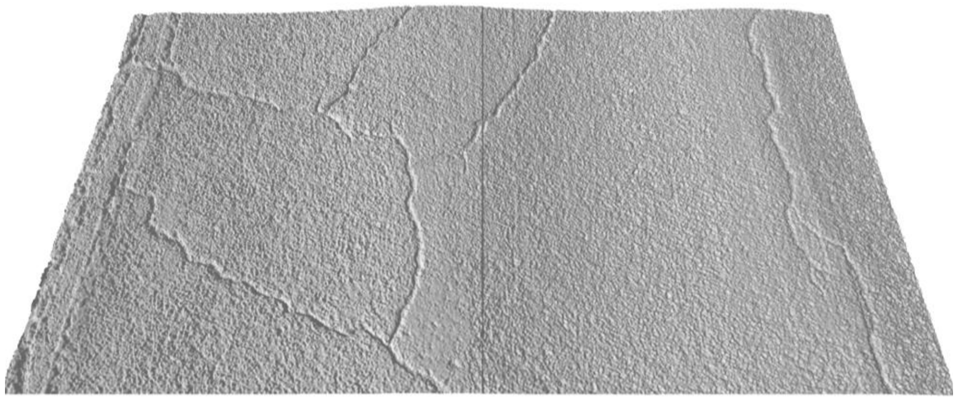


Figure 2.3. Pavement Surface Visualization based on 3D Graphic API

2.2 Automated Pavement Crack Detection

Although researches on automating identification and classification of other pavement distresses have been studied for many years, the pavement crack detection still receives a great deal of interests in pavement engineering (Wang and Smadi, 2011). As one major task of automated condition survey, traditional pavement crack detection has been demonstrated at a certain level of success. Thresholding algorithms were proposed to separate cracks from local background (Koutsopoulos and Downey, 1993; Elbehiery et al., 2005; Teomete et al., 2005; Katakam, 2009; Oliveira and Correia, 2009). However, the thresholding algorithms failed to achieve high accuracies for most of the time due to unevenly distributed illuminance on images. Although Morphological Algorithms were also introduced to reduce false detections (Tanaka and Uematsu, 1998; Coster and Chermant, 2001; Iyer and Sinha, 2005; Oliveira and Correia, 2009), the detection results were still heavily dependent on the choices of threshold values. Wavelet-based Approaches utilized Wavelet Transform to decompose the original pavement data into different frequency sub-bands so that cracks can be captured more easily following the assumption that cracks are mainly preserved in high frequency sub-bands (Zhou et al., 2005; Wang et al., 2007; Chambon et al., 2009; Liang and Sun, 2010; Georgieva et al., 2015). Nevertheless, the decomposition of original data into frequency domain adversely impacts the spatial entirety of pavement cracks and thus may result in discontinued cracks. Filter-based Algorithms were also introduced by researchers to transform original data and seek for cracks with anticipated responses (Chaudhuri et al., 1989; Abas and Martinez, 2003; Subirats et al., 2006; Chambon et al., 2009; Zhang et al., 2013). However, the filter-based approaches may fail to detect some cracks that have weak responses to the predesigned filters. Machine Learning Algorithms, such as Artificial Neural Networks (ANNs) and Support Vector Machines (SVMs), can also be found in many studies to perform classifications on pavement cracks (Kaseko and Ritchie, 1993; Chou and Cheng, 1994; Hsu et al., 2001; Nguyen et al., 2009; Moussa and Hussain, 2011; Daniel and Preeja, 2014). However, these machine learning techniques generally represent only one or two layers of abstraction and may not fully reflect the complexity of pavement surface.

The traditional automated algorithms above for crack detection have demonstrated various successes under specific pavement environments. However, none of them have demonstrated consistently high accuracies on diverse road surfaces. Over the past decades, the private and public endeavors on worldwide basis vastly underestimated the challenges and difficulties in developing automated analysis tools for pavement cracks with full considerations on the complexity as well as the diversity of pavement surfaces (Zhang et al., 2016a). The limitations of traditional automated algorithms are primarily resulted from attempts to mimic human cognition capability based on shallow level of abstractions or simple and uncomprehensive hypotheses (Zhang et al., 2017b).

The recent progresses in deep learning show an opportunity to improve the performance of automated crack detection algorithms through learning from diverse examples. Deep learning is an effective machine learning approach that allows computers to understand the world by learning multiple levels of data representations from experiences (LeCun et al. 2015; Goodfellow et al., 2016). The powerful adaptability of deep learning for complex problems has been demonstrated in many recent researches (Sutskever et al., 2014; Jean et al., 2015; Silver et al. 2016; Shrivastava et al., 2016; Ledig et al., 2016). Meanwhile, deep learning based transportation solutions were also developed recently for traffic data imputation (Duan et al., 2016), traffic flow prediction (Polson et al., 2017), and passenger demand forecasting (Ke et al., 2017).

Automated pavement crack detection also experiences a technical revolution inspired by deep learning, especially by the deep Convolutional Neural Network (CNN). The CNN structure is specifically designed to classify grid-like images based on shift- and distortion-invariant features obtained via sparse connectivity combined with shared weights and pooling layers (LeCun et al., 1998a). Since AlexNet won the champion in ImageNet Large Scale Visual Recognition Competition (ILSVRC) in 2012, deep CNN structure has been continuously improved and revealed remarkable capability in image classification based on large-scale comprehensive datasets (Krizhevsky et al., 2012; Russakovsky et al., 2014; Szegedy et al., 2014; Simonyan and Zisserman, 2014; He et al. 2016). Based on the CNN structure, several applications have also been introduced in pavement crack detection. An application of CNN in crack recognition was developed for classifying small image cells with size of 99×99, demonstrating the superior performances of deep CNN in comparison with Support Vector Machines (SVMs) and Boosting methods (Zhang et al., 2016b). Similarly in 2017, another deep CNN was proposed for crack recognition based on pavement image cells and further showed the advantages of deep learning techniques over traditional algorithms such as Canny and Sobel edge detectors (Cha et al., 2017). However, both of the two aforementioned networks were not developed on the basis of true pixel-level accuracy, and did not train the networks with a large set of diversified pavement surface data representing varying textures. Also in 2017, an improved CNN network named “CrackNet” was developed for crack detection based on a large set of diverse 3D pavement surface data (Zhang et al., 2017b). Unlike the traditional CNN architecture, CrackNet does not use any pooling layers that downsize the original data. Consequently, the supervised learning can be conducted at pixel level with invariant data size. It was demonstrated that CrackNet achieved high level of pixel-perfect accuracy in detecting cracks on asphalt surfaces and outperformed traditional algorithms such as SVM and 3D Shadow Modeling (Zhang et al., 2017a). However, the feature generator used in CrackNet implements fixed operations and produces handcrafted features using pre-designed line filters, which resulted in some limitations in learning capability. More importantly, the processing speed of CrackNet is slow due to huge number of parameters and large data depth at hidden layers.

2.3 Summary

Visual inspection is inevitable in modern pavement condition survey when automatic algorithms are unavailable or not accurate enough. Currently, the office visual inspection based on imaging data tends to replace the old-style field inspection due to the rapid development of automated data collection and computer graphic techniques. As the basic requirement of office visual inspection, 3D pavement surface reconstruction can be implemented by either stereo matching or 3D graphic API. However, stereo matching lacks in accuracy as well as efficiency due to the conversion from 2D to 3D data. On the other hand, 3D graphic API is unable to visualize long enough pavement for general condition evaluation and distress measurement due to limited capacity of computer graphic cards. Therefore, one objective in this dissertation is to investigate 3D graphic algorithms for better reconstructing 3D pavement geometrical characteristics.

Secondly, as major components of automatic distress survey, traditional pavement crack detection algorithms encounter the problems of losing accuracy and consistency due to diverse, complex, and massive pavement texture data. The Convolutional Neural Network (CNN), as one of the deep learning based algorithms, is proposed in the dissertation to improve the performance of automatic crack detection based on imaging data. However, most of the CNN algorithms for crack detection are at Cell-based, which are not practical for providing detailed crack information. Even though some pixel-level CNN based crack detections are proposed, the low-efficient speed of them is not fitting for processing big pavement datasets. Therefore, another objective of this dissertation is to explore the improvements of CNN algorithm to achieve high-accurate and high-efficient pixel-level crack detection.

Chapter 3 Three-D Visualization for Long Pavement Surfaces

3.1 Introduction to Level-of-Details (LOD) Methodologies

Three-dimensional (3D) graphics is an important subject of computer graphics, aiming at creating vivid or life-like visual representations using computer. The first public appearance of 3D graphics was in the movie Star Wars: A New Hope in 1977, which generated a great deal of interests in 3D graphic studies at that time. Afterwards, due to a series of successes in 3D graphic algorithm studies, the modern application area of 3D graphics covers user interfaces to scientific simulations, product designs, special effects in movies, and even full-length animated filters. Nowadays, 3D technologies are inseparable from various practical applications such as Google Earth, 3D GPS Navigation, cartography projects, urban planning and architecture, website advertisement, video games, etc. (Dicker, 2003; Pulli et al., 2008; Ruzinoor et al, 2012; Kim, 2004; Google Inc., 2018; Garmin Ltd., 2018).

The process of automatically generating a photorealistic 2D images based on 3D datasets using computer is called "3D Rendering". The 3D datasets can be obtained by either actual data collection or virtual data creation. Since 3D geometrical datasets can be too complex to be rendered in a desired performance with limited computer resources (i.e. CPU, GPU, RAM, etc.), an algorithm called "Level-of-Detail (LOD)" is proposed for geometry simplification of a 3D object as it moves away from the viewer (Luebke and Hallen, 2001). As shown in **Fig. 3.1**, the same object can be represented by 3D model at different detail-levels. The more detailed 3D model is applied to render an object in high resolution if it is close to viewer, while the less detailed 3D model is applied to represent a small, distant, or unimportant object. In other words, the LOD algorithm is able to find a balance between 3D object fidelity and rendering performance by automatically picking object at different detail-levels.

The LOD frameworks can be divided into three basic groups (Luebke et al., 2003): Discrete LOD, Continuous LOD, and View-dependent LOD. The original scheme of discrete LOD (Clark, 1976) was applied to pre-create multiple models for an object at different LODs in advance of 3D rendering. Only one appropriate LOD model is selected during 3D rendering based on its distance to viewer or other similar criteria. But sometimes discrete LOD is not the best approach for 3D object simplification, especially for those large objects (e.g. stadium) that could be subdivided into small objects, or those small objects (e.g. machine components) that could be combined as a whole.

Instead of pre-processing LODs, continuous LOD approach creates a simplified data structure to extract desired LOD extraction in a real time. This method results in better fidelity since the LOD for each object is exactly specified rather than chosen from fixed number of preprocessed results. Meanwhile, because no redundant polygons are used to display an object, computer resources can be better utilized for other objects' rendering.

View-dependent LOD, derived from Continuous LOD, is able to obtain the best representation of the current visual field based on various viewing parameters. In this case, a single object may have several different levels of detail to be rendered at one time. For example, the adjacent portion of an object can be shown at high resolution while the farther portion goes to the opposite. This approach provides the best overall fidelity and enables excellent simplification of huge objects such as large stadium or outdoor terrain, or a long pavement section at 1-mm resolution.

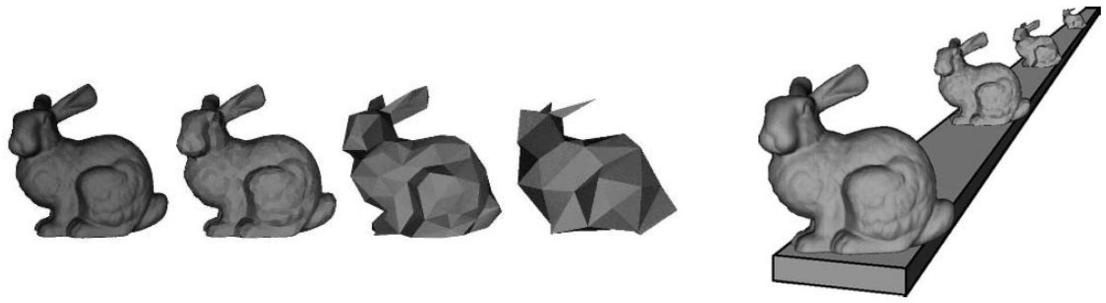


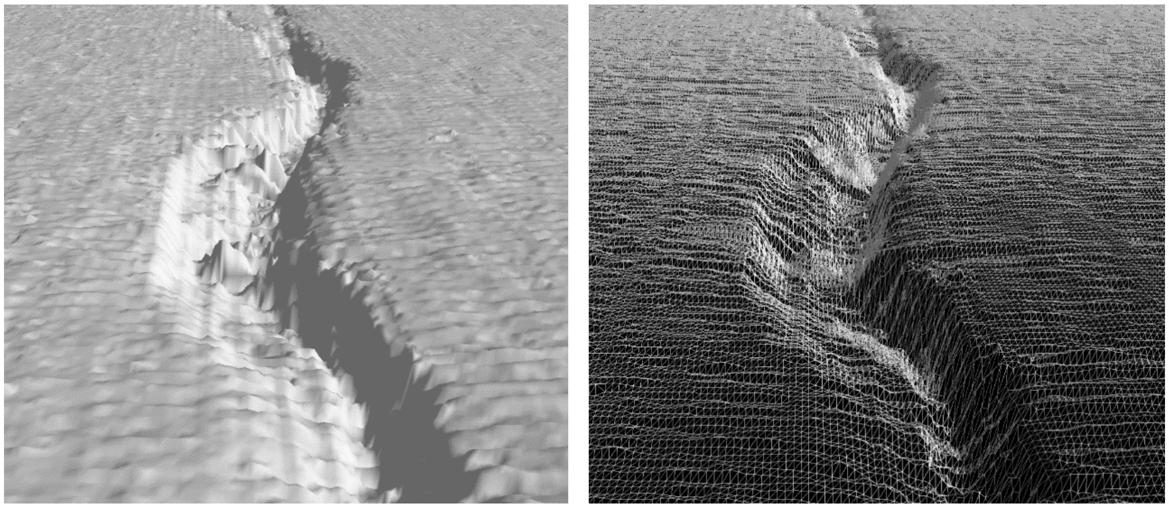
Figure 3.1. Concept of Level-of-Details (LOD)

3.2 Geometry Clipmap

Generally, the pavement surface geometry in 3D graphic API is rendered in the form of numerous adjacent triangles (**Fig. 3.2**). The more triangles are rendered, the more computer resources are consumed. However, the number of triangles will skyrocket dramatically to cover a long pavement surface geometry. Such a great number of triangles can lead to significant decrease of rendering performance or even stop on rendering. In this case, geometry clipmap is a good algorithm to reduce the number of triangles that are required to be rendered so that the long pavement surface geometry can be displayed.

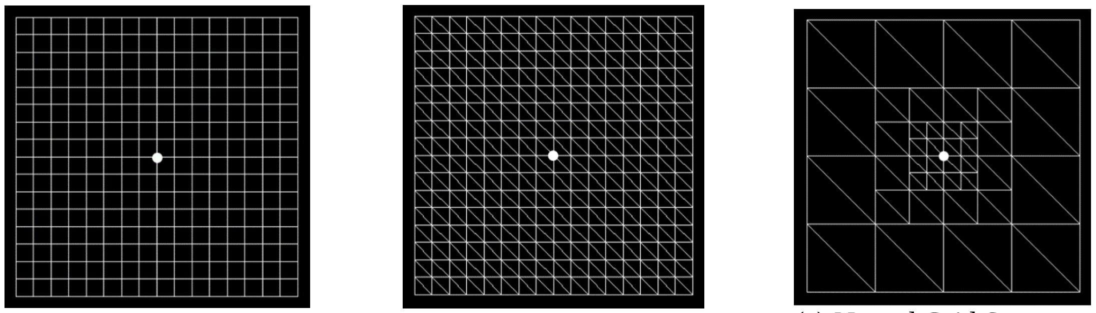
Geometry clipmap utilizes a nested grid structure to effectively decrease the number of rendered triangles for huge 3D terrain visualization (Losasso and Hoppe, 2004). Assuming a viewer is located in the center of a height map, only the geometry close to the viewer is rendered with high-resolution triangles, while the geometry far away from the viewer is rendered with low-resolution triangles. **Fig. 3.3** illustrates an example of nested grids to represent a height map. For a height map with 17 * 17 pixels (**Fig. 3.3(a)**), traditional rendering method displays the height map for all the evenly distributed discrete triangles (**Fig. 3.3(b)**), and the number of triangles is 512 (i.e. 16 * 16 * 2). For the nested grids structure (**Fig. 3.3(c)**), it uses 3 different sizes of triangles, and the number of rendered triangles decreases to 80. Comparing with traditional method, nested grid structure reduces the geometrical complexity of a terrain and therefore improve 3D rendering performance for large-scale 3D scene visualization.

In addition, the nested grid structure renders multiple height maps with the same size but with different resolution levels at one time, as shown in **Fig. 3.4**. Height map at the finest level of data is pre-filtered (**Fig. 3.4(a)**), and the same size of height data is grabbed at different levels of height maps and stored as multiple 2D elevation textures in the program flow (red, green and blue region). Combined with corresponding scaling and translation factors, 2D elevation textures are rescaled and projected onto the same plane (**Fig. 3.4(b)**). Only the finest level data is rendered as a 1:1 scale grid square (red region), while other level data is enlarged and rendered as a hallow ring (green & blue region), and finally a regular nested grid structure is established (**Fig. 3.4(c)**).



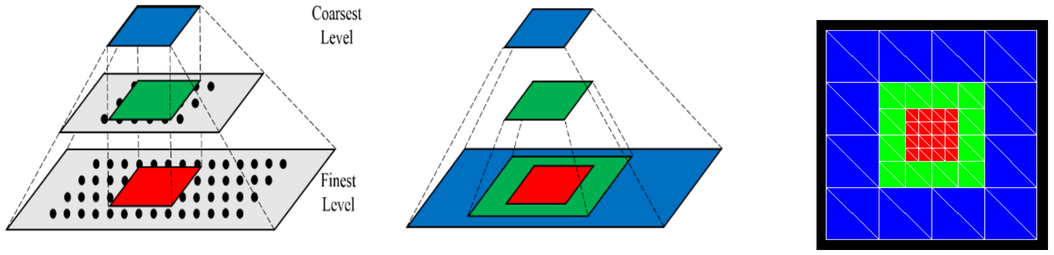
(a) 3D Rendering in Graphic API **(b)** Numerous Adjacent Triangles

Figure 3.2. Pavement Surface Geometry in 3D Graphic API



(a) 17 * 17 Height Map **(b)** Traditional Rendering **(c)** Nested Grid Structure

Figure 3.3. How the Nested Grids Represent a Height Map



(a) Height Maps with Different Levels **(b)** 2D Elevation Textures Projection **(c)** Nested Grid Generation

Figure 3.4. How Geometry Clipmap Works with Multiple Height Maps

3.3 Long Pavement Surface Visualization

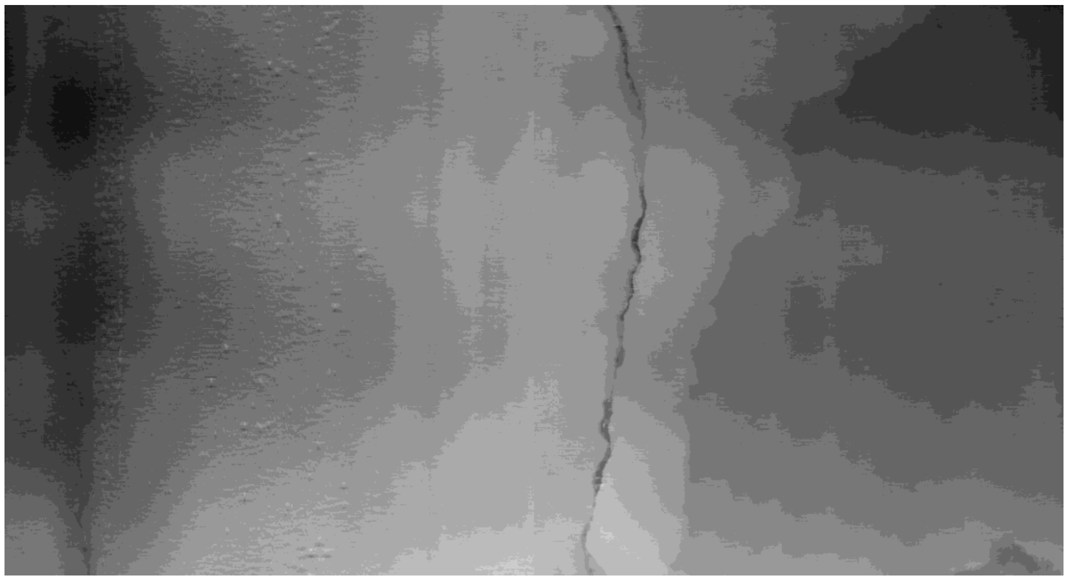
3.3.1 Initial Implementation

Traditionally, 3D scene is generated based on three steps in OpenGL graphic API pipeline: 3D model determination, lighting model establishment and texture mapping if it is required. In this dissertation, the first two steps are applied for the initial implementation of 3D pavement.

The first step of 3D pavement display is to determine pavement surface geometry. In computer graphics, an object is made of a series of adjacent triangles that define the outer surface of an object, and such series of triangles are often called "geometry" (McKesson, 2012). As **Fig. 3.2** shown before, the pavement surface geometry is generated from thousands of adjacent triangles. Those triangle vertices coordinate data (3D height data) is collected using the 3D Ultra laser imaging technology at 60 mph (100 km/h) and stored as a series of raw image data with the size of 4096 pixel wide and 2048 pixel long. Through the programmable graphic rendering pipeline, the 3D coordinate data from each raw image can be converted to 2D image that is projected onto screen.

Lighting model, an important part of 3D graphic rendering pipeline, is vital for determination of how pavement surface interacts with lights. Properly modeling the interaction between light source and pavement surface is critical in restoring the realistic visual effects of pavement features, such as crack, pothole, groove etc. Combined with ambient light, diffuse light, specular light, and emissive light, lighting model computes the pixel colors for every rendered triangle vertex to simulate the realistic lighting effects on the final projected 2D image. In this dissertation, Gaussian Lighting Shader is applied to compute the colors of every triangle vertex for composition of pavement surface.

Fig. 3.5 shows the height map converted from one raw image which provides pavement surface geometry information. **Fig. 3.6** shows the 3D pavement surface with Gaussian Lighting Model.



(a) 3D Height Map Displayed in Grayscale Form

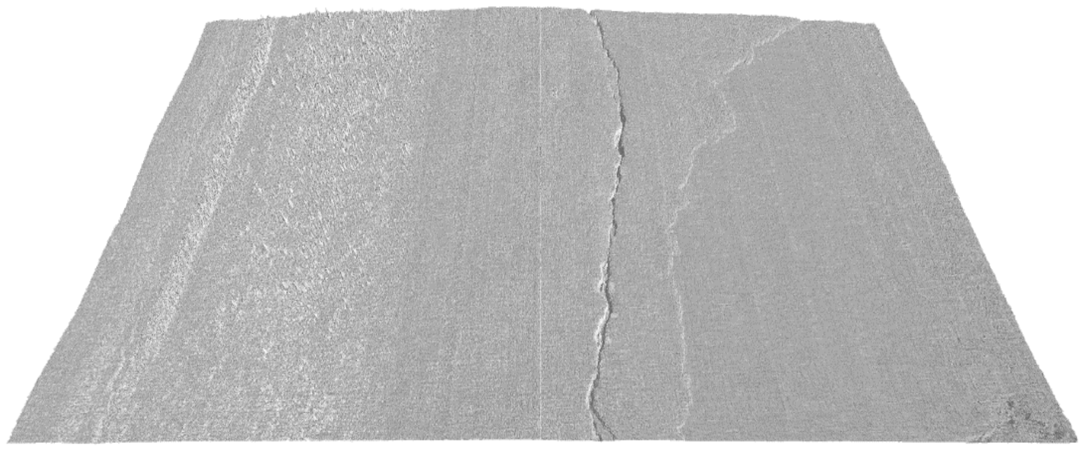

(b) Corresponding 3D Pavement Surface with Gaussian Lighting Model

Figure 3.5. Raw 3D Height Map & Corresponding Stereo Pavement Surface

3.3.2 Grid Structure

Pharr and Fernando (2005) proposed a simple grid structure (**Fig. 3.6**) to hide the boundaries between successive resolution levels and maintaining a watertight mesh structure. There are four key points for a nested grid structure: a) the grid size n = 2k – 1 (k is determined by programmer); b) grid square (white region) in the innermost of grid structure is made of the finest triangles; c) other hallow rings consist of triangles that is twice-coarser than those at adjacent high resolution level; and d) each hallow ring is broken into twelve square blocks (red regions), four ring fix-up blocks (green regions), one L-shape block (blue region) and one degeneration block (gold line) for graphic card memory saving and view culling. The degeneration block contains triangles that is perpendicular to the grid plane for gap-fill between adjacent resolution levels.

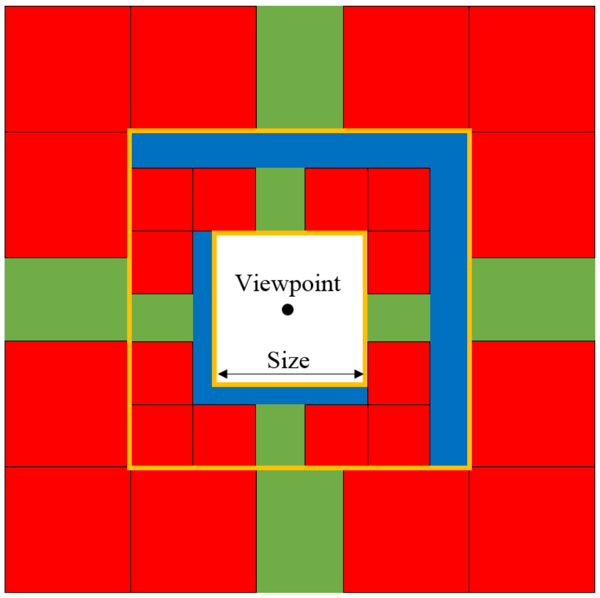


Figure 3.6. Simple grid structure (Pharr and Fernando 2005)

3.3.3 Alpha Blending Region

Alpha blending region is a geometric-blending or transition region that is vital for smooth transition and gap-fill between adjacent resolution levels. As **Fig. 3.7** shows, three height maps with the same size but different resolution levels are rescaled and rendered into the nest grid structure, in which alpha blending region is shown as the red zones. The height value "$\boldsymbol{H}$" of triangle vertices at arbitrary location $(\boldsymbol{x}, \boldsymbol{y})$ (i.e. vertex coordinate is $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{H})$) is co-determined by three parameters: the height value "$\boldsymbol{H_f}$" at $(\boldsymbol{x}, \boldsymbol{y})$ in the world coordinate, the height value "$\boldsymbol{H_c}$" at the same location $(\boldsymbol{x}, \boldsymbol{y})$ but in next-coarser level height map, and blending parameter "$\boldsymbol{\alpha}$" (**Eq. 3.1**). The blending parameter "$\boldsymbol{\alpha}$" is the maximum value of blending component "$\boldsymbol{\alpha_x}$" and "$\boldsymbol{\alpha_y}$" in both x-direction and y-direction (**Eq. 3.2**). "$\boldsymbol{\alpha_x}$" and "$\boldsymbol{\alpha_y}$" is derived from the linear interpolation shown in **Eq. 3.3** and **Eq. 3.4**, where $(\boldsymbol{x}, \boldsymbol{y})$ is arbitrary location coordinate of current height map, $(\boldsymbol{v_x}, \boldsymbol{v_x})$ is view point coordinate, "$\boldsymbol{n}$" is grid size, "$\boldsymbol{w}$" is the width of transition region (in this dissertation, "$\boldsymbol{w}$" is "$\boldsymbol{n}/\mathbf{10}$"), and "$\boldsymbol{clamp}$" is a OpenGL Shading Language (GLSL) built-in function to clamp a value between two given numbers (in this dissertation, the $\boldsymbol{\alpha_x}$ and $\boldsymbol{\alpha_y}$ are clamped between 0 and 1). The desired property of **Eq. 3.2**, **Eq. 3.3**, and **Eq. 3.4** is that "$\boldsymbol{\alpha}$" is started as 0 when the coordinate locates at the inner boundary, then "$\boldsymbol{\alpha}$" ramps up linearly in the transition region and finally reaches 1 at the outer boundary.

Fig. 3.8 shows the geometric difference between height maps without alpha blending (**Fig. 3.8(a)**) and with alpha blending (**Fig. 3.8(b)**). Based on **Fig. 3.8**, the geometrical gap can be completely removed by alpha blending so that different height maps can be smoothly stitched together.

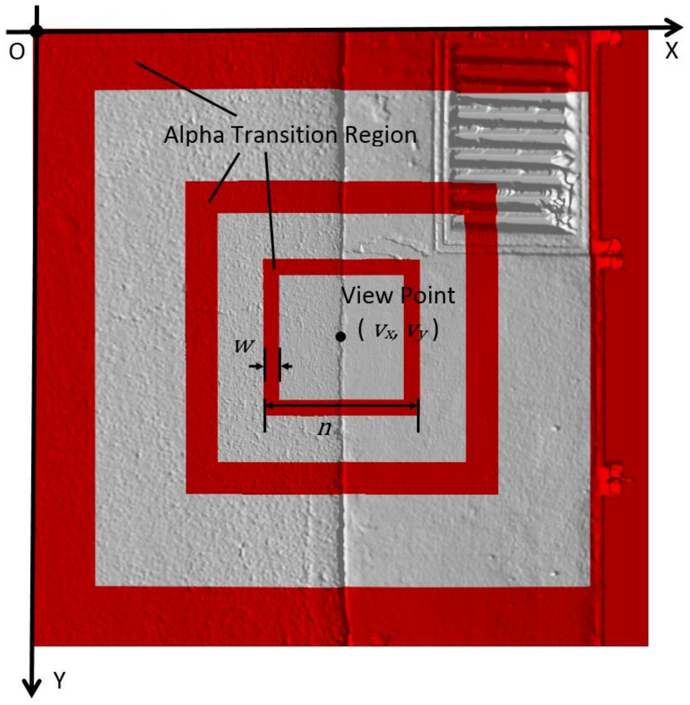


Figure 3.7. Alpha Blending Illustration

$$\boldsymbol{H = (1-\alpha) * H_f + \alpha * H_c} \qquad \textbf{Eq. 3.1}$$

$$\boldsymbol{\alpha = max(\alpha_x, \alpha_y)} \qquad \textbf{Eq. 3.2}$$

$$\boldsymbol{\alpha_x = clamp\left\{\left(x - v_x - (\frac{n-1}{2} - w - 1)\right), 0, 1\right\}} \qquad \textbf{Eq. 3.3}$$

$$\alpha_y = clamp\left\{\left(y - v_y - (\frac{n-1}{2} - w - 1)\right), 0, 1\right\} \qquad \textbf{Eq. 3.4}$$

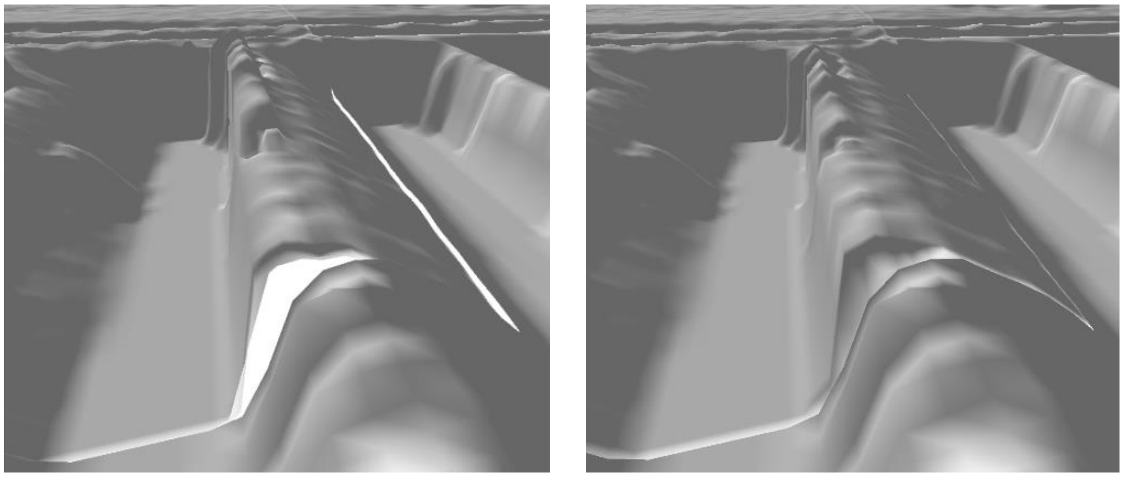

(a) Geometry before Alpha Blending **(b)** Geometry after Alpha Blending

Figure 3.8. Geometric Gap Elimination by Alpha Blending

3.3.4 View Frustum Culling

Since usually only a small portion of terrain is visible in computer screen at one time, view frustum culling is applied to determine which geometrical block in each hallow ring to be rendered so that the computer resources can be saved. Generally, the *Axis Aligned Bounding Box* (AABB) could be applied to implement view frustum culling in CPU (Assarsson and Moller, 2000; Brettell and Mukundan, 2005). This excessive and complex culling computation may consume significant amount of CPU resources for 3D rendering. In this dissertation, considering the height information on a long pavement is relatively uniform, 2D projection of view frustum is utilized to substitute for traditional 3D *AABB* method.

View frustum is a hexahedron that determines how 3D object inside it is projected from camera space to projection space (Microsoft Developer Network 2015). Each block in each hallow ring can be viewed as an independent 3D model. As **Fig. 3.9** shows, both view frustum and 3D model are projected onto the same plane. There are three conditions for the 2D projection:

- model projection is contained by view frustum projection
- model projection intersects with view frustum projection
- model projection is outside of view frustum projection

The 3D model is rendered for the first and second conditions, while the 3D block is not rendered for the third condition. **Fig. 3.10** shows the result of view frustum culling implemented by this method. The red region represents the actual visible portion based on the 2D projection of view frustum. Most of blocks in each ring have been culled to avoid redundant 3D rendering.

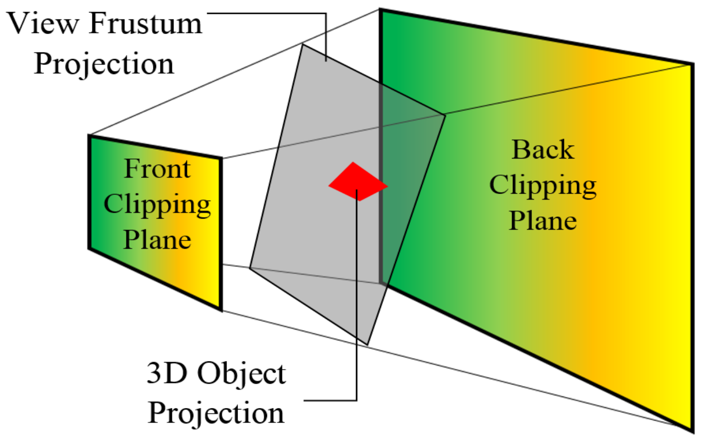


Figure 3.9. View Frustum Culling based on 2D Projection

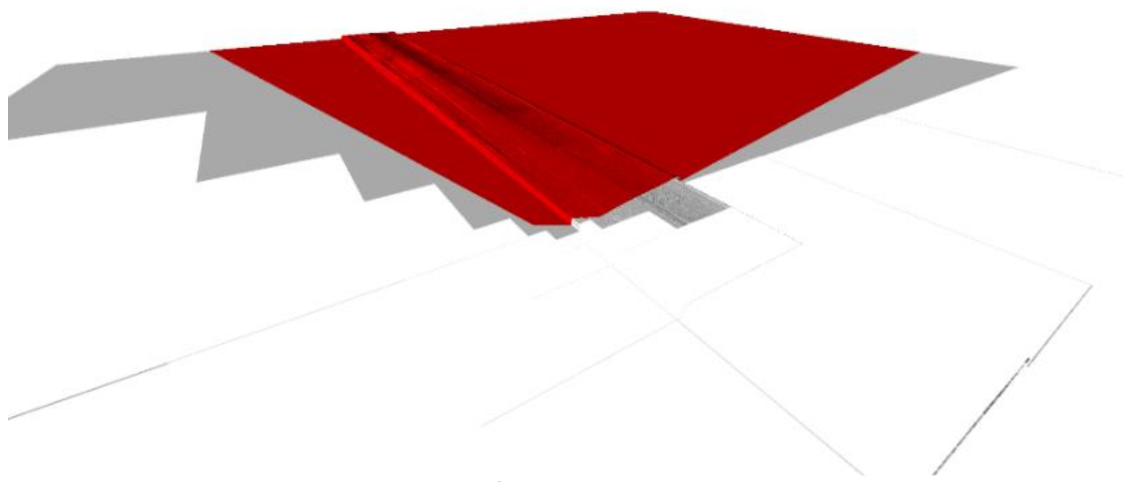
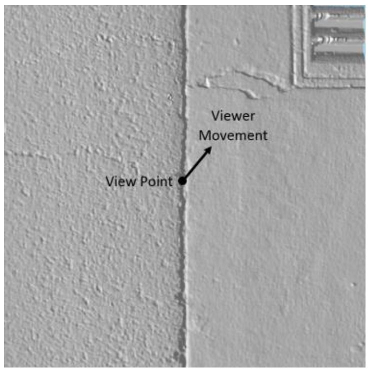


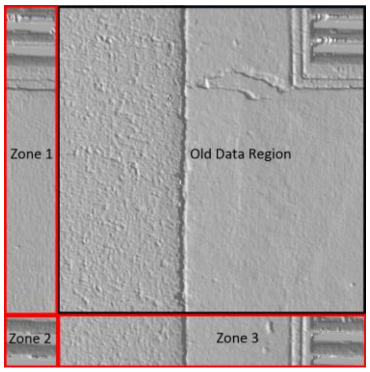
Figure 3.10. Results of Proposed 2D Projection Method

3.3.5 Data Update

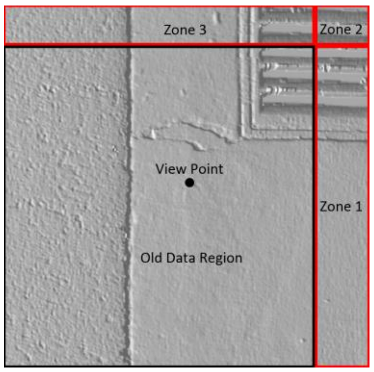
As the viewer wanders in the 3D scene, the surface textures should be updated dynamically to guarantee the viewer remained in the center of grid structure. For each elevation texture, it is unnecessary to update all the height data. Instead, only the data in two to maximum four (depending on the location of update region) quads need to be updated for each elevation texture, and the height data can be accessed toroidally for final 3D display (Losasso and Hoppe, 2004). To put it simply, the key idea is that the way of data storage in elevation texture is not the actual way for final display (**Fig. 3.11**). In case of **Fig. 11(a)**, viewer located in the center of terrain tends to move in a short length towards top-right direction. After viewer movement happens, only the data in zone 1, 2, and 3 is updated for current elevation texture, while the data in the old data region remains unchanged (**Fig. 11(b)**). Both new and old data of the elevation texture are retrieved using OpenGL shaders and repositioned for final display as shown in **Fig. 11(c)**.



(a) Move Towards Top-right Corner



(b) Save Updated Elevation Texture Data



(c) Display Updated Elevation Texture

Figure 3.11. 3D Texture Data Update Illustration with A Street Pavement Section

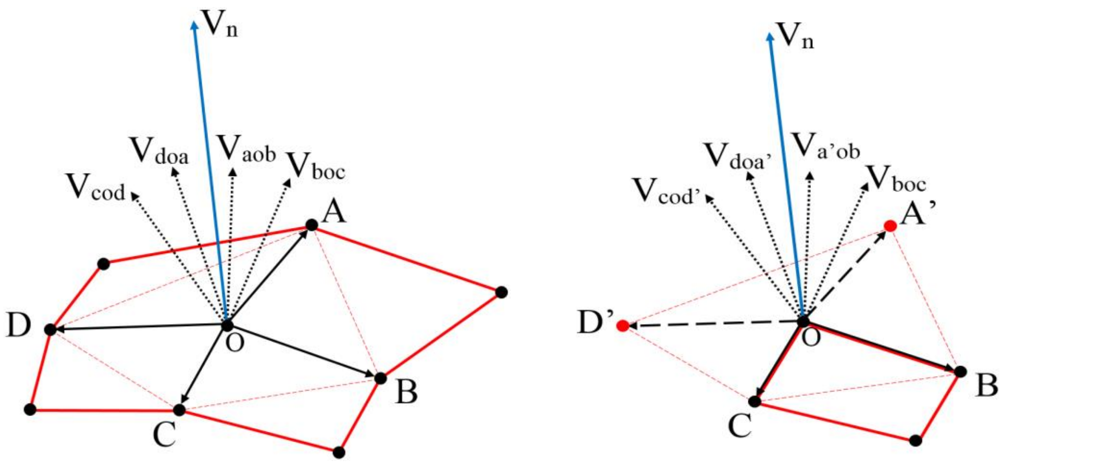
3.3.6 Vertex Normal Computation in GLSL Shader

Shaders are several mini-programs for manipulation of each vertex property (such as vertex position shifting, vertex color computation, texture color query, etc.). Shader programs are composed of specific shaders and executed simultaneously by separate threads of Graphics Processing Unit (GPU) with incredibly efficiency (Wolff, 2013). Generally, the normals of all triangle vertices are computed in CPU and sent to shaders for vertex color calculation based on the lighting model. In this dissertation, the vertex normals are directly computed in shaders based on texture color query to save both CPU resources and data transmission time between CPU and GPU.

Vertex normal is a critical component for smooth lighting effects on every triangle piece. In computer graphics, the vertex normal can be viewed as a normal of an underlying smooth surface approximated by the facets sharing that vertex, and estimated as a weighted sum of the normal of facets (Max, 1999). The coordinate's data for vertex normal computation can be obtained in OpenGL shaders based on commands of texture color query. The texture color represents the elevation data at specific location because geometry clipmap is implemented by the projection of elevation textures.

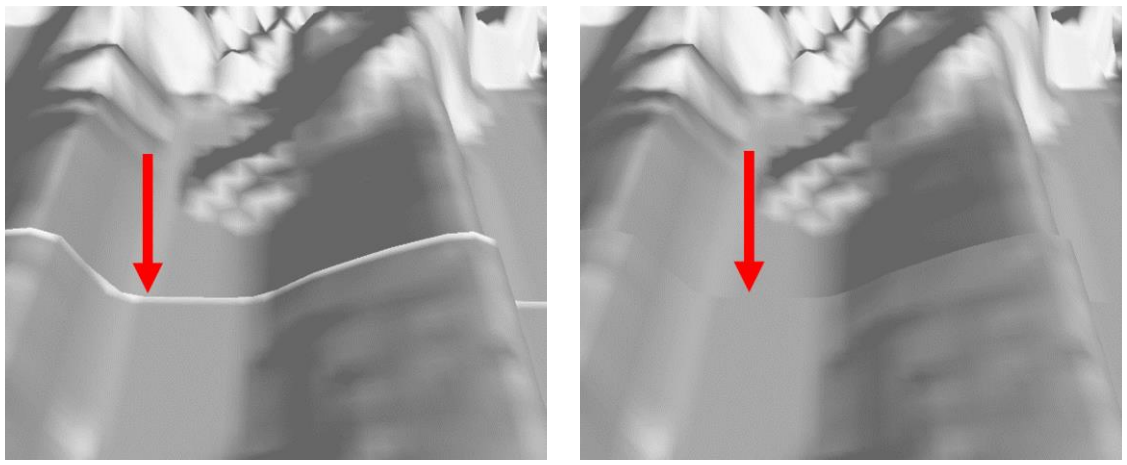
Fig. 12 shows how to calculate normal at a vertex *O*. In case of **Fig. 12(a)**, the vertex *O* which locates in the center of a geometry made by nine successive vertices (black points) is shared by four facets: *AOB*, *BOC*, *COD* and *DOA*. Therefore the normal at vertex *O* can be obtained as the normalized result of the sum of four surface: V_{aob}, V_{boc}, V_{cod}, and V_{doa}. If the vertex *O* is located in the boundary or corner of an elevation texture, the "fake vertices" are created during vertex normal computation for smooth lighting effects. As **Fig. 12(b)** shows, a corner vertex *O* is located in a corner of a geometry made by four adjacent vertices (black points) and possessed by only one surface *BOC*. The two fake vertexes *A'* and *D'* (red points) are created in the OpenGL shader using the same height value of vertex *O* but different location value. Assuming the coordinate of vertex *O* is (X_O, Y_O, H_O). The fake vertex *A'* coordinate is (X_O - 1, Y_O, H_O), and *D'* is (X_O, Y_O - 1, H_O) which make three fake surfaces *A'OB*, *COD'* and *D'OA'* for vertex normal computation.

Fig. 13 shows the different lighting effects caused by the "fake vertices". In **Fig. 13(a)**, since no "fake vertices" are created during normal computation for vertex locating at the boundary or corner of an elevation texture, abrupt lighting effects are observed between two adjacent elevation textures. In **Fig. 13(b)**, the lighting effects are much smoother by a large degree because "fake vertices" contributes to normal computation for the boundary and corner vertices in the elevation texture.



(a) normal calculation when vertex is located inside elevation texture **(b)** normal calculation when vertex is located in the corner of elevation texture

Figure 3.12. Calculation of Normal at a Vertex "*O*"



(a) lighting effects without "fake vertices" **(b)** lighting effects with "fake vertices"

Figure 3.13. Lighting Effects based on "Fake Vertices"

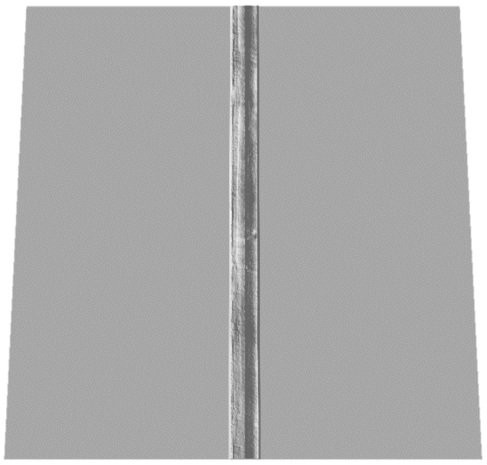
3.4 Experimental Results

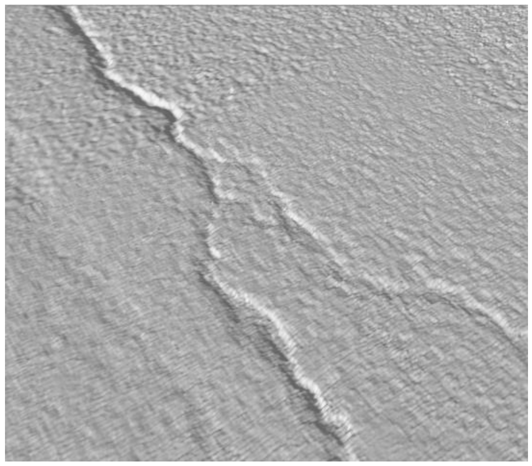
3.4.1 3D Visualization Results

The experimental virtual 3D long pavement surface visualization is based on 32 consecutive 1-mm resolution 3D raw images, around 65 meter length and 4 meter width with a total of 270 million pixels. **Table 1** lists the hardware configuration and software tools applied for developing and executing program. Based on geometry clipmap, the virtual 3D long pavement surface is rendered as a relative stable frame per seconds (FPS) for visual evaluation of pavement condition. **Fig. 14** provides screenshots of visualization of the long pavement surface based on geometry clipmap. This algorithm provides a bird-view of the pavement surface when viewer is far away from pavement (**Fig. 14(a)**), and a detailed view of the 1-mm resolution pavement surface when viewer is close to pavement (**Fig. 14(b)**). **Fig.14(c)** provides the visualization of the pavement surface while the viewer roams on the surface.

Table 3.1. Development Environment for 3D Long Pavement Visualization

Development Environment	Name	Description
CPU	i7-3930K	Main frequency: 3.2GHz
Video Card	GTX 770	Nvidia Kepler GK104, 4GB Graphic Card Memory
Software Tool	VS2013 MFC	Microsoft Foundation Classes (MFC) is applied for future software development
Graphics API	OpenGL	A open standard 3D graphic API
Language	GLSL	A flexible language to program with OpenGL


(a) long pavement surface bird view


(b) 1-mm resolution crack on surface

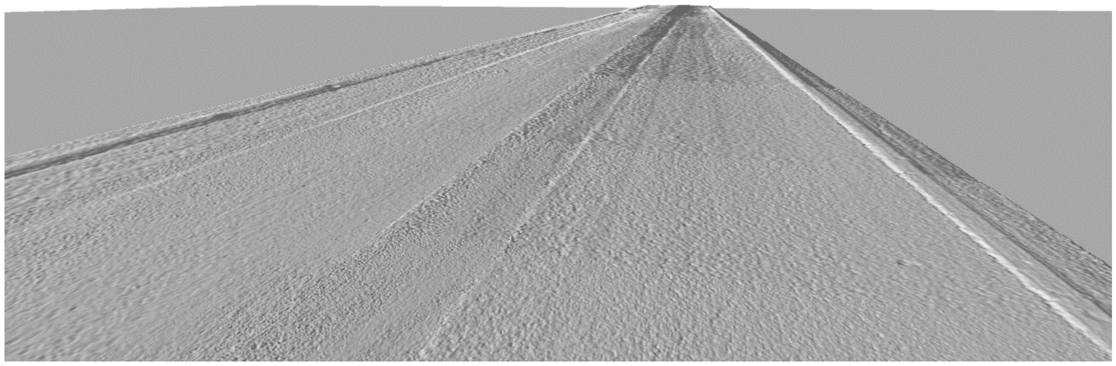

(c) walking on long pavement surface

Figure 3.14. Visual Results for 3D Long Pavement Surface

3.4.2 Statistics of Frames per Second (FPS)

The experimental pavement surface dataset is around 270 million pixels (i.e. 2048 * 4096 * 32). To evaluate the performance of geometry clipmap, the FPS of 3D display in 1 minute is summarized as **Fig. 3.15**. The same pavement is rendered in three different configurations: (a) grid size of 2047 with 6 layers of textures, (b) grid size of 1023 with 7 layers of textures, and (c) grid size of 511 with 8 layers of textures. Configuration (b) achieves the higher FPS, which is more stable than configurations (a) and (c) who have approximately identical FPS. The results seem to be logical since the number of rendered triangles grows as square when grid size doubles. On the other hand, even though gird size shrinks, the increase of rendered texture levels causes lower FPS. Therefore, the highest FPS is balanced between grid size and the number of texture levels.

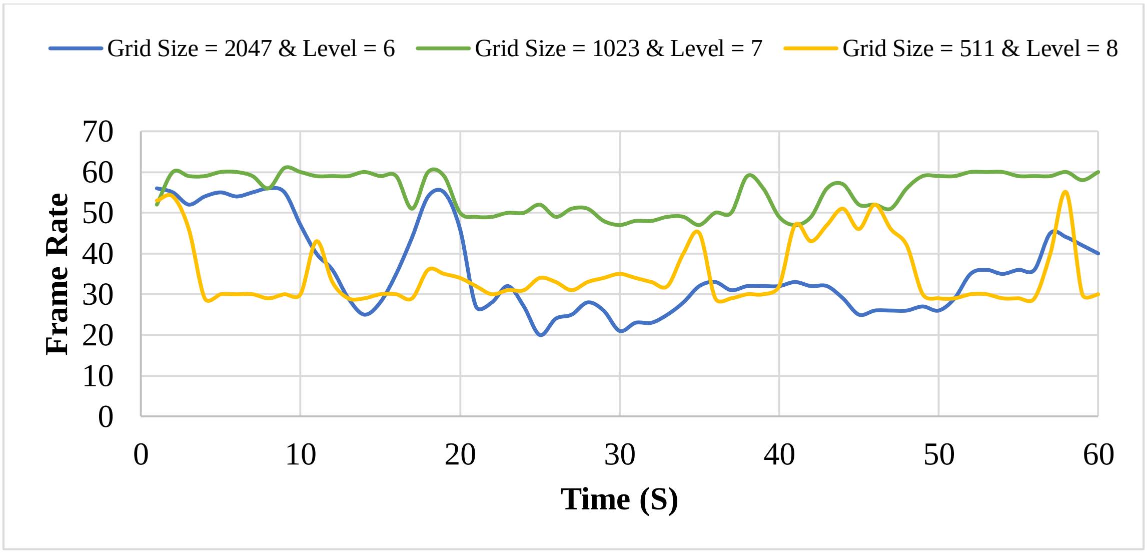
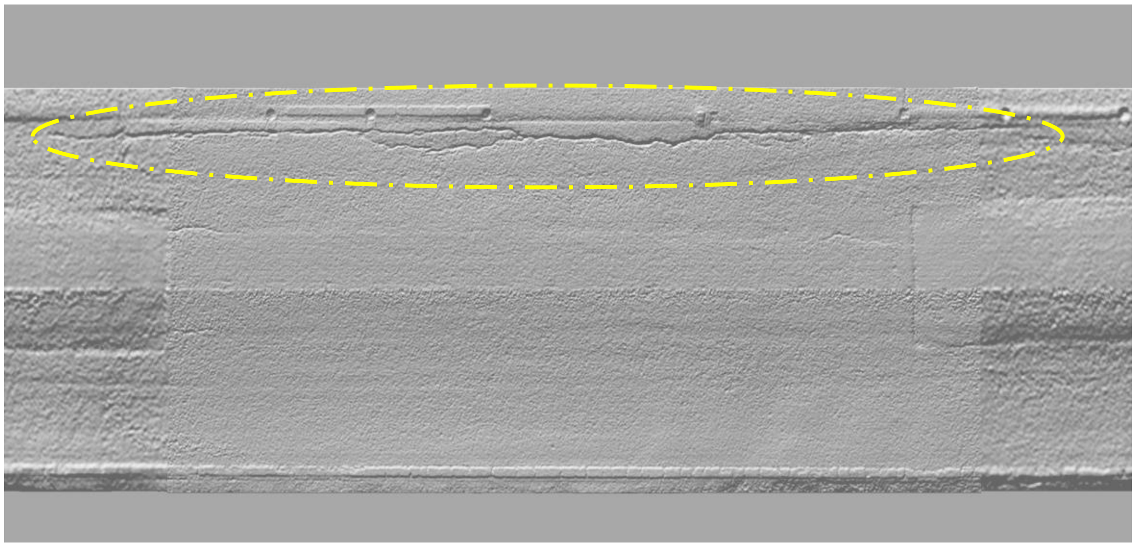


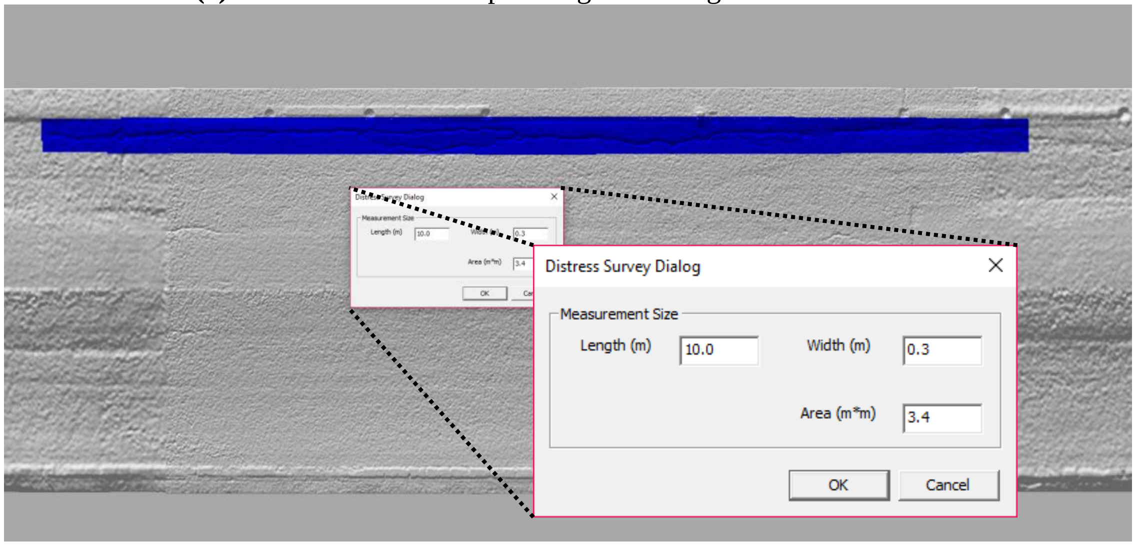
Figure 3.15. Frame Rate of the Same Datasets Rendered by Different Configurations

3.4.3 Distress Measurement on Long Pavement

The benefits of large pavement 3D visualization are not only to provide an overall condition of the pavement for visual inspector, but also to offer the actual field environment helpful to pavement distress size measurement. For instance, there is a long crack distributed in multiple consecutive images that is easier to be observed on the long pavement surface (**Fig. 3.16(a)**). Through large pavement 3D visualization and proper computer programming, the size of long crack can be directly measured in the software as **Fig. 3.16b** shows. In **Fig. 3.16b**, software users can easily use mouse to draw a blue box (i.e. the blue box that covers the whole crack) on stereo pavement surface and get the crack information for the crack. In other words, the pavement surface distresses (e.g. thick asphalt patching, wheel path cracking, etc.) can also be direct measured with aid of large-scale pavement visualization, even if their imaging data are at different consecutive images during automated data collection. On the contrary, if the stereo pavement is visualized only based on single image as traditional method does, the whole long crack will be broken into incomplete cracks on consecutive stereo images, which causes inconvenience to measure the entire crack when need arises.



(a) A Crack across Multiple Images on Long Pavement Surface



Distress Survey Dialog

Measurement Size

Length (m) 10.0

Width (m) 0.3

Area (m*m) 3.4

(b) Direct Size Measurement of the Long Crack in OpenGL Graphic API

Figure 3.16. Distress Size Measurement on Long Pavement Surface

3.5 Summary

A robust 3D pavement model is a necessary component to support pavement condition survey. However, limited research has directed toward long pavement visualization in 3D environment. In this dissertation, the Geometry Clipmap is first-time utilized to display the virtual long-distance pavement surfaces using 1-mm 3D laser imaging data, a LOD application to long-distance pavement 3D visualization. The results of the experiment demonstrates that all the 1-mm details of long virtual pavement are retained for the comprehensive pavement condition survey while it also provides the bird's eye view for the overall pavement condition evaluation. Besides, both visual continuity and frame rate keep uniform during rendering at run-time. In addition, pavement surface defects distributed across multiple images can be measured in a simulated realistic environment with a software tool specifically developed in the dissertation.

Two technical concepts are proposed and their implementations in the research make geometry clipmap algorithm useful to practical pavement visualization. First, the 2D projection of view frustum is presented to replace traditional AABB method for removing geometrical blocks that are not need to be rendered, so that the time required for 3D rendering can be saved. Further, the vertex normal computing based on fake vertexes is utilized to take full advantage of graphic card resources so that the data size required by the 3D rendering can be further decreased for better computational efficiency.

For future work, 3D virtual long pavement should be rendered in a specific mesh grid structure instead of the square structure to further decrease the GPU resource consumption. More work is also needed to implement a more efficient algorithm to address the long computing time required for preparation of full pyramid data structure.

Chapter 4 A Deep Learning System for Cell-based Automated Crack Detection

4.1 Introduction to Convolutional Neural Network (CNN)

The term “Deep Learning” increasingly attracted public attentions since Google computer program named “AlphaGo” defeated South Korea Go Master in earlier 2016. In fact, deep learning has had a theoretical prototype for a long time. Historically, deep learning has many different terms in history: it was known as “Cybernetics” in the 1940s-1960s, as “Connectionism” or “Artificial Neural Networks” (ANNs) in the 1980s-1990s, and finally renamed as “Deep Learning” after 2006. All the changes involved substantial increase in complexity and functionality. Through self-modification of network internal adjustable weights, deep learning is able to identify objects in images, transcribe speech into text, make intelligent decisions, and realize human-machine interaction, etc. Nowadays, numerous amazing unmanned applications based on deep learning have emerged, such as the driverless vehicles, unmanned aerial vehicles, and medical robots. What’s even more exciting is that deep learning has assisted NASA in the discovery of two new planets in a far-away solar system (i.e. Kepler-90 Planetary System), which may be an important step in future space colonization or extraterrestrial life exploration. (Hinton et al., 2006; Hagan et al., 2014; LeCun et al., 2015; Goodfellow et al., 2016; Sahllue and Vanderburg, 2017).

Deep Convolutional Neural Network (CNN) is a representative model of deep learning that achieves remarkable successes in recent years. CNN are designed to process data that come in the form of multiple arrays, for example a color image composed of three 2D arrays containing pixel intensities in the three color channels. The layer-wise structure of CNN is inspired by the natural visual perception mechanism of the living creatures. The weights of each CNN layer are responsible for the connections inside the model, such that input features can be abstracted or extracted hierarchically for meaningful outputs. Different from traditional algorithms with hand-designed input feature extractors, CNN has a capacity to automatically learn good features from the input datasets by adjusting weights based on general-purpose training procedure. Besides, the quality of features learnt in CNN is positively proportional to the amount of both weights and available training data, which is a critical characteristic benefitting for big data processing. (LeCun et al., 1998a; Krizhevsky et al., 2012; LeCun et al., 2015; Goodfellow et al., 2016).

A primary application of CNN is in the field of 2D image recognition though it can also involve 1D or 3D data processing like signals or videos. The *ImageNet Large Scale Visual Recognition Challenge* (ILSVRC) is a yearly competition to evaluate algorithms for object detection and classification on large-scale image datasets. In ImageNet datasets, over 1.2 million images labeled in 1,000 different categories are used for image detection, classification and localization. Most of the well-known CNNs trained by ILSVRC data wins the ILSVRC competition with outstanding performance. As early as 2012, *AlexNet* demonstrated the powerful capability of deep CNN on large-scale image recognition. In the next few years, various deep CNNs (e.g. *VGG*, *GoogleNet*, *ResNet*, etc.) had been proposed to improve image recognition ability by continuously increasing the network depth (**Fig. 4.1**). It was noteworthy that in 2015, *ResNet* with 152-layer reached as low as 3.57% of top-5 error at ILSVRC, which was already more excellent than human-level performance on image recognition (5.1%). (Krizhevsky et al., 2012; Russakovsky et al., 2014; Simonyan and Zisserman, 2014; Szegedy et al., 2014; He et al. 2015; PaddlePaddle, 2018).

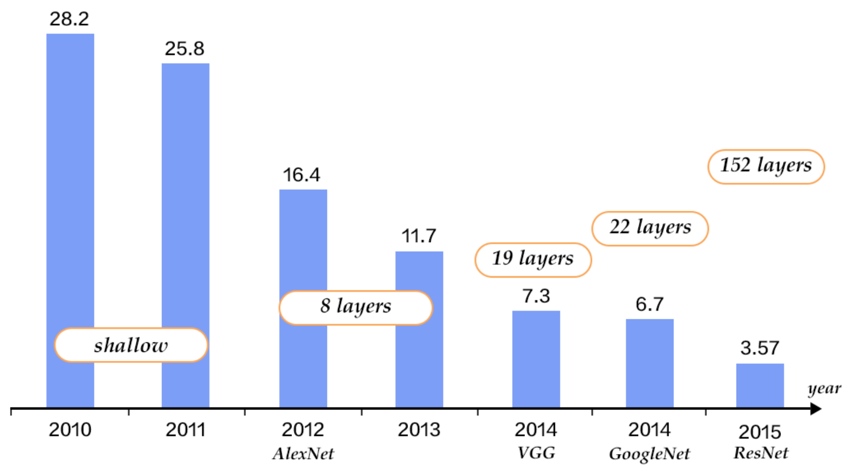


Figure 4.1. ImageNet Classification Top-5 Error (%) (PaddlePaddle, 2018)

4.2 CNN Work Principle

It is important for pavement engineers to understand the work principle of CNN so that the powerful CNN capacity on image classification can be applied on automated pavement surface crack detection. A basic CNN model (**Fig. 4.2**) can be regarded as a variant of the Multi-layer Perceptron (MLP) based feedforward network (**Fig. 4.3**). The whole architecture is mainly composed of multi-layer cascaded convolutional layers, pooling layers (i.e. subsampling layers), and fully connected layers. Based on internal connections of CNN structure, digital image can be hierarchically abstracted as a series of feature maps and propagated layer by layer starting from the input node all the way to the output node. Through reversely propagating the errors computing at output node (i.e. the Back-Propagation), all the parameters in CNN can be self-adjusted based on Stochastic Gradient Descent (SGD). As a result, the outputs of CNN can be gradually approximated to the expected targets after repeatedly weights update. (Rosenblatt, 1958;

Minsky, 1969; Rumelhart et al., 1986; LeCun et al., 1990; Kreinovich, 1991; LeCun et al., 1998a; LeCun et al., 1998b; Hagan et al., 2014; Goodfellow et al., 2016).

There are three critical concepts in CNN for image pattern recognition (LeCun et al., 1998a; Krizhevsky et al., 2012): 1) local receptive field, 2) shared weights, and 3) pooling layers. Firstly, inspired by the neural structure in cat's visual system, the local receptive field is utilized to extract locally-sensitive, orientational-selective low-level features that can be combined together for high-level abstract features. The second idea is the shared weights: the parameters in each filter are shared to detect the same features at all possible locations in the inputs, and then to generate feature maps so that units in this map were connected by the same parameters. Thirdly, the down-sampling method is applied to decrease the spatial resolution of the feature map while to reduce the output sensitivity to object shifts and distortions. With these key notions, the CNN structure largely lowers the number of weights in the network while still ensures the accuracy to small image processing tasks.

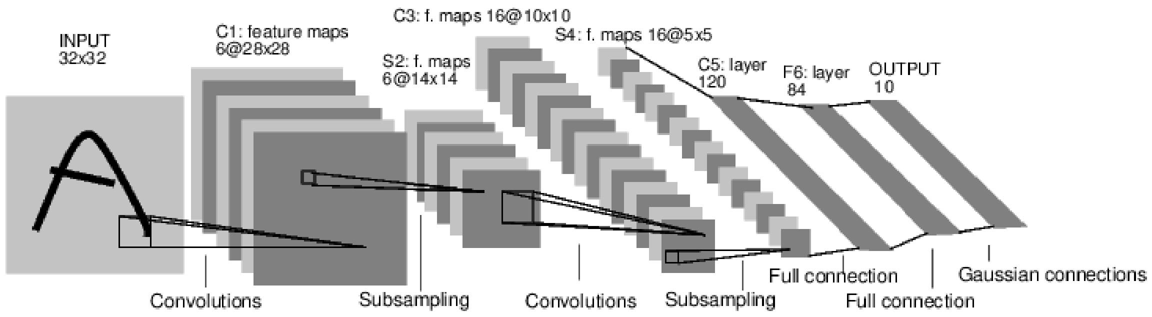
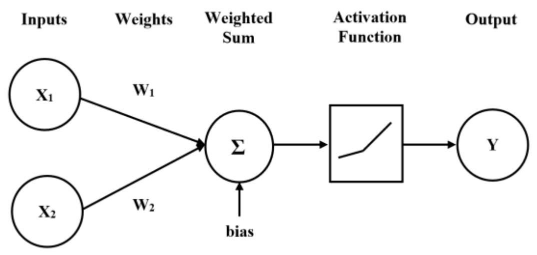
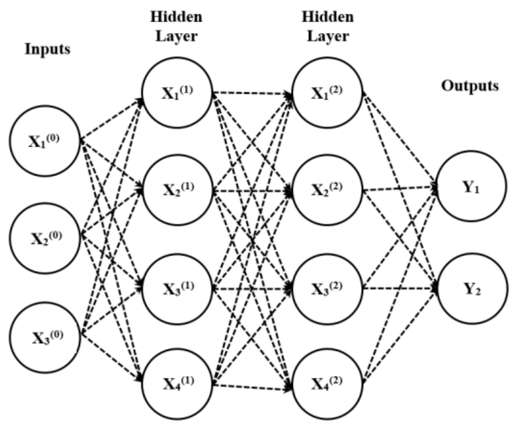


Figure 4.2. LeNet-5 Network Structure (LeCun et al., 1998a)



(a) A Perceptron Connects Inputs by Weights and Bias



(b) Multi-layer Perceptron for Pattern Recognition

Figure 4.3. Feedforward Network based on Multi-layer Perceptron (MLP)

4.3 Cell-based Automated Crack Detection

4.3.1 Network Architecture

The CNN architecture for crack detection on pavement image cells is shown in **Fig. 4.4**. As the CNN structure is designed to classify grid-like images, the network inputs are pavement image cells in size of 64×64 came from standard image in size of 1024×512. In other words, each standard image has 128 (i.e. 16×8) image cells that can be imported in the network. The network output is a number ranged in [-1.7159, 1.7159]. If the output is larger than 0, then a single input image cell is considered having cracks; otherwise it is considered having no cracks. This network architecture consists of three major components: two convolutional layers, two max-pooling layers and three fully-connected layers (including the output layer).

In the convolutional layers, feature maps are extracted from the inputs in each layer by different filters in each layer. The first convolutional layer *C1* is connected by 16 filters, while the second convolutional layer *C3* is bridged by 32 filters. The filter sizes for both convolutional layers are all set to 9×9 for feature maps generation. After convolutional operations, the feature map size is reshaped as following equation: $F_{size} = I_{size} - Ftr_{size} + 1$, where F_{size} is the size of feature map,

I_{size} is the size of input image before convolution, and Ftr_{size} is the size of filters that convolute the input images. Therefore, the image sizes in *C1* and *C3* layer are 56×56 and 20×20 respectively.

In the pooling layers, the size of feature maps is down-sized to remain shift- and scale-invariant features while reduce the number of weights to train. Max-pooling is applied to keep features in a pool window with size of 2×2. In other words, the maximum value of a 2×2 image block is the value of an individual pixel on the down-sized image. The stride of the pool window is set as 2, which means there is no overlap area when pool window sampling on the input images. After max-pooling operation, the image sizes in *S2* and *S4* are 28×28 and 10×10 respectively.

Finally, Fully-connected layers (i.e. *F5*, *F6*, and *Output Layer*) are used to construct the Multi-layer Perceptron for crack recognition on image cells. As mentioned in **Fig. 4.3** before, the MLP in CNN grabs feature maps in *S4* as inputs, and then propagates features in forward direction to generate the output value. The number of neurons are set to 64 and 96 for *F5* and *F6* layer respectively. What needs to notice is that all the pixels in *S4* feature maps are connected to corresponding neural node, which means *S4* and *F5* layers are bridged by a series of filters in size of 10×10. At the *Output Layer*, convolutions using 1×1 filters are applied to merge results from *F6* layer and then to generate a predicted value.

As shown in **Table 4.1**, the CNN for Cell-based crack detection has roughly 250,000 parameters need to be trained. The amount of parameters is much less than it in those large networks with millions of parameters such as *AlexNet* or *GoogleNet*. However, the number of parameters should be adequate for the 2-objects classification on image cells (i.e. having crack or non-crack). Besides, redundant parameters for only 2-objects classification can result in greater possibility of errors instead of performance improvement. In addition, training a network with massive parameters requires more extra labeled image datasets, which are expensive and laborious to obtain.

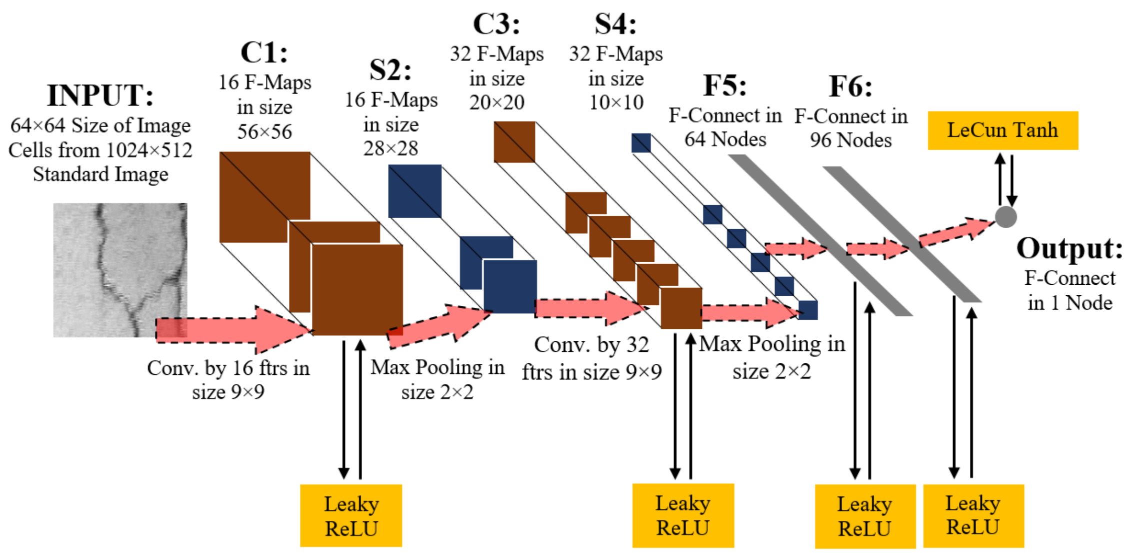


Figure 4.4. CNN Architecture for Crack Detection on Image Cells

Table 4.1. Trainable Weights and Biases in Each CNN Layer

Layer	#inputs	#Nodes	Ftr Size	Weights	Biases	Total
C1	1	16	9×9	1296	16	1,312
C3	16	32	9×9	41472	32	41,504
F5	32	64	10×10	204,800	64	204,864
F6	64	96	1×1	6,144	96	6,240
Output	96	1	1×1	96	1	97
Total	-	-	-	253,808	209	254,017

4.3.2 Convolutional Layers

Convolution is a process of modifying the spatial frequency characteristics of the images using the given filter kernels. For instance, there is a classic filter type named “Gabor” for image edge detection (Marčelja, 1980; Daugman, 1985). Simply speaking, as **Fig. 4.5** shows, Gabor filters simulate human visual system to concentrate on looking more details in the center (i.e. white color) while ignoring some information in the surroundings (i.e. black color). Also, Gabor filters possess characteristics at different angles. The convolution process is to compute a weighted sum of each element from the image based on a given filter as **Fig. 4.6** shows. Therefore, as **Fig. 4.7** shows, the edge features of different angles on an image can be extracted after getting the maximum response of the convolutional results between a given image and a set of Gabor filters.

Similarly in the CNN for Cell-based crack detection, the convolutional process attempts to extract meaningful features from the inputs at each convolutional layer. In the meanwhile, the noises or non-relevant information of the inputs can be suppressed or eliminated during convolutional process. Thus, the useful information can be propagated in layer-wised form to the output layer so that the final object classification can be conducted correctly. However, different from traditional convolutions based on hand-designed feature extractor like Gabor filters, the weights of each filter kernel in the network can be self-learnt with given inputs and corresponding targets, which gets rid of the limitations in pre-designed or fixed feature extractors. In other words, the convolutional layers in the network can be more flexible to extract the optimal features for image classification, hence the name "Convolutional Neural Network".

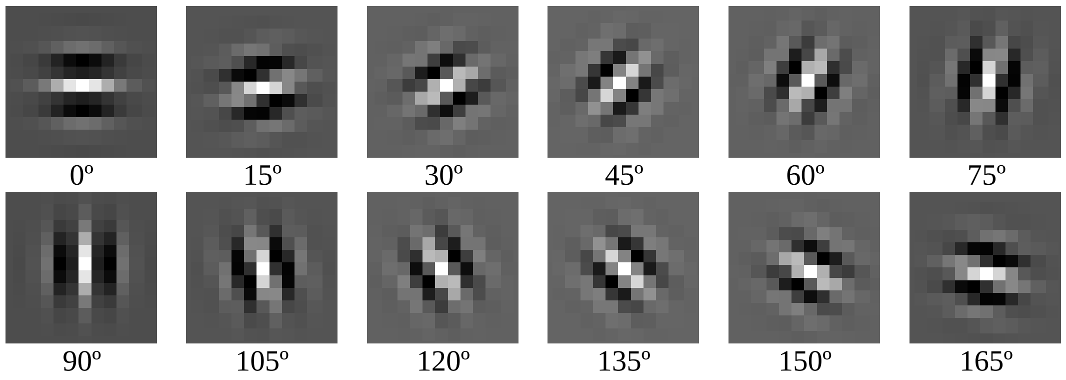


Figure 4.5. Gabor Filters at Different Angles

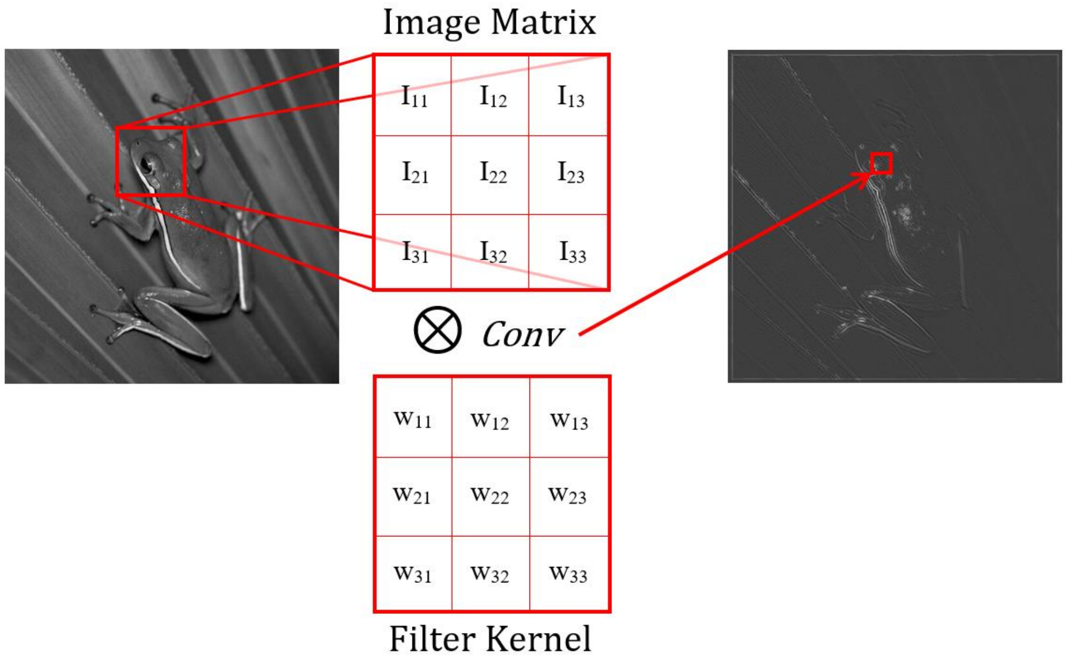
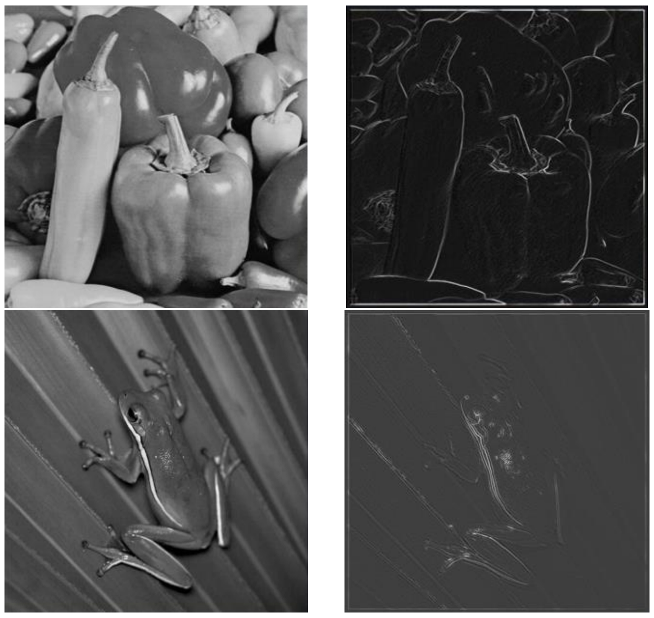


Image Matrix
I_{11} I_{12} I_{13}
I_{21} I_{22} I_{23}
I_{31} I_{32} I_{33}
$\otimes$ Conv
w_{11} w_{12} w_{13}
w_{21} w_{22} w_{23}
w_{31} w_{32} w_{33}
Filter Kernel

Figure 4.6. Convolutional Process between Image and Filter



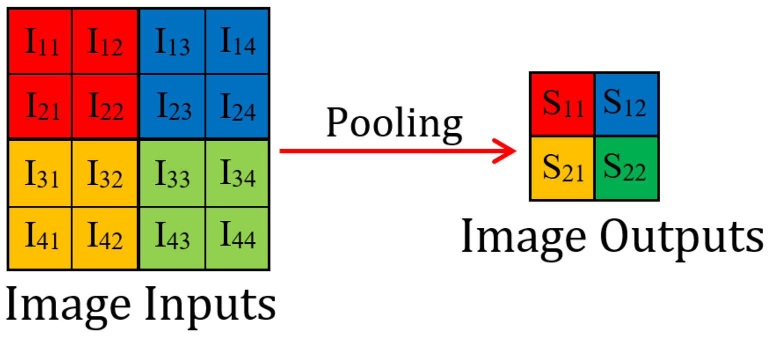
(a) Original Images **(b)** Edge Detection Results

Figure 4.7. Edge Detection Results based on Gabor Filters

4.3.3 Pooling Layers

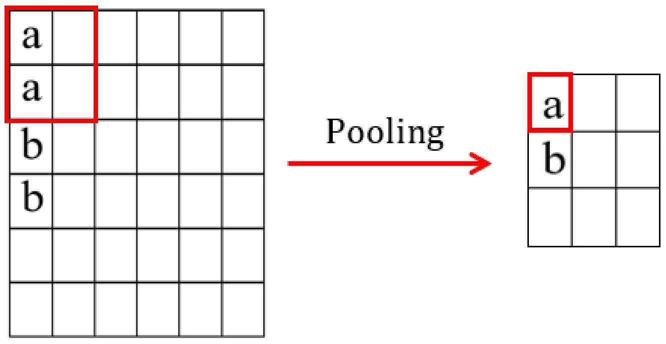
Max pooling method is adopted to remain the shift-, distortion- and scale-invariant features on input images. The pooling process is a down-sampling process. For example, **Fig. 4.8** shows a pooling process using a 2×2 window to sample inputs with stride of 2. Based on max pooling method, $S_{11} = max(I_{11}, I_{22}, I_{33}, I_{44})$, which means S_{11} is the maximum response in the 2×2 red area. As a result, S_{11} is possible to keep invariant feature even if the objects on image inputs have a little position shift or scale change, etc. (**Fig. 4.9**).

Another advantage of pooling layer is to reduce the number of weights that requires to be trained. Assume there are no pooling layers in the CNN structure as **Fig.4.4** shown before, the size of each feature map before *F5* layer is 48×48, which means 2,304 weights required to connect the single feature map. On the other hand, with two pooling layers *S2* and *S4*, the size of single feature map before *F5* layer is 10×10, which means only 100 weights are needed. In other words, the decrease in number of weights caused by pooling layers is able to reduce the network model complexity, such that the overfitting phenomenon in neural network can be prevented.

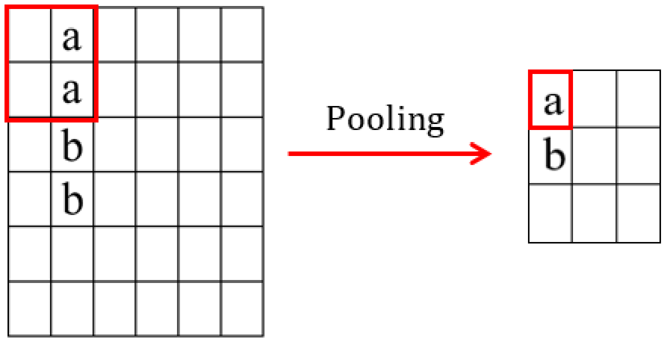


I_{11} I_{12} I_{13} I_{14}
I_{21} I_{22} I_{23} I_{24}
I_{31} I_{32} I_{33} I_{34}
I_{41} I_{42} I_{43} I_{44}
Image Inputs
Pooling
S_{11} S_{12}
S_{21} S_{22}
Image Outputs

Figure 4.8. Pooling Process Using 2×2 Window to Down-Sample on Image Input



(a) Max Pooling Results before Image Object Translation



(b) Max Pooling Results after Image Object Translation

Figure 4.9. How the Max Pooling Remain Shift-Invariant Features

4.3.4. Weights Initialization

Effective weights initialization is extremely important to a network training since poor initialization may result in limited network performance or even unmanageable behaviors during network training. There are many weights initialization techniques such as *Gaussian Initialization* (Krizhevsky et al., 2012), *Xavier Initialization* (Glorot and Bengio, 2010), *He Initialization* (He et al., 2015), etc. In this dissertation, *Xavier Initialization* is applied to initialize all the parameters of CNN model.

Simply speaking, Xavier technique keeps the data variance in the reasonable range during both forward- and backward-propagation, so that the network can be trained easily. Due to *Xavier* technique requires inputs in a “zero mean” range, the pixel values on image cells before imported in the network will subtract their own mean values. Based on this technique, the weight of a neuron in a certain layer should be drawn from a uniform distribution as **Eq. 4.1** shows below:

$$\boldsymbol{W} \sim [-\sqrt{\frac{\mathbf{6}}{\boldsymbol{n_i}+\boldsymbol{n_j}}}, \sqrt{\frac{\mathbf{6}}{\boldsymbol{n_i}+\boldsymbol{n_j}}}] \qquad \textbf{Eq. 4.1}$$

where $\boldsymbol{n_i}$ is the number of input connections and $\boldsymbol{n_j}$ is the number of output connections.

Besides, it is worth noticing that the weights of filters in convolutional layers are initialized based on variant of **Eq. 4.1**. For a $w \times h$ filter connecting between n_i feature maps from the previous layer and n_j feature maps in the current layer, the weights in this filter are uniform distribution as **Eq. 4.2** shows below:

$$\boldsymbol{W_{ftr}} \sim [-\sqrt{\frac{\mathbf{6}}{\boldsymbol{w}\times\boldsymbol{h}\times\boldsymbol{n_i}+\boldsymbol{n_j}}}, \sqrt{\frac{\mathbf{6}}{\boldsymbol{w}\times\boldsymbol{h}\times\boldsymbol{n_i}+\boldsymbol{n_j}}}] \qquad \textbf{Eq. 4.2}$$

4.3.5 Activation Functions

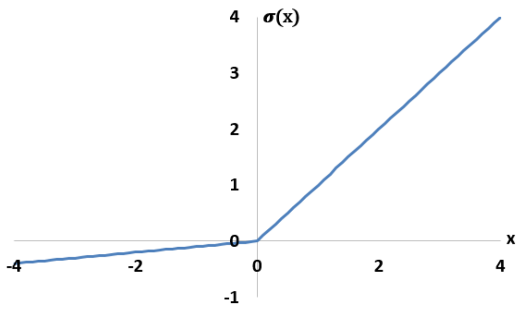
In neural network, the activation function of a node defines the output of that node given an input or set of inputs as **Fig. 4.3(a)** shows before (Hagan et al., 2014). When the activation function is non-linear, then a two-layer neural network can be proven to be a universal function approximator (Cybenko, 2006). Therefore, non-linear activation units are inevitable for CNN structure to approximate general functions for problem solutions.

In this dissertation, all except the output layer applies a general activation unit Leaky Rectifier Linear Unit (Leaky ReLU) (Maas et al., 2013). The Leaky ReLU is defined as **Eq. 4.3** (**Fig. 4.10(a)**). Mathematically, Leaky ReLU directly transfer the inputs $\boldsymbol{x}$ to next layer if this unit is active, otherwise the inputs $\boldsymbol{x}$ are suppressed in small and non-zero signals controlled by *alpha* parameter and then sent to next layer. Based on Leaky ReLU, the sparse representations for image object can be easily obtained in CNN, which is helpful to enhance the network classification capacity.

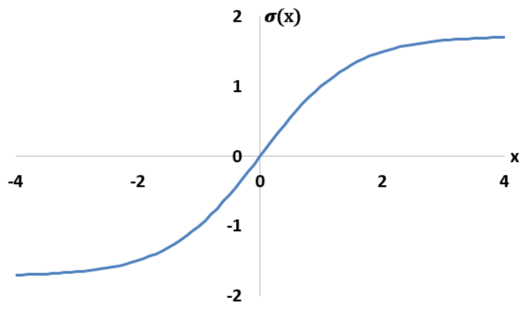
A sigmoid-like unit is applied at output layer for a 2-objects classification problem. The recommended activation unit at output neuron is the LeCun Tanh (LeCun et al., 1998b).
The LeCun Tanh is defined in **Eq. 4.4** (**Fig. 4.10(b)**). Compared with the traditional sigmoid unit, the LeCun Tanh has several advantages (**Fig. 4.11**). First of all, the range of LeCun Tanh is larger than that of traditional sigmoid unit, which means the learnable parameters have more space to be adjusted by Back-Propagation algorithm. Secondly, LeCun Tanh has both positive and negative parts in y-axis, which is very suitable for a typical binary classification problem. Lastly, the first order derivative of LeCun Tanh is larger and more effective than that of sigmoid unit, which means the gradients can be better back-propagated inside the network.

$$\boldsymbol{\sigma(x)} = \begin{cases} \boldsymbol{x} & \boldsymbol{if\ x > 0} \\ \boldsymbol{alpha \times x} & \boldsymbol{otherwise} \end{cases} \qquad \textbf{Eq. 4.3}$$

$$\boldsymbol{\sigma(x)} = \boldsymbol{1.7159 \times tanh(\frac{2}{3}x)} \qquad \textbf{Eq. 4.4}$$

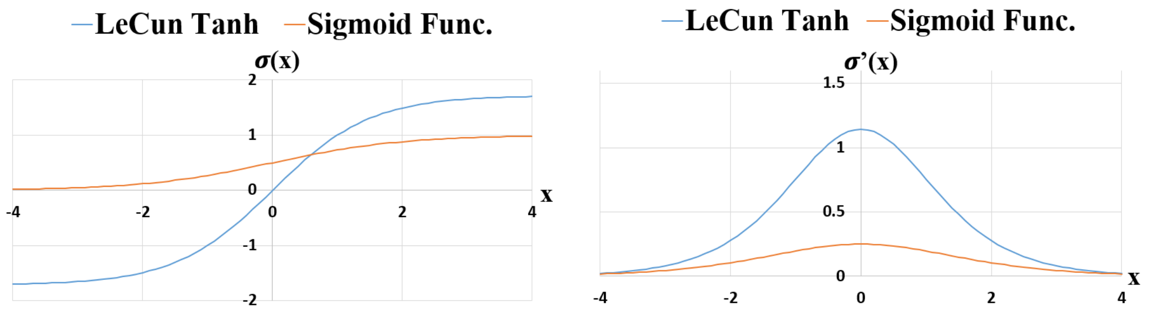


(a) Leaky ReLU



(b) LeCun Tanh

Figure 4.10. Activation Units in Cell-based CNN for Crack Detection



(a) Activated Units Comparison **(b)** 1st Order Derivatives Comparison

Figure 4.11. Comparison between the LeCun Tanh Unit and Traditional Sigmoid Unit

4.3.6 Learning Rule

The network is trained through supervised learning. In supervised learning, the training datasets consist of input examples $\boldsymbol{x}$ and corresponding targets $\boldsymbol{t}$ that participate network training together. Assume the network outputs by given inputs $\boldsymbol{x}$ are $\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x})$. The goal of network training is to adjust weights, so that the outputs $\boldsymbol{y}$ by given inputs $\boldsymbol{x}$ can gradually approximate the targets $\boldsymbol{t}$.

In supervised learning, the cost function is set at the output layer of a neural network to measure the errors between network outputs $\boldsymbol{y}$ to targets $\boldsymbol{t}$. There are two basic properties for a cost function: 1) cost function should be non-negative; and 2) the cost function is close to zero when the network outputs $\boldsymbol{y}$ are close to the expected targets $\boldsymbol{t}$. This dissertation adopts the cost function in the form of cross-entropy combined with L_2 weight decay. The cross-entropy is helpful to speed up training as the errors between predicted and target values are directly back-propagated (Solla et al., 1988). On the other hand, L2 weight decay prevents network overfitting by penalizing large weights in the network (Krogh and Hertz, 1992). The cost function can be expressed as:

$$\boldsymbol{C} = \mathbf{0.75} \times [(\boldsymbol{t} - \mathbf{1.7159}) \times \boldsymbol{Ln}(\mathbf{1.7159} - \boldsymbol{z}) - (\boldsymbol{t} + \mathbf{1.7159}) \times \boldsymbol{Ln}(\mathbf{1.7159} + \boldsymbol{z})] + \mathbf{3.4318} + \boldsymbol{L_2}\ \boldsymbol{Item} \qquad \textbf{Eq. 4.5}$$

where $\boldsymbol{t}$ is the target value of a pixel; $\boldsymbol{z}$ is the predicted value of a pixel; $\boldsymbol{C}$ is the evaluated error of a pixel; $\boldsymbol{L_2\ Item}$ equals to $\mathbf{0.5} \times \boldsymbol{\lambda} \times \sum \boldsymbol{W}^{\mathbf{2}}$ ($\boldsymbol{\lambda}$ is the weight decay parameter, and $\boldsymbol{W}$ represents all learnable weights in the network).

The internal weights update is based on Back-Propagation (BP) combining with Stochastic Gradient Descent (SGD). BP is a chain-rule based process of reversely propagating cost function errors after completion of each network forward propagation. Subsequently, SGD is utilized to compute gradients for weights update so that the cost function errors can be minimized. In this dissertation, momentum based SGD is adopted to compute gradients, where the momentum is a common method for fast training. Besides, Mini-batch SGD is implemented to update weights accurately at each iteration. Thus, the weights update with momentum γ during training at each iteration can be expressed as:

$$\begin{cases} \Delta \boldsymbol{W}_{\boldsymbol{i+1}} = \gamma * \Delta \boldsymbol{W}_{\boldsymbol{i}} - (\mathbf{1} - \gamma) * \boldsymbol{\varepsilon} * \left(\dfrac{\partial \boldsymbol{C}}{\partial \boldsymbol{w}} | \boldsymbol{w}_{\boldsymbol{i}} \right)_{\boldsymbol{B}_{\boldsymbol{i}}} \\ \boldsymbol{w}_{\boldsymbol{i+1}} = \boldsymbol{w}_{\boldsymbol{i}} + \Delta \boldsymbol{W}_{\boldsymbol{i+1}} \end{cases} \qquad \textbf{Eq. 4.6}$$

where $\boldsymbol{i}$ is the iteration index; $\boldsymbol{w}_{\boldsymbol{i}}$ is the weights updated at ith iteration; $\Delta \boldsymbol{W}_{\boldsymbol{i}}$is the momentum variable at $\boldsymbol{i}^{\text{th}}$ iteration; γ is the momentum parameter (usually be set as 0.9); $\boldsymbol{\varepsilon}$ is the learning rate; $\left(\frac{\partial \boldsymbol{C}}{\partial \boldsymbol{w}} | \boldsymbol{w}_{\boldsymbol{i}} \right)_{\boldsymbol{B}_{\boldsymbol{i}}}$ is the average gradient over the batch $\boldsymbol{B}_{\boldsymbol{i}}$ evaluated at $\boldsymbol{w}_{\boldsymbol{i}}$.

4.4 Experimental Results

4.4.1 Standard Pavement Image Library

For machine learning purposes, the research team built an image library that consists of more than 6,000 3D pavement images collected by WayLink PaveVision3D system and corresponding ground-truth images with hand-labeled pavement cracks. All 3D pavement images in the image library were collected by the PaveVision3D system in last 5 years on different pavements. Each image covers an area of 4-meter (width) by 2-meter (length). The established image library represents diversified variations of cracks and pavement surface textures. There is no overlap between any two images, and no more than 100 images are from the same pavement section. The ground-truths of cracks on all images are manually processed with close supervisions on pixel-perfect accuracies by multiple teams. A three-round inspection is conducted to ensure the ground-truths are accurate at pixel level. For the first round, several well-trained operators manually mark the cracks on provided 3D pavement images with full resolution. For the second round, several other well-trained operators examine and refine the ground-truths for correcting errors and reducing subjectivity. Finally, the ground-truths are further inspected and verified by experts. The entire process of preparing ground-truths of cracks is completed on continuing basis for more than one year. Each image then is downsized in a standard image size from 4096×2048 to 1024×512 using the min-pooling method. In other words, the minimal value of a 4×4 data block becomes the value of an individual pixel on the downsized image. **Fig. 4.12** shows several examples of 1-mm 3D pavement images with corresponding ground truths. All the data used in the remaining chapters in this dissertation are came from this image library.

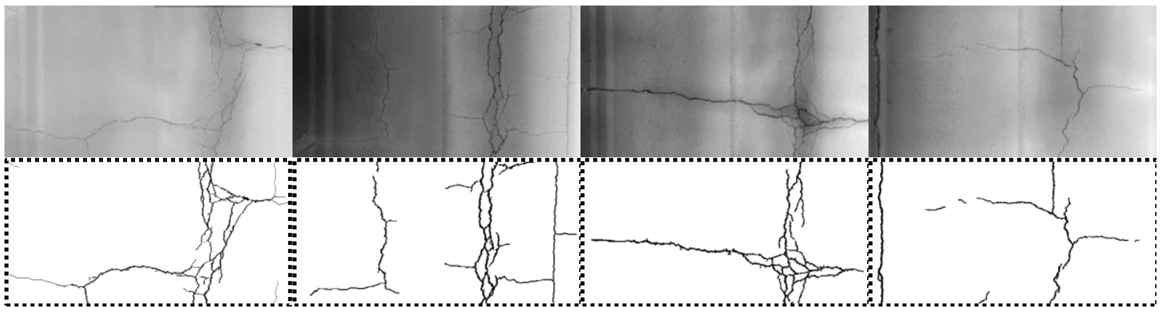


Figure 4.12. Examples from Image Library and Corresponding Manually Labeled Ground-Truths

4.4.2 Evaluation Method

Due to the fact that the output is only a 2-objects classifier to determine if a single image cell has cracks or not, the index "accuracy" is not proper to measure the performance of the classification results. Instead, the performance indicators precision ($\boldsymbol{Pr}$), Recall ($\boldsymbol{Re}$), and F1 score ($\boldsymbol{F1}$) are utilized to evaluate the network performance during both training and testing process (Fawcett et al., 2006; Zhang et al., 2016a). The precision and recall can be computed on true positive ($\boldsymbol{TP}$), false negative ($\boldsymbol{FN}$) and false positive ($\boldsymbol{FP}$) as following equations:

$$\boldsymbol{Pr = TP/(TP + FP)} \qquad \textbf{Eq. 4.7}$$

$$\boldsymbol{Re = TP/(TP + FN)} \qquad \textbf{Eq. 4.8}$$

$$\boldsymbol{F1 = 2 \times P \times R/(P + R)} \qquad \textbf{Eq. 4.9}$$

where $\boldsymbol{TP}$ is a crack recognized correctly on an image cell; $\boldsymbol{FP}$ is a non-crack recognized incorrectly on an image cell; and $\boldsymbol{FN}$ is a crack recognized incorrectly on an image cell.

4.4.3 Network Training

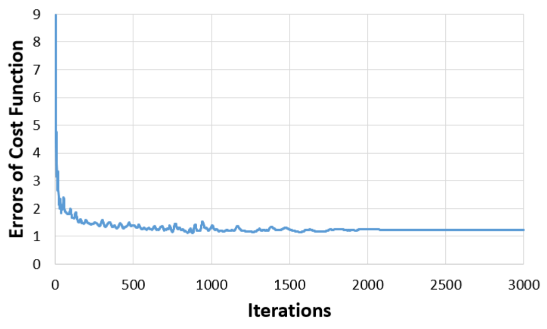
Roughly 2,500 asphalt pavement images from the standard image library with corresponding manually labeled target images are used as training set to train the crack detection network. The image cell size is of 64×64, which means each standard image in size of 1024×512 has 128 image cells in totally. If an image cell contains cracks, it will be assigned with binary value "1.7159", otherwise it will be assigned with binary value "-1.7159".

10 asphalt images are selected as validation set. "Validation set" is a subset from the entire dataset used to estimate the network performance during the training process. In the training process, the estimation on the validation set contributes to network configuration settings (e.g. learning rate, number of batch size, etc.) and best-performing weights selection. Both the errors of

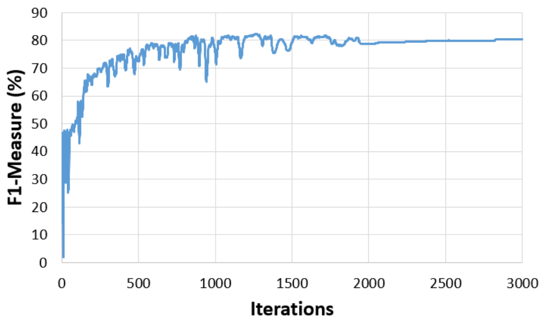
cost function and F-1 Measure are monitored in the training phase. In addition, 10 asphalt images are considered to be adequate for the binary classification task. There is no overlap between training set and validation set.

The network training is based on the BP and SGD algorithm. The mini-batch size is 32 at the beginning and then gradually increases to 128 as training continues. The momentum variable is set as 0.9, which is found to be very helpful for stabilizing network training. L_2 weight decay is 0.0005 during entire training process. All layers share the same learning rate. The learning rate is set as 0.01 at the beginning, and then divided by 2 (i.e. 0.005) after 200 iterations training or divided by 3 (i.e. 0.0033) after 500 iterations training. Finally, the network is fine-tuned with learning rate 0.0025 after 1,000 iterations and rate 0.001 after 2,000 iterations to seek for further optimization. No Drop-Out technique is applied in the network. The network training with 3,000 iterations is completed in roughly one and half day with the aid of a single GPU device (NVidia GeForce GTX 1080Ti).

Fig. 4.13 shows the performance on validation set in the training phase. In **Fig. 4.13(a)**, the errors of cost function decrease as training goes on, which indicates that the network attempts to minimize the difference between network outputs and excepted targets. The error decrease is very significant on the first 1,000 iterations. After 1,000 iterations, the error tends to become stable even though the learning rate is set at a small value. In **Fig. 4.13(b)**, the F1-Score increases as training continues. Similarly, F-1 score increases obviously at the first 1,000 iterations and gradually tends to become stable from 1,000 to 3,000 iterations. Therefore, the training for crack detection on image cells are stopped at the 3,000th iteration.



(a) Errors of Cost Func. vs. Iterations



(b) F1-Measure (%) vs. Iterations

Figure 4.13. Network Performance on Validation Set

4.4.4 Performance on Test Data

The weights configuration with the best performance is used for the final test data. The test set consist of 200 asphalt images that do not participate the training process. The F1-Score on 200 asphalt images is 70.45% (with precision 60.67% and recall 83.99%). **Fig. 4.14** shows several examples of Cell-based CNN performance on the test set. The blue box with crossed lines means there is no cracks in a single image cell, otherwise it means an image cell having cracks. Generally speaking, the crack detection results based on CNN is reasonable because most cracks on image cells can be recognized correctly.

On the other hand, the Cell-based CNN crack detection may not meet the practical requirements of pavement crack detection. For example, the practical crack detection is able to provide two important indices: the crack width (i.e. severity) and crack length. However, those two indices are unable to be obtained based on Cell-based results because the CNN structure is specifically designed for “grid-like” image object recognition. Even though continuous improvements can be achieved, the Cell-based accuracy does not catch up with the accuracy in the practical pavement crack detection. In other words, there is a need to modify CNN structure so that it can be improved to a higher accuracy level in detection. Therefore, the goal next chapter is to achieve the pixel-level crack detection while improve the network performance.

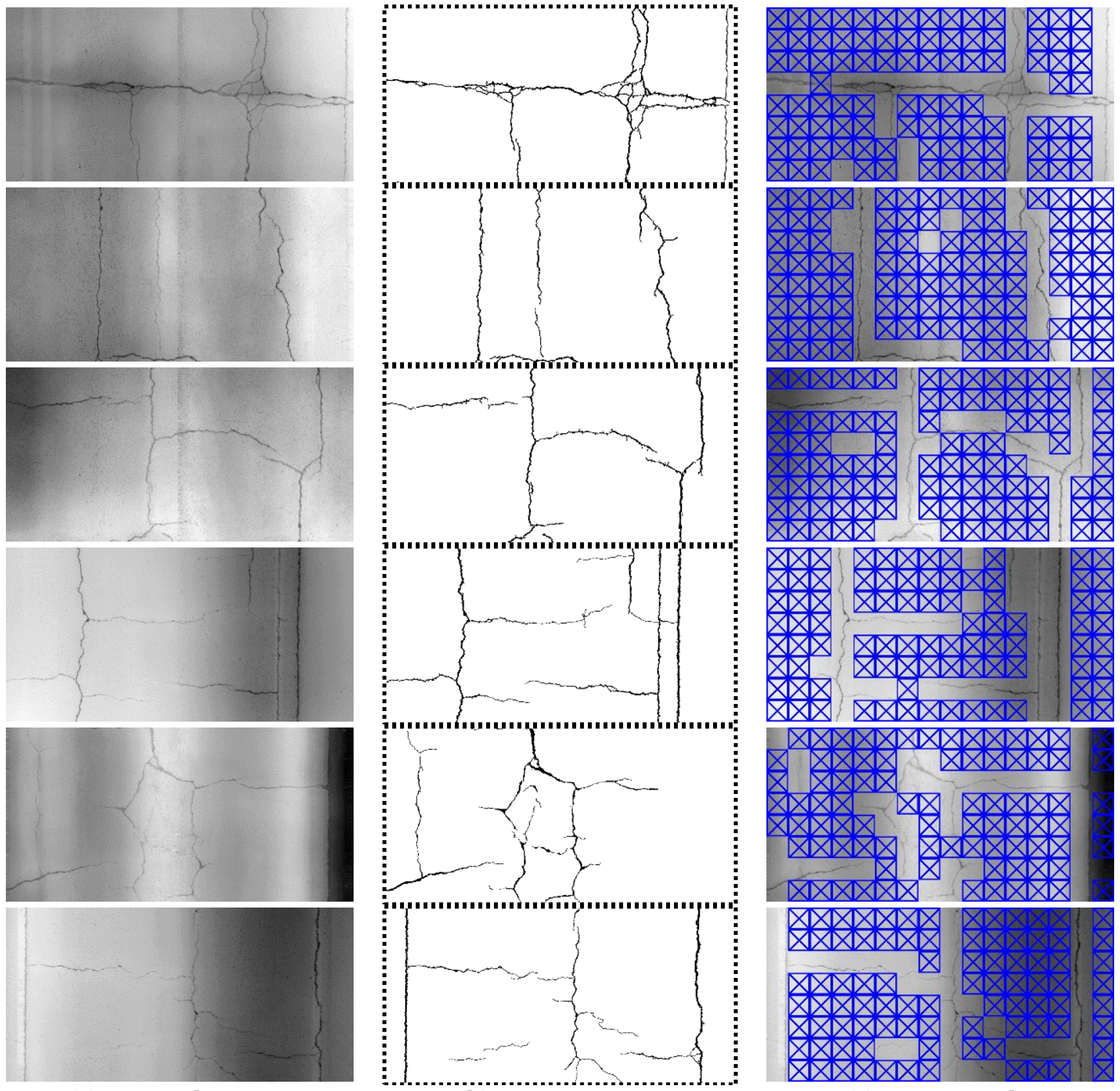


Figure 4.14. Results of Cell-based CNN Crack Detection

4.5 Summary

Because of the excellent performance of Convolutional Neural Network (CNN) on image object recognition, it was natural to investigate a CNN application on a typical and most interesting development: crack detection. This dissertation section adopts CNN algorithm for Cell-based crack detection on 1-mm 3D pavement laser images. The data source is the standard image library that consists of over 6,000 pavement images in total and corresponding target images manually labeled by multiple research teams. Through 3,000 iterations of weights update, the Cell-based crack detection performance achieves the F1-Score 70.45% (with precision 60.67% and recall 83.99%). The crack recognition results are found to be reasonable, which demonstrates the potentials of deep learning on crack detection task.

However, the accuracy of CNN structure is at Grid-Level, which is not able to provide for the detailed crack indices such as crack width and length. Thus, it is necessary to develop a deep-learning based network at a higher accuracy level such as at pixel-level. This objective and its development is detailed in the next chapter to explore the CNN modification to achieve the pixel-level crack detection.

Chapter 5 A Deep Learning Methodology for Pixel-Level Detection of Pavement Cracks

5.1 Difficulties in Pixel-Level CNN

Chapter 4 demonstrates the feasibility and performance of Convolutional Neural Network (CNN) for pavement crack detection. On the one hand, CNN was revealed to have the potentials to make reasonable predictions of surface cracks based on huge amount of 1-mm pavement data. On the other hand, however, CNN framework itself is specifically designed for image cell recognition, which impedes the determination of actual crack positions and the further crack classification. To implement the pixel-level accuracy on crack detection, several problems of present CNN framework need to be solved as summarized below:

- Pavement Unevenness

  The elevations of realistic pavement surfaces are more or less up-and-down instead of being totally flat as **Fig. 5.1(a)** shows. This type of pavement unevenness negatively impacts the feature extraction process based on the concept of "shared weights" by using traditional CNN frameworks, resulting in limited performance of pixel-level crack detection. Assume a CNN network is able to approximate the function $\boldsymbol{z = \sigma(x)}$, where $\boldsymbol{x}$ and $\boldsymbol{z}$ are the inputs and outputs respectively, function $\boldsymbol{\sigma}$ can be regarded as the shared weights in CNN. Due to the significant height difference $\boldsymbol{h_d}$ between $\boldsymbol{h_1}$ and $\boldsymbol{h_2}$ as **Fig. 5.1(b)** shows, $\boldsymbol{(h_1)}$ is not equal to $\boldsymbol{\sigma(h_2)}$ (i.e. $\boldsymbol{\sigma(h_1) \neq \sigma(h_{1+}\, h_d)}$). As a result, cracking pixels at different surface elevations cannot have consistent outputs using the same set of shared weights in the network.

- Pooling Layers

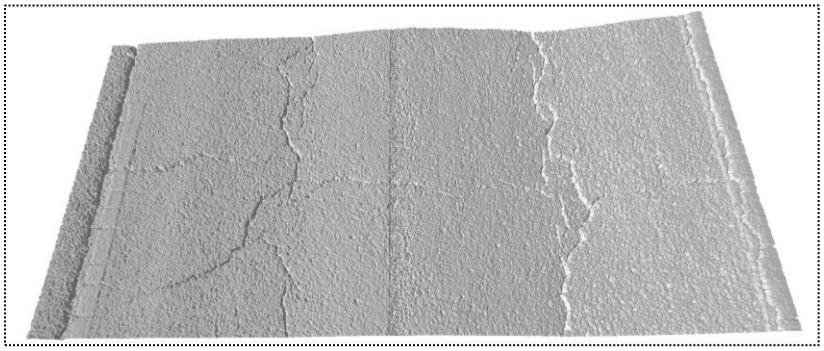
  As discussed in Chapter 4, pooling layers in cell-based CNN are critical for cell-based image recognition by keeping shift-, distortion- and scale-invariant features while reducing the number of trainable parameters as well as preventing overfitting in the network. In contrast to cell-based CNN model, pixel-level crack detection aims at determining a specific pixel is crack or not. In this case, the advantages of pooling layers for cell-based detection, such as shift-invariant feature extraction, can contrarily impact the accuracy of pixel-level detection.

- Error Regularization

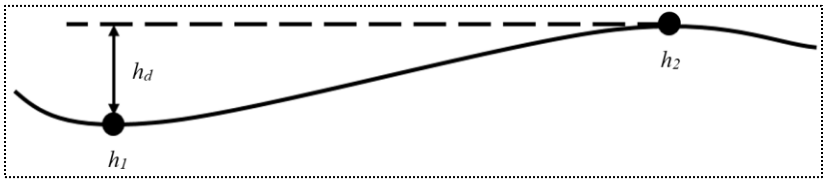
  In cell-based CNN model, each image block has only one error factor (i.e. the difference between learning target and network output) that needs to be propagated backward at each iteration. By contrast, pixel-level CNN model predicts the cracking probability for every pixel in the input image simultaneously, which means there are over one hundred thousands of error factors at the output layer. Accumulation of such amount of errors during back-propagation can easily cause gradient explosion, which leads to limited network performance or even a termination in network training. In other words, error factors in pixel-level CNN model can not directly participate in training process.

- Time-Efficiency

  After the optimization using General-Purpose Graphics Processing Unit (GPGPU), the cell-based CNN model in Chapter 4 spends roughly 2.4 second generating results for all 128 image cells on a standard image. If the cell-based CNN structure is directly applied to check each pixel in an image, it would consume much longer time to calculate results for huge amounts of pixels from the input. In other words, considering there are gigabytes to terabytes of pavement imaging data for miles of roads, direct application of cell-based CNN on the pixel-level detection is unacceptable in time efficiency due to extremely long time computational needs with commonly available computer hardware.



(a) Pavement Data under 3D Visualization



(b) Pavement Side Profile Illustration

Figure 5.1. Unevenness in Pavement Surfaces

5.2 CrackNet-V for Pixel-Level Detection

To achieve the pixel-level detection accuracy on pavement cracks, it is necessary to find an effective framework to overcome problems mentioned above. In the meanwhile, the network performance as well as the speed efficiency are required to be enhanced for big pavement data processing. Therefore, the CNN architecture named "CrackNet-V" based on concept of "shared weights at each pixel" is designed and developed in the dissertation to complete the pixel-level crack detection with needed accuracy and efficiency.

5.2.1 Network Architecture

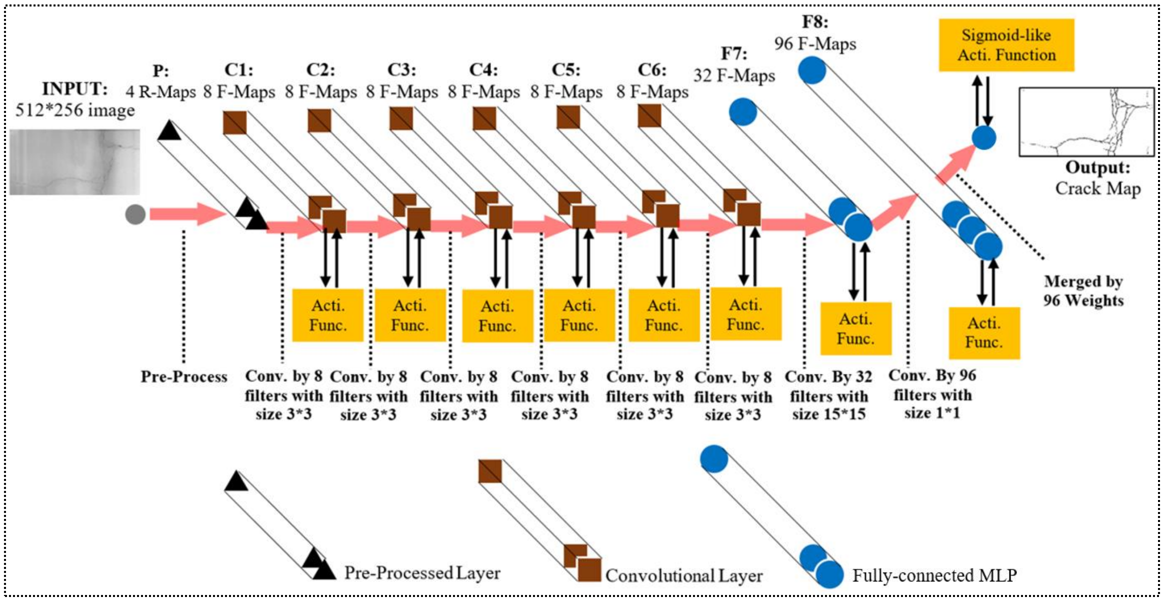
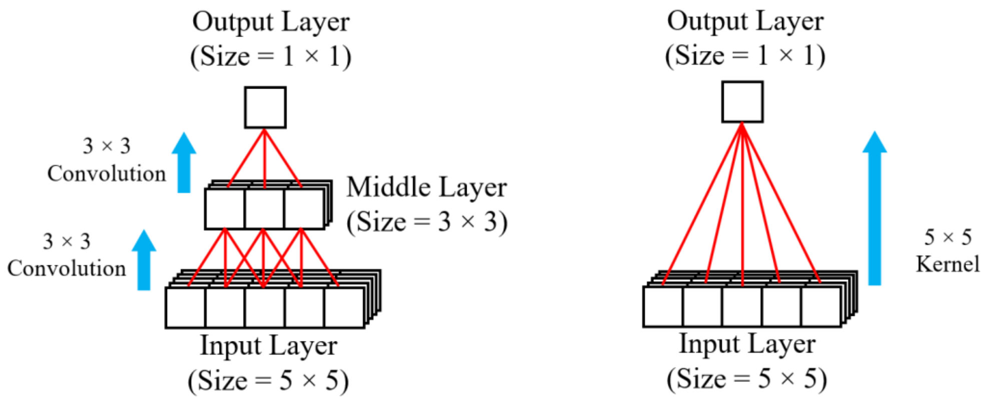


Figure 5.2. Architecture of CrackNet-V

As shown in **Fig. 5.2**, the “CrackNet-V” architecture designed for pixel-level crack detection consists of three major components: a) one pre-processing layer; b) eight deep convolutional layers and c) one output layer. At the beginning of this model, the pre-processing layer prepares 4 rectified inputs in a standard data range. Subsequently, deep cascaded convolutional layers are implemented for hierarchical feature extraction and then to generate a series of feature maps. Finally, the output layer merges feature maps to generate a possibility map for crack recognition at each pixel from the input image.

In convolutional layers, feature maps between adjacent layers are generated by corresponding filter kernels. Each pixel of input maps at previous layer shares the same set of filters. The first six convolutional layers (C1 to C6) with a series of fixed 3×3 filters are utilized to hierarchically extract spatial invariant features and then summarize them as feature maps. At the seventh convolutional layer (F7), a series of 15×15 filters are applied to connect a small visual receptive field surrounding at each pixel. The subsequent two fully-connected layers (F8 and output layer) use 1×1 convolutions to provide more non-linear activations for accuracy improvement of crack pixel detection. In addition, F7, F8 and output layers together can be regarded as fully-connected Multi-layer Perceptron (MLP) for pixel-level pattern recognition using feature maps from previous convolutional layers.

It is noticeable that in the first six convolutional layers, the sizes of all filter kernels are set as 3×3 for reducing the number of learnable parameters while increasing network depth at the same time to introduce more non-linear transformations for network performance enhancement (Simonyan and Zisserman, 2014). As **Fig. 5.3** shows, two successive convolutional layers using 3×3 kernels (**Fig. 5.3(a)**) are able to represent a 5×5 visual receptive field, which are almost equal to one single convolutional layer using a 5×5 kernel (**Fig. 5.3(b)**). However, the number of parameters in **Fig. 5.3(a)** is 2×3×3 = 18, which is much fewer than that in **Fig. 5.3(b)** (i.e. 5×5 = 25). Similarly, three successive convolutional layers using 3×3 kernels have a 7×7 effective receptive field, but the number of parameters are only 27. Based on such a concept, CrackNet-V increases the depth of convolutional layers while introducing only a small number of extra parameters.



(a) A Stack of Two 3×3 Conv. Layers **(b)** One 5×5 Conv. Layer

Figure 5.3. A Stack of Two 3×3 Conv. Layers Has an Effective Receptive Fields of 5×5

Different from traditional CNN, no any pooling layers are involved in this network. Pooling layer is an important characteristic of CNN to reduce trainable parameters while still keeping the useful information for classifying an object even if this object has shifting or scale-change (e.g. the position of a car may be shifted to anywhere in an image, which causes negative impacts for object classification). However, the goal of pixel-level crack detection is to determine a specific pixel centered at its surrounding neighbors is crack or not. Therefore, if the position of a cracking pixel is shifted away from the center pixel, the information of this shifted pixel is not necessary to be remained, which means pooling layer is not necessary in the case of pixel-level crack detection. In fact, some experiments indicate that pooling layers can increase computational costs while have little helpful effects for network accuracy enhancement (Springenberg et al., 2014; Zhang et al., 2017b).

Similar to those used in the cell-based CNN of Chapter 4, the sigmoid-like activation unit LeCun Tanh is adopted at the output layer of the CrackNet-V for a binary classification problem at each pixel. The *Xavier Initialization* is applied to initialize all the parameters in the network. The Back-Propagation (BP) combining with mini-batch Stochastic Gradient Descent (SGD) is utilized to compute gradients and update weights at each iteration.

Table 5.1 summarizes the parameter configurations in CrackNet-V. CrackNet-V only has 64,113 learnable parameters in total, which is much fewer than roughly 250,000 parameters in the cell-based CNN of Chapter 4. Although there are fewer parameters to be tuned automatically, CrackNet-V is able to implement binary classification at each pixel using a deeper network structure. Due to the decrease in trainable parameters, the time efficiency of CrackNet-V is higher than that of cell-based CNN.

Table 5.1. Trainable Weights and Biases in CrackNet-V

Layer	#inputs	#Nodes	Ftr Size	Weights	Biases	Total
C1	4	8	3×3	288	8	296
C2 to C6	8	8	3×3	576 (2880)*	8 (40)*	584 (2920)*
F7	8	32	15×15	57600	32	57632
F8	32	96	1×1	3072	96	3168
Output	96	1	1×1	96	1	97
Total	-	-	-	63,936	177	64,113
*the data in brackets are the sum of the five layers from C2 to C6						

5.2.2 Pre-Processing Layer

Image pre-processing is a critical step to prepare input data for CrackNet-V. As shown **Fig. 5.4(a)**, the original pavement 3D surfaces have shape distortion caused by the unevenness of pavement surface. This type of data distortion, regardless of its severity degree, can adversely impact on pixel-level detection performance because the pixel cracking features are extracted using the concept of "shared weights at each pixel". In other words, the cracking information at each pixel should be processed for information uniformization as much as possible so that the results at varying locations can be consistent through the same network. Therefore, the pre-processing layer is a necessary component to rectify surface data distortions, so that the corrected images with higher quality can be obtained for further cracking detection purpose.

The goal of rectifying process is to remain details on the pavement images while remove pavement unevenness as much as possible. Two procedures are conducted at the pre-processing layer to maintain the local details while ignoring global surface unevenness. First, median filter is used to determine the representative elevation of a local region. Subsequently, the elevation of each pixel within the local region is then adjusted based on the representative local elevation. Second, Z-score Normalization is implemented to confine the data in a standard range.

Median filter is a nonlinear low-pass filter with capability of removing high-frequency information (e.g. cracks or spot noises on pavement) while introducing minor distortion to low-frequency parts (e.g. terrain surface) (Huang et al., 1979). In other words, the height value of a local image patch around a specific pixel can be estimated by the median filter. Consequently, the rectified height of a pixel can be represented by the difference between its original height and estimated height of the local terrain surface. Such a rectifying procedure for each pixel is defined as follows:

$$H_{(x,y)} = \boldsymbol{med}(I_{(x-r,y-r)}, \dots, I_{(x,y)}, \dots, I_{(x+r,y+r)}) \qquad \textbf{Eq. 5.1}$$

$$I'_{(x,y)} = I_{(x,y)} - H_{(x,y)} \qquad \textbf{Eq. 5.2}$$

where $I_{(x,y)}$ is the original height of pixel (x, y); r defines the kernel size of the median filter; $H_{(x,y)}$ is the estimated local terrain surface height around pixel (x, y); $I'_{(x,y)}$ is the rectified height of pixel (x, y); and $\boldsymbol{med}(x)$ is the median value among the heights of pixel sets x.

After completion of the rectifying process using median filter, a common normalization technique, Z-Score Normalization, is applied to the whole image to guarantee the input data has a zero mean and a standard deviation of 1. The Z-score Norm is expressed as the following equation:

$$Z_{(x,y)} = \frac{I'_{(x,y)} - \boldsymbol{mean}(I')}{\boldsymbol{std}(I')} \qquad \textbf{Eq. 5.3}$$

where $Z_{(x,y)}$ is for the z-score of pixel (x, y); $\boldsymbol{mean}(I')$ is the mean value of rectified image I'; and $\boldsymbol{std}(I')$ is the standard deviation of rectified image I'.

As a result, the pre-processing can greatly suppress the negative effects caused by surface unevenness presented on the down-sampled 3D image (**Fig. 5.4(b)**), and prepare well-rectified input data for CrackNet-V. It should be noticed that the pre-processing layer produces 4-channel input data. As the optimal size of median filter is hard to be determined, 4 different sizes of median filters are respectively applied for each channel in the pre-processed layer. In other words, totally 4 pre-processed images participate as input data in network training and testing. The median filter size at the 1st channel is set to 9×9, and then gradually expands by 2×2 so that the 4th channel eventually reaches the size of 15×15. According to experiments, the network performance based on multiple pre-processed inputs is significantly improved by contrast with that based on only one pre-processed input.

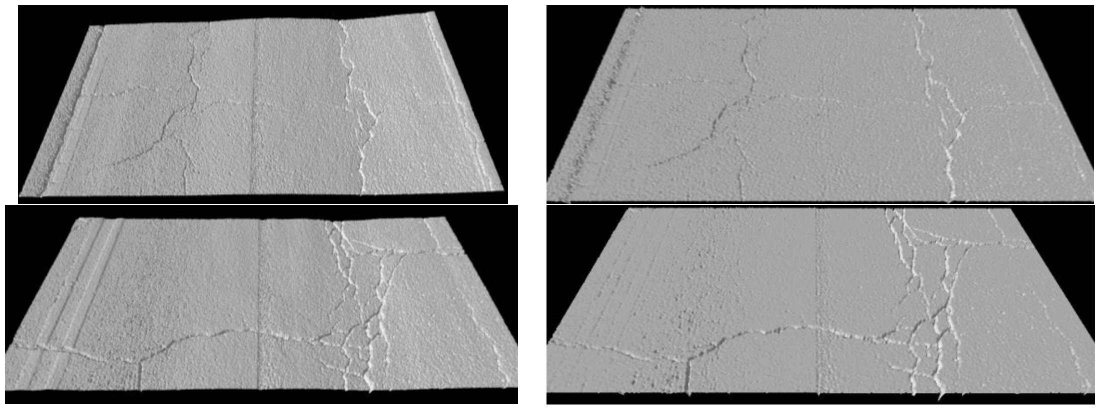


Figure 5.4. OpenGL Visualization for Pavement 3D Image before and after Pre-processing

5.2.3 Activation Units in Hidden Layers

3D data sets are able to represent realistic pavement cracks at different depths. However, the responses of cracks after convolution may have a large difference due to the variation of crack depths. After passing through the general activation unit Leaky Rectifier Linear Unit (ReLU) (Maas et al., 2013), such a difference is still remained for the following layers of the network, which causes the network to concentrate on learning the depth range of cracks. Besides, the depth range of cracks is not unique when other noise patterns are considered. The uniqueness of cracks is highly related with the morphological information instead of the depth information.

In fact, the proposed network always ignores some shallow cracks if Leaky ReLU is applied at the hidden layers. To solve this problem, the new activation unit called Leaky Rectified Tanh is proposed in this dissertation to control the magnitude of convolution results such that the shallow cracks have similar responses with deeper cracks.

The proposed activation unit Leaky Rectified Tanh with $alpha = 0.1$ can be defined as:

$$\boldsymbol{\sigma}(x) = max\left(1.7159 \times tanh\left(\frac{2}{3}\,x\right), 0\right) + min(0.1\,x, 0) \qquad \textbf{Eq. 5.4}$$

The $tanh$ with scale factors of 1.7159 in y-axis and $\frac{2}{3}$ in x-axis is adopted to regulate data magnitude as same as data magnitude of output neuron. **Fig. 5.5** describes the difference between Leaky Rectified Tanh and Leaky ReLU. Generally speaking, the pattern of Leaky Rectified Tanh is almost as same as Leaky ReLU when $x \in (-\infty, 1]$; however, Leaky Rectified Tanh can effectively control the data magnitude if $x \in (1, +\infty]$. In other words, if the responses after convolution are negative, the parameter α can compress the responses into a small range. On the other hand, if the convolutional responses are positive, the differences among responses of cracks with different depths can be suppressed with Leaky Rectified Tanh.

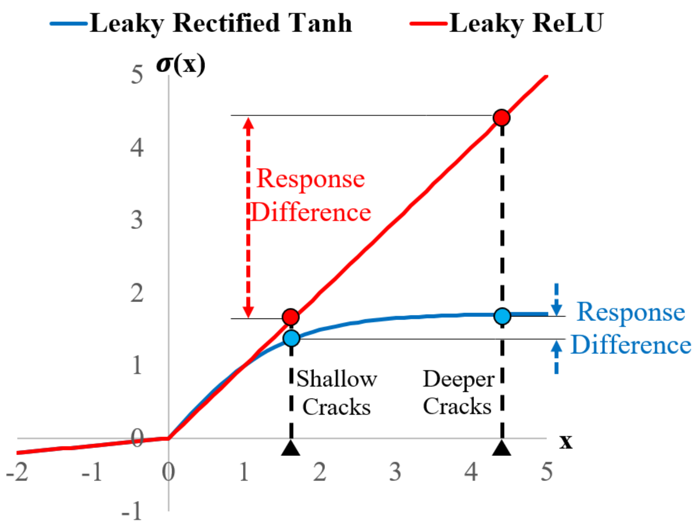


Figure 5.5. Comparison between Leaky Rectified Tanh and Leaky ReLU

5.2.4 Sensitivity Regularization at Output Layer

Since the output layer of CrackNet-V utilizes the activation unit LeCun Tanh for binary classification at each pixel, the cost function is the corresponding cross-entropy of LeCun Tanh combined with L2 weight decay. And the derivative of cost function $\boldsymbol{C}$ with respect to network weights $\boldsymbol{w}$ determines the amounts of weights updating as following equations express:

$$\frac{\partial \boldsymbol{C}}{\partial \boldsymbol{w}} = \frac{\partial \boldsymbol{C}}{\partial \boldsymbol{z}} * \frac{\partial \boldsymbol{z}}{\partial \boldsymbol{w}} \qquad \textbf{Eq. 5.5}$$

$$\boldsymbol{z} = \boldsymbol{\sigma}(\boldsymbol{w} * \boldsymbol{x} + \boldsymbol{b}) \qquad \textbf{Eq. 5.6}$$

where $\boldsymbol{x}$ are the inputs of a network layer; $\boldsymbol{w}$ and $\boldsymbol{b}$ are the weights and bias respectively in this layer; $\boldsymbol{\sigma}$ is the activation function of this layer; and $\boldsymbol{z}$ is the output.

In **Eq. 5.5**, $\frac{\partial \boldsymbol{C}}{\partial \boldsymbol{z}}$ is defined as the "sensitivity". Since the output neural utilizes cross-entropy to evaluate the errors between network outputs and targets, the sensitivity at output layer can be computed through the following equation:

$$\boldsymbol{S_{out}} = \frac{\partial \boldsymbol{C}}{\partial \boldsymbol{z}} = \boldsymbol{t} - \boldsymbol{z} \qquad \textbf{Eq. 5.6}$$

where $\boldsymbol{S_{out}}$ is the sensitivity value at the output layer; $\boldsymbol{t}$ is the network target; and $\boldsymbol{z}$ is the network output.

Different from the cell-based CNN, however, CrackNet-V cannot directly use the sensitivity values for weights updating during training process. In the cell-based CNN model of Chapter 4, single image is divided as 128 image blocks, which means only 128 sensitivity factors at output layer for each block are back-propagated once. In CrackNet-V, by contrast, single image passing through the network has over one hundred thousands of sensitivity factors (i.e. 512×256 =

131,072). Back-propagation of such a large number of sensitivity factors can easily cause gradient explosions, resulting in the failure of network training process. Therefore, it is necessary to regularize the sensitivity factors at the output layer so that the network can be trained successfully.

In this dissertation, a regularization rule is proposed to prevent gradient explosion caused by accumulation of sensitivity factors, while still remaining valid information for network training. This regularization rule is conducted at each pixel of the sensitivity map at output layer based on the number of both cracking pixels and non-cracking pixels, which is summarized as following equation:

$$\boldsymbol{S_c = S_{out} \, / \, p_c} \qquad \textbf{Eq. 5.7}$$

$$\boldsymbol{S_{nc} = S_{out} \, / \, p_{nc}} \qquad \textbf{Eq. 5.8}$$

where $\boldsymbol{S_{out}}$ is the sensitivity factors at the output layer; $\boldsymbol{S_c}$ and $\boldsymbol{S_{nc}}$ are the sensitivity factors of cracking and non-cracking pixels respectively that are used in back-propagation process; $\boldsymbol{p_c}$ is the number of cracking pixels in an image; and $\boldsymbol{p_{nc}}$ is the number of non-cracking pixels in an image.

Fig. 5.6 illustrates the sensitivity regularization rule on an output map. Assume this sensitivity map at the output layer is in size of 4×4 with 5 pixels labeled as cracks in it. Before beginning back-propagation of this map, the sensitivity values at all cracking pixels are divided by 5, while those at non-cracking pixels are divided by 11. It means that the regularization rule is equivalent to setting the ratio of cracking samples to non-cracking samples as 1:1. In other words, this rule can be considered as a way to balance the number of training samples (i.e. cracking and non-cracking pixels).In addition, this regularization computes the average of all sensitivity values, so that the noises in sensitivity map can be suppressed for accurate weight updating. Moreover, although there are still one hundred thousands of sensitivity values needing to be propagated backward, the magnitudes of gradients for weight updating are controlled in a reasonable range due to value decreasing in the sensitivity map. As a result, the network can be trained successfully using this regularization strategy.

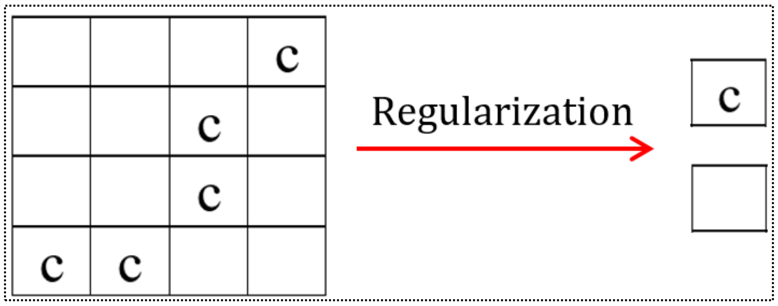


Figure 5.6. Illustration of Sensitivity Regularization

5.2.5 Summary of CrackNet-V

Table 5.2 summarizes the difference between cell-based CNN and pixel-level CrackNet-V. According to this table, the essential concept applied in CrackNet-V is to implement shared weights at each pixel. Particularly, Pooling layers are removed from the network to prevent cracking pixel from position shifting. In addition, pre-processing layer is specifically designed to remain local crack characteristics while ignore pavement surface unevenness. Moreover, convolutional layers utilize a series of filters in the size of 3×3 to reduce the number of parameters while increase network depth for performance improvements. It is noticeable that decrease in the number of parameters means increase in the network speed, which benefits for big pavement data processing. Furthermore, the change of activation units in pixel-CNN is able to suppress the difference in convolutional responses of cracks at varying depths, resulting in improved detection results on shallow cracks. Last but not least, a special regularization rule is implemented on the sensitivity map at output layer of CrackNet-V for network training purpose.

Table 5.2. Comparison between Cell-Based CNN and CrackNet-V

Modifications	Cell-based CNN	CrackNet-V	Targets of Modifications
Concept	Weights Shared by Image Cells	Weights Shared by Pixels	Conversion of detection accuracy from grid-level to pixel level
Pooling Layers	Yes	No	Prevention of pixel shifting
Pre-processing Layer	No	Yes	Elimination of pavement unevenness for sharing weights at pixels
Conv. Layers	Two Layers with Filter Kernel (9×9)	Six Layers with Filter Kernel (3×3)	Reducing the number of parameters, while introducing more non-linear transformation
Activation Units in Hidden Layers	Leaky ReLU	Leaky Tanh	Uniformization of convolutional responses for cracks at varying depths
Sensitivity Regularization	No	Yes	Prevention of gradients explosion for network training

5.3 Experimental Results

5.3.1 Evaluation Method

The indicators precision ($\boldsymbol{Pr_{pixel}}$), recall ($\boldsymbol{Re_{pixel}}$), and F1 score ($\boldsymbol{F1_{pixel}}$) are utilized to evaluate CrackNet-V performance during both training and testing process. The pixel-level precision and recall can be computed on true positive ($\boldsymbol{TP}$), false negative ($\boldsymbol{FN}$) and false positive ($\boldsymbol{FP}$) as following equations:

$$\boldsymbol{Pr_{pixel} = TP/(TP + FP)} \quad \textbf{Eq. 5.9}$$

$$\boldsymbol{Re_{pixel} = TP/(TP + FN)} \quad \textbf{Eq. 5.10}$$

$$\boldsymbol{F1_{pixel} = 2 \times Pr_{pixel} \times Re_{pixel}/(Pr_{pixel} + Re_{pixel})} \quad \textbf{Eq. 5.11}$$

where $\boldsymbol{TP}$ is a crack classified correctly at a pixel; $\boldsymbol{FP}$ is a non-crack classified incorrectly at a pixel; and $\boldsymbol{FN}$ is a crack classified incorrectly at a pixel.

Considering that the pixel-level ground truth is manually labeled and there may be transitional areas between crack pixels and non-crack pixels in real images, small distance between the detected results and the truths is accepted during evaluation. In this dissertation, the detected pixels which are no more than 3 pixels away from the manually labeled truths are considered as true positive pixels.

5.3.2 Validation Set

To estimate the network performance during the training process, 15 different asphalt images are carefully selected from the standard image library mentioned in Chapter 4. As **Fig. 5.7** shows, validation sets from #1 to #7 are selected to represent normal cracks on asphalt pavements; sets from #8 to #10 are selected to represent tiny cracks; sets from #11 to #12 have pavement shoulders on the right side of the images; sets from #13 to #14 have shallow cracks near the pavement markers on the left of the images; #15 is applied to validate performance on the pavement without any cracks. During training process, both the error of cost function and F1 indicator are measured on the validation set.

Although validation set only contains 15 samples, it is sufficient for monitoring the network performance during training process because only two objects need to be classified, and the quality of entire 3D dataset is stable and consistent. In addition, the small scale of validation set can save time in the entire training process.

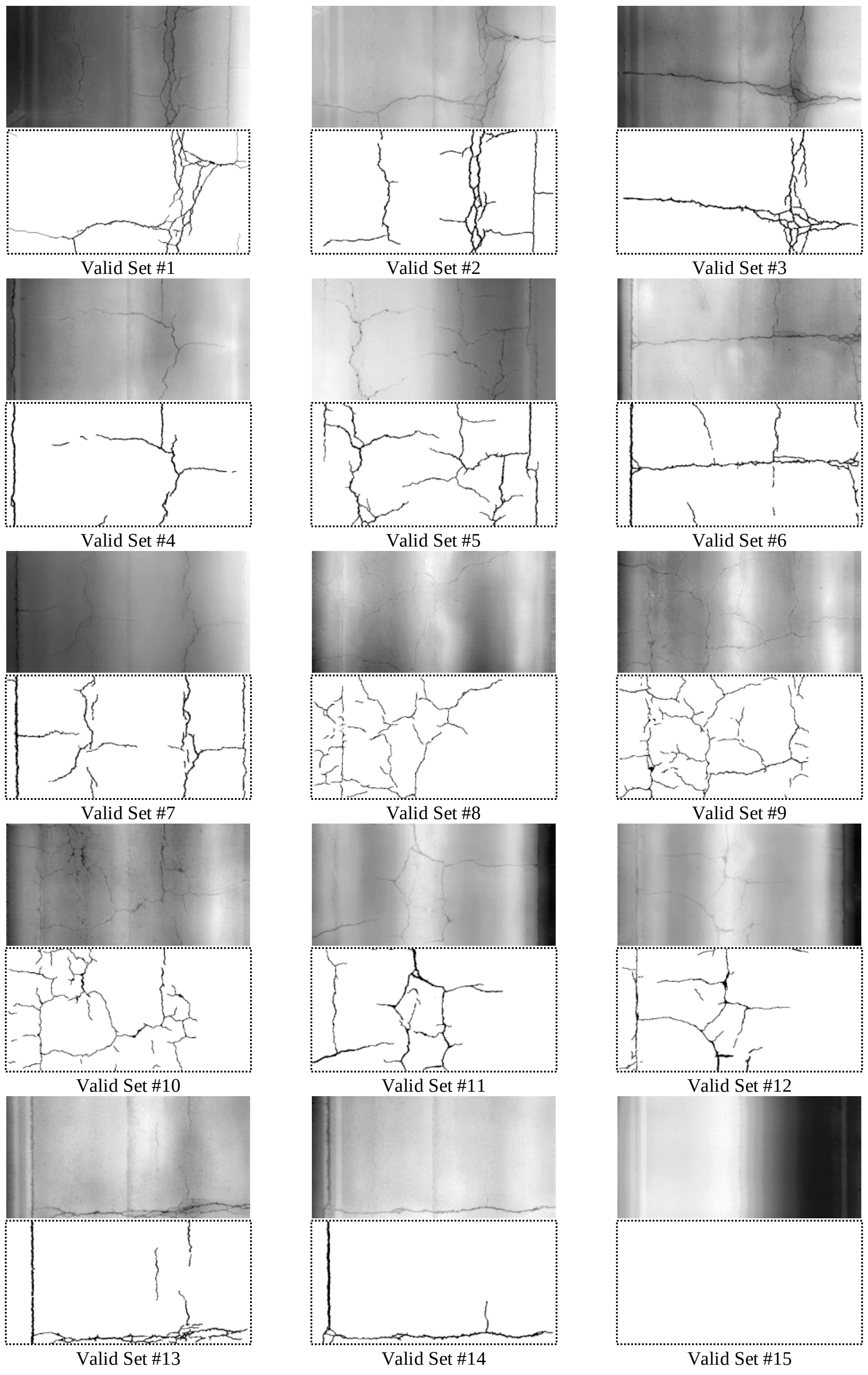


Figure 5.7. Validation Set for Monitoring Network Performance during Training Process

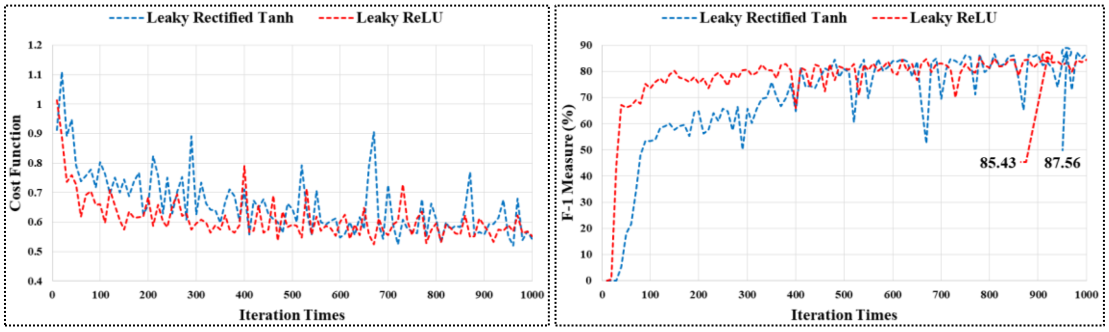
5.3.3 Network Training

To evaluate the effectiveness of Leaky Rectified Tanh proposed in the dissertation, two networks with different activation units in the hidden layers are trained for comparison study. The network with proposed Leaky Rectified Tanh is denoted as "Config-A", while the network with traditional Leaky ReLU is denoted as "Config-B".

The target value of a pixel is set as 1.7159 if it is a cracking pixel, otherwise it is -1.7159. The mini-batch size is 4 at the beginning and then gradually increases to 8 as training continues. The momentum variable is set as 0.9, which is found to be very helpful for stabilizing network training. L2 weight decay is 0.0005 during entire training process. All layers share the same learning rate. The learning rate is set as 0.01 at the beginning, and then divided by 2 (i.e. 0.005) after 200 iterations training or divided by 3 (i.e. 0.0033) after 500 iterations training. Finally, the network is fine-tuned with learning rate 0.0025 to seek for further optimization after 1,000 iterations. No Drop-Out technique is applied in the network. The network training with 3,000 iterations is completed in only one day with the aid of a single GPU device (NVidia GeForce GTX 1080Ti).

Totally 2,568 images are involved in network training. **Fig. 5.8** shows the comparison on the validation set between the two networks at every 10 iterations. **Fig. 5.8(a)** shows the cost function values through iterations. The Config-B is slightly faster than the Config-A to reduce the cost function value. However, there is no significant difference at last 600 iterations. In **Fig. 5.8(b)**, Config-B yields higher F-1 scores at the first 400 iterations compared with Config-A. In other words, Config-B learns faster at the beginning. This is reasonable because Leaky Rectified Tanh uses sigmoid-like function and may result in zero gradients due to saturation. In a general view, the Config-B yields more consistent F-1 scores.

However, for the first 1,000 iterations, the best F-1 score achieved by Config-A on validation data is better than that produced by Config-B (**Table 5.3**). The best F-1 score has little change from 1,000 to 3,000 iterations, thus the network training is finally stopped after 3,000 iterations. As shown in **Table 5.3**, the best F-1 score of Config-A performs generally better than Config-B, particularly with higher Recalls, indicating that Config-A is able to detect more shallow cracks.



(a) Objective Function **(b)** F-1 Measure (%)

Figure 5.8. Performance Comparison on Validation Set during Training Process

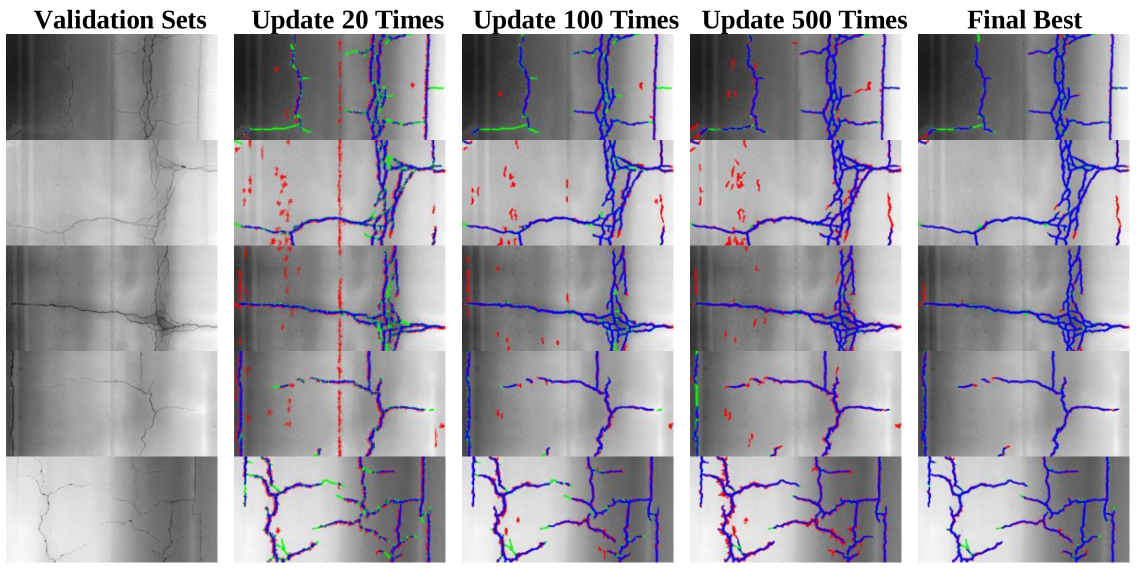
Table 5.3. Comparison between Leaky Rectify Tanh and Leaky ReLU on Valid Set

Config	**Iter.**	**Precision (%)**	**Recall (%)**	**F1 Score (%)**
A (Leaky Rec. Tanh)	1,000	87.36	87.77	87.56
	2,000	87.54	88.42	87.98
	3,000	88.08	88.54	88.31
B (Leaky ReLU)	1,000	86.96	83.96	85.43
	2,000	87.73	84.57	86.12
	3,000	88.80	84.45	86.57

For Config-B (Leaky ReLU) (**Fig. 5.9**), the network works well during training process for most validation images excepting for validation set #13 and #14 (i.e. the two used to represent shallow cracks on asphalt road). At the 20th training iteration, crack responses (i.e. color blue: true-positive) begin to appear for all validation images with a lot of noises (i.e. color red: false-positive pixels). As training goes on, crack responses become more and more stable especially on

#1 to #7. Besides, more tiny cracks on #8 to #12 can be captured, while noise responses on pavement shoulders (#11 to #12) or on non-cracking environment (#15) are largely removed by the network. However, Config-B suffers a problem in detecting shallow cracks. For validation image #13 and #14, shallow cracks near the pavement marker on the left-side are gradually abandoned during network training. In other words, the network with Leaky ReLU tends to detect middle and fine cracks more but miss shallow cracks.

On the other hands, different performance is shown in Config-A (Leaky Rec. Tanh) (**Fig. 5.10**). At the 20th training iteration, there is no crack responses shown up for all validation images (i.e. color green: false-negative), which also demonstrates the learning speed of Config-A is slower than it of Config-B. As training goes on, crack responses also become to appear and then to stabilize while noise responses are mostly removed. Generally speaking, the Config-A performance on validation set is similar to Config-B. For several images with fine cracks (#8 to #10), Config-A perform slightly worse than Config-B, but it is worth to notice that the shallow cracks on #13 and #14 are still remained under Leaky Rec. Tanh while those are discarded by Leaky ReLU, which causes the recall of Config-A is a little higher than it of Config-B.



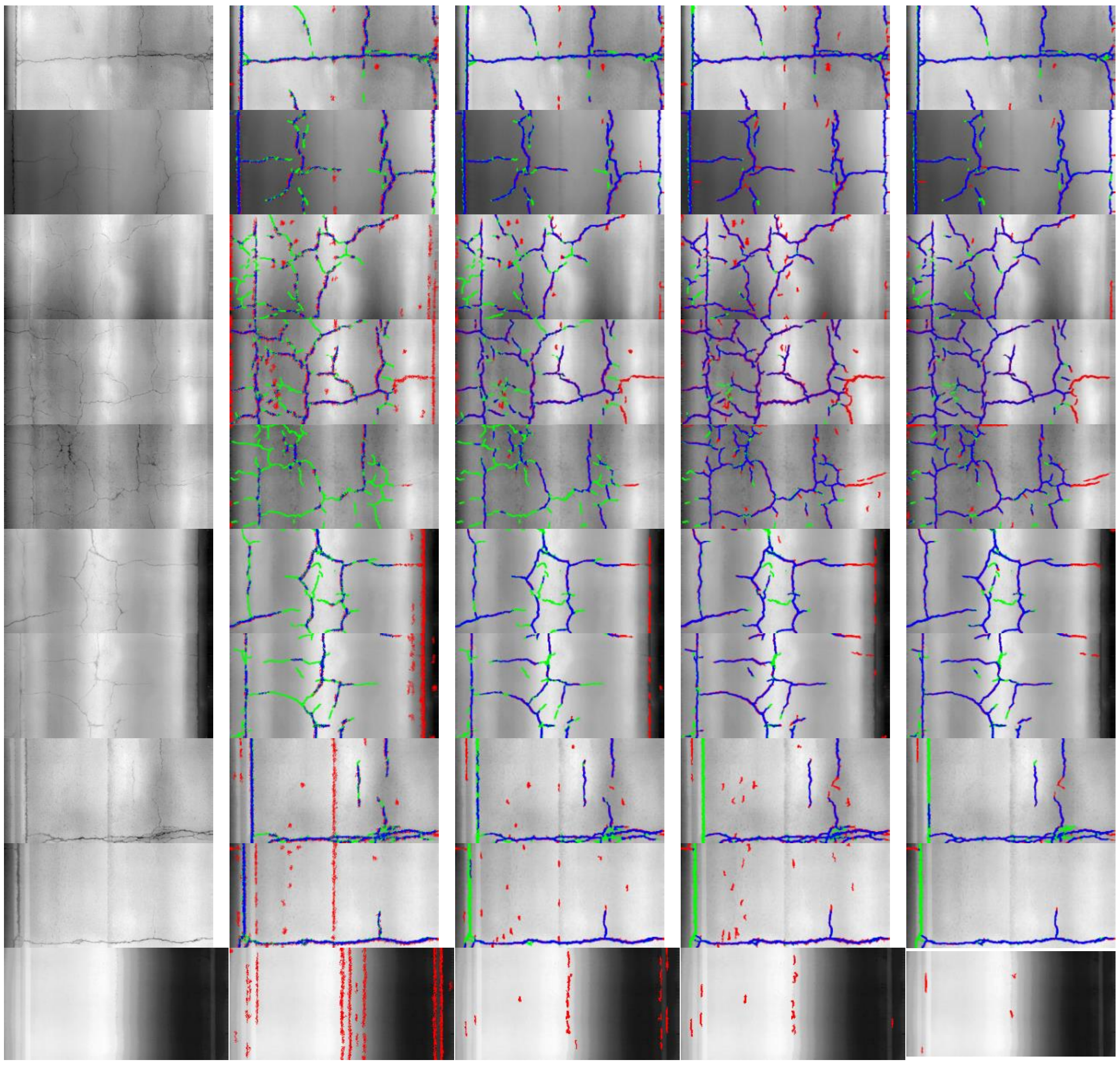
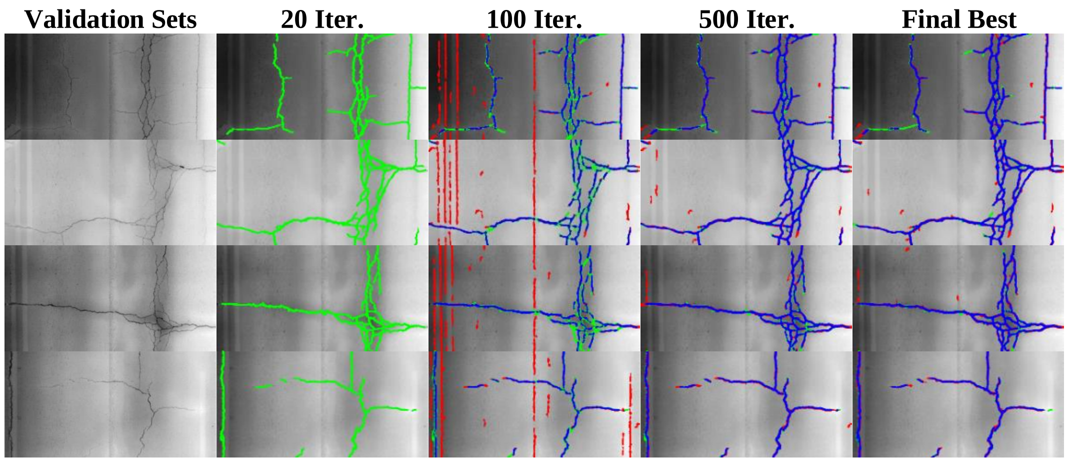


Figure 5.9. Performance Changes during Training Process (Leaky ReLU)



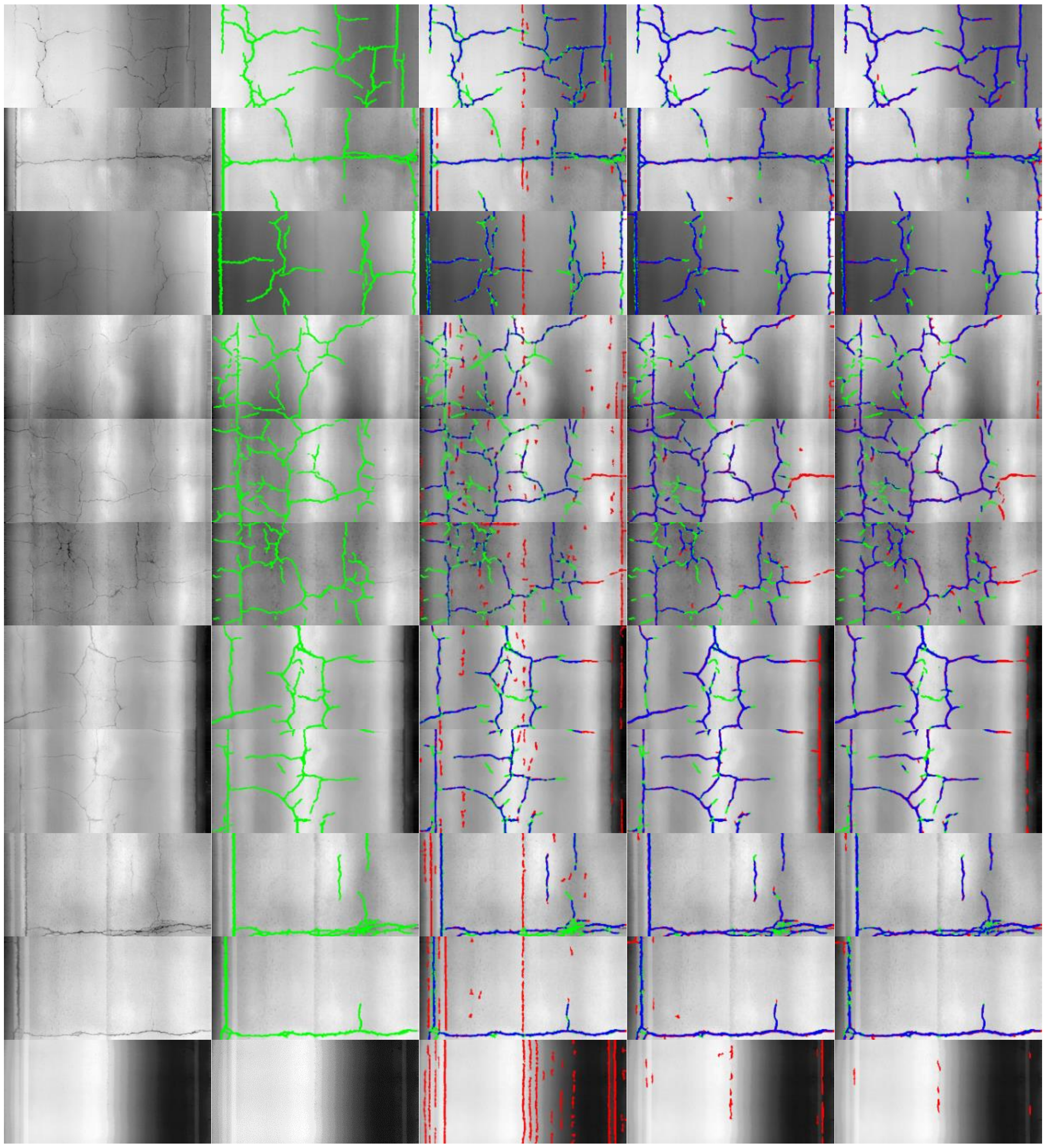
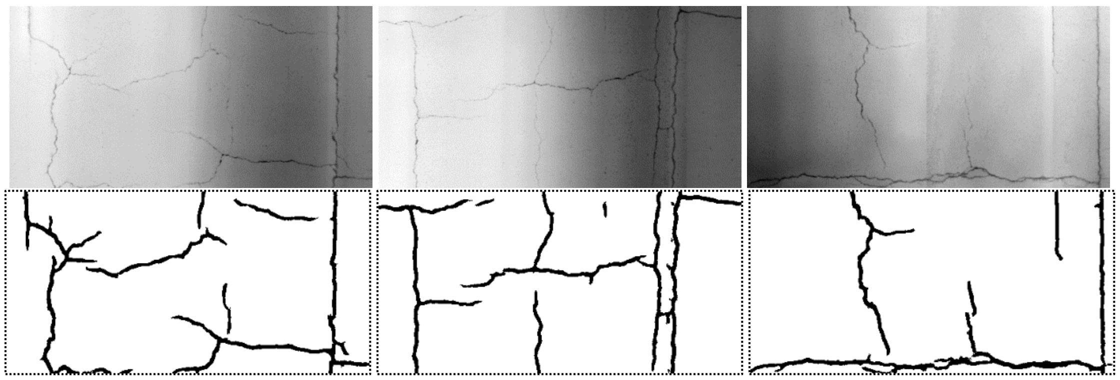


Figure 5.10. Performance Changes during Training Process (Leaky Rec. Tanh)

5.3.4 Performance on Test Set

After the network training is completed with 3,000 iterations, the network parameters that yield the highest F1 score on validation data set are applied to the testing data for further evaluating the network performance. In all 500 asphalt images are used as the testing data to evaluate the network performance. **Fig. 5.11** shows several representative outputs produced by CrackNet-V with Config-A (Leaky Rectified Tanh). **Table 5.4** summarizes the performance of CrackNet-V on the testing data. According to this table, there is no big difference between Config-A and Config-B. However, both precision and recall in Config-A are slightly higher than those in Config-B. As a result, the F1 score of Config-A (87.12%) slightly surpasses that of Config-B (86.46%). The difference between Config-A and Config-B is discussed in next section. It is noticeable that the performance on testing set shown in **Table 5.4** is similar to those on validation set in **Table 5.3**, which indicates that the small scale of validation set is sufficient for assessing the pixel-level accuracy of a two-object classification when the quality of entire dataset is consistent.


3D Testing Images and Corresponding Detection Results (#1-3)

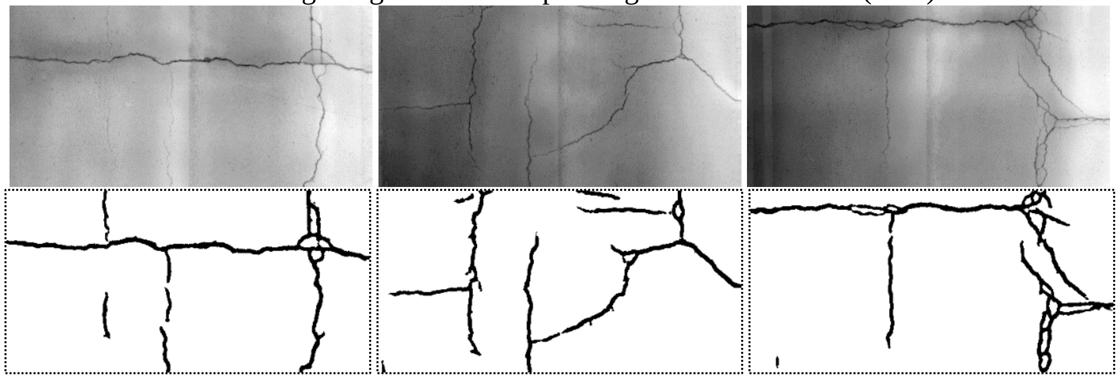

3D Testing Images and Corresponding Detection Results (#4-6)

Figure 5.11. Representative Detection Results of Pixel-Level CNN (Config.A)

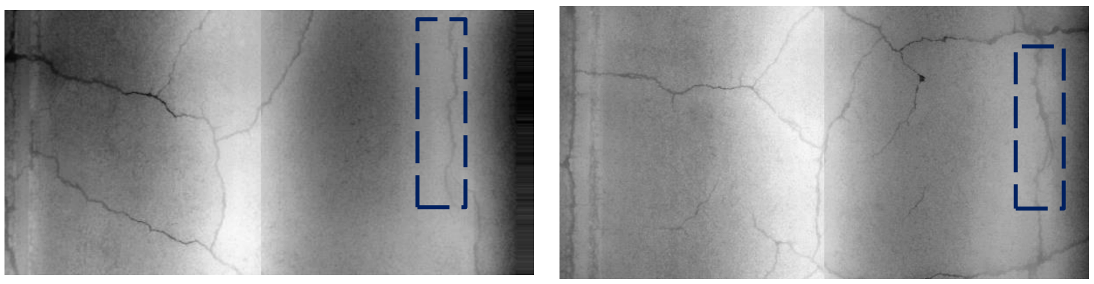
Table 5.4. CrackNet-V Performance on Test Set

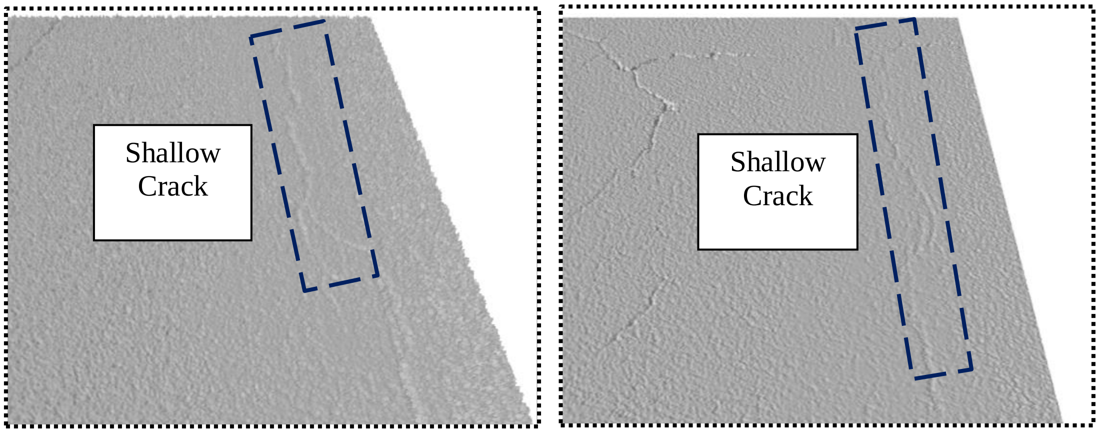
Config.	Precision (%)	Recall (%)	F1 Score (%)
A (Leaky Rec. Tanh)	84.31	90.12	87.12
B (Leaky ReLU)	83.51	89.63	86.46

5.4 Discussion

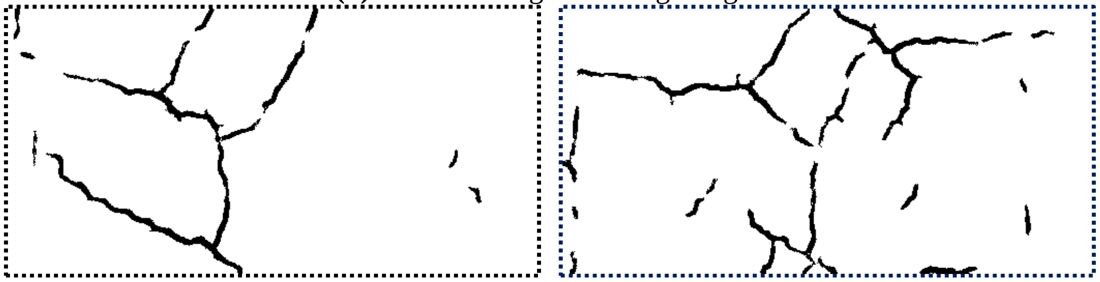
5.4.1 Leaky Rectified Tanh vs. Leaky ReLU

Comparing to the traditional activation unit Leaky ReLU, CrackNet-V with proposed Leaky Rectified Tanh has better performance in detecting shallow cracks, as illustrated in **Fig. 5.12**. **Fig. 5.12(a)** and **(b)** show several typical cracks that have shallow depths under 3D environment. These shallow cracks are difficult to be perceived even by human eyes under the 3D environment. In **Fig. 5.12(c)**, the network with activation unit Leaky ReLU tends to detect cracks whose depths exceed certain values, while omitting the shallow cracks. On the other hand, the network with Leaky Rectified Tanh can yield more accurate detection on shallow cracks, as illustrated in **Fig. 5.12(d)**. This demonstrates that the bounding property of Leaky Rectified Tanh intends to minimize the difference between a shallow crack and a deep crack, and thus encourages the network to learn more geometrical features of cracks instead of being purely dependent on the depth information. As a result, Leaky Rectified Tanh is able to detect more shallow cracks by contrast with Leaky ReLU. **Fig. 13** provides more comparisons between Leaky Rectified Tanh and ReLU on shallow crack detection.

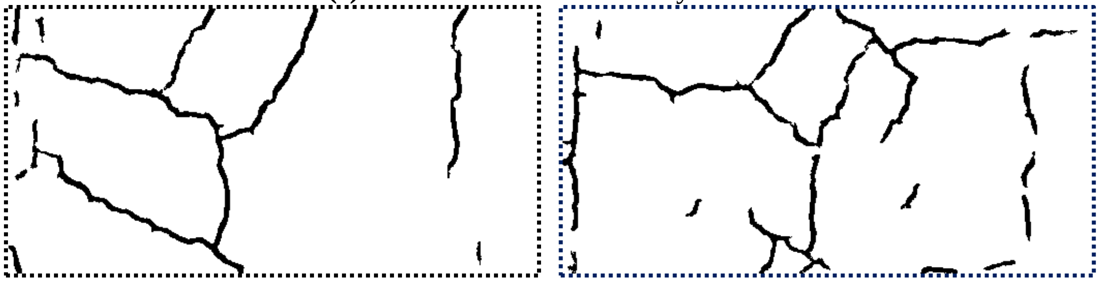

(a) 3D Testing Images



(b) 3D Rendering of Testing Images

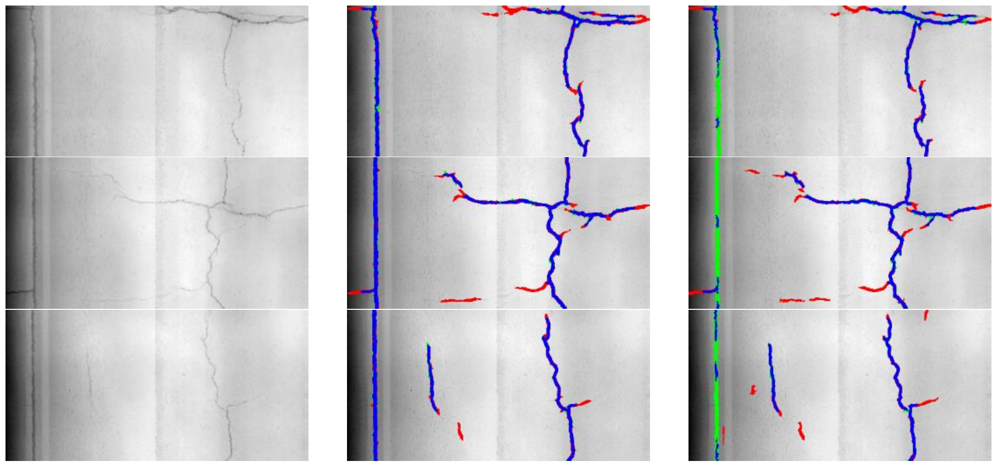


(c) Detection Results of Leaky ReLU



(d) Detection Results of Leaky Rectified Tanh

Figure 5.12. Comparison between Leaky ReLU and Leaky Rectified Tanh - I



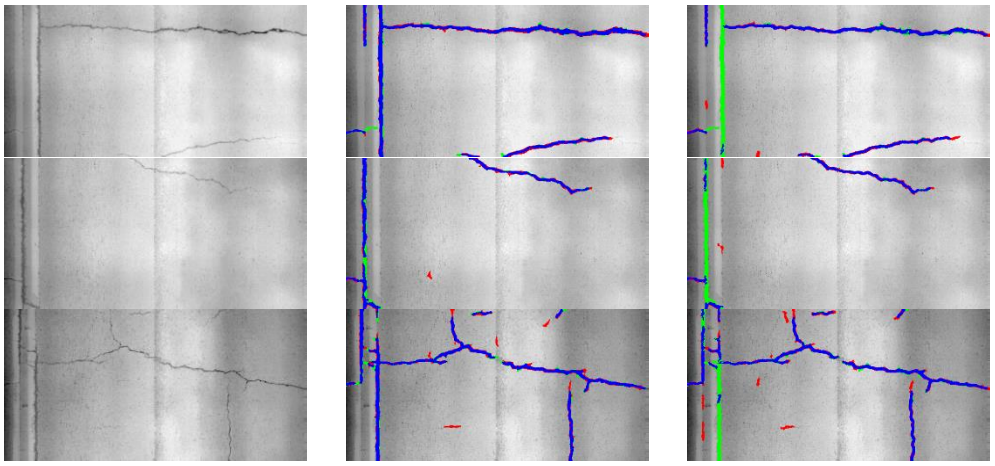


Figure 5.13. Comparison between Leaky ReLU and Leaky Rectified Tanh - II

5.4.2 CrackNet-V vs. CrackNet

To further estimate the CrackNet-V performance, CrackNet (Zhang et al., 2017b) is utilized for a comparison study. **Table 5.5** summarizes the comparison results between CrackNet and CrackNet-V on the same test set containing 500 asphalt images. Although the parameters in CrackNet-V is roughly 17 times fewer than those of CrackNet, the F1 score difference in between is very little. The F1 score of CrackNet-V (87.12% or 86.46%) slightly surpasses the F-1 score produced by CrackNet (85.62%). CrackNet yields a high Precision up to 90.86%, while the Precision of CrackNet-V is roughly 84%. However, CrackNet-V can achieve a significantly higher Recall 90.12% in comparison with the Recall 80.96% resulted from CrackNet. Generally, CrackNet-V is more robust in detecting fine or shallow cracks, while CrackNet performs better for wide cracks. **Fig. 5.14** and **Fig. 5.15** illustrate the comparison between CrackNet and CrackNet-V in detecting fine cracks and wide cracks respectively.

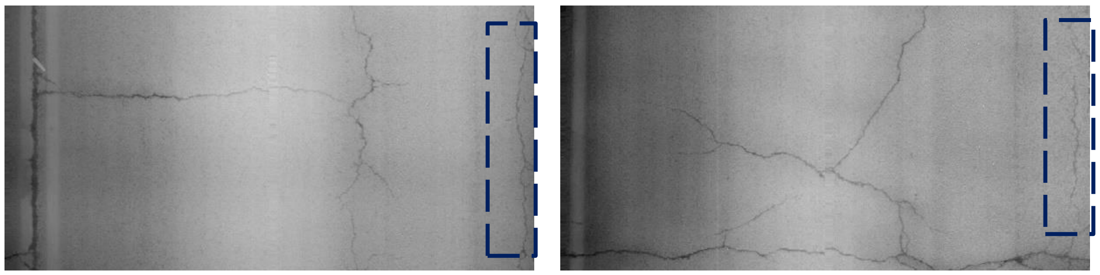
Table 5.5. Comparison between CrackNet and CrackNet-V on Testing Data

Network Type	**Number of Parameters**	**Configuration**	**Precision (%)**	**Recall (%)**	**F1 (%)**
CrackNet	1,159,561	N/A	90.86	80.96	85.62
CrackNet-V	64,113	*A* (Leaky Recti. Tanh)	84.31	90.12	87.12
		B (Leaky ReLU)	83.51	89.63	86.46

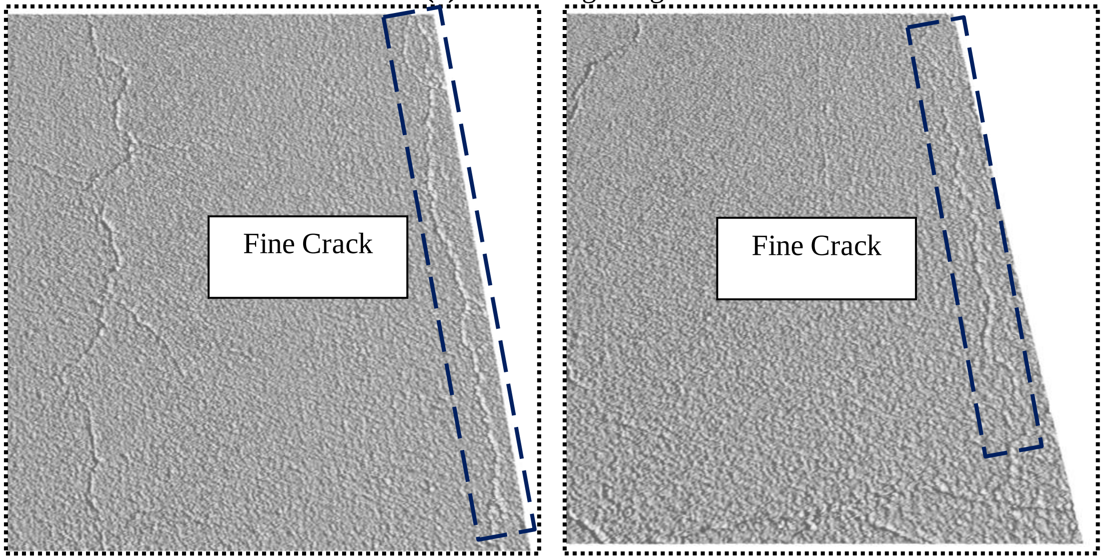
Fig. 5.14(a) and **(b)** show some typical fine cracks under 3D environment locating on the right-side of the images. In **Fig. 5.14(c)**, CrackNet is able to detect most cracks in the middle but misses some fine cracks on the right-side, although these fine cracks have no obvious changes in width compared with other cracks in the middle. A possible reason is that these fine cracks are not fully extracted by the pre-designed filters used in CrackNet due to their shallow depths. In **Fig. 5.14(d)**, the fine cracks missed by CrackNet are captured by CrackNet-V successfully. In other words, CrackNet-V is more sensitive to fine cracks, which probably benefits from the use of filter kernels with extremely small size (3×3).

In **Fig. 5.15**, however, CrackNet outperforms CrackNet-V in detecting wide cracks. **Fig. 5.15(a)** and **(b)** show some wide cracks near the pavement markers on the left-side of images. In **Fig. 5.15(c)**, CrackNet is able to detect the whole of wide cracks near the markers. In **Fig. 5.15(d)**, CrackNet-V only recognizes parts of the wide cracks while missing some wide parts, which is probably caused by the adverse impacts of small filters. Even though the successive layers with small filters theoretically represent a wide range of local receptive fields, the wide cracks could not be completely perceived by such small filters.

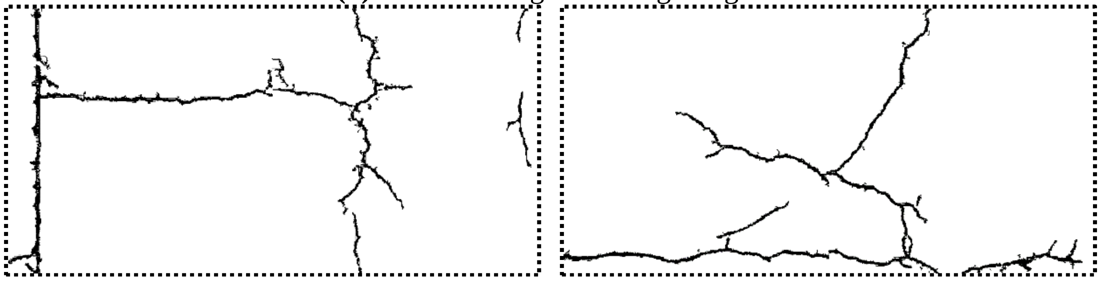
In a summary, the small size of filter kernels used in CrackNet-V has both advantage and disadvantage. On the one hand, small filters concentrate on feature extraction in small receptive field, thus they are more sensitive and efficient in detecting fine cracks. On the other hand, small filters in successive layers could not perfectly represent a wide receptive field, resulting in partial detection of wide cracks.



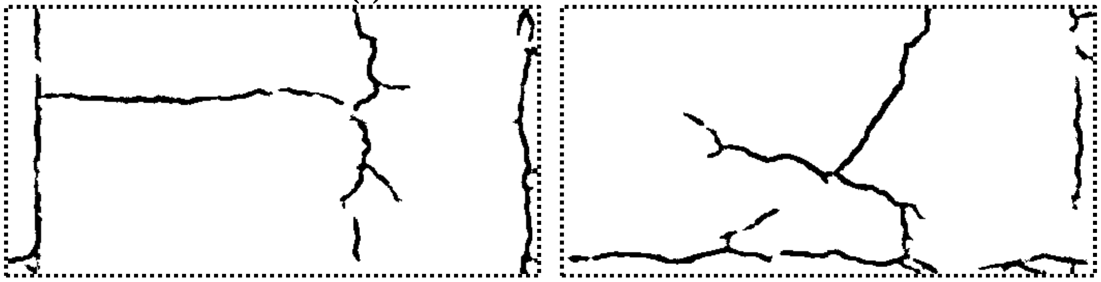
(a) 3D Testing Images



(b) 3D Rendering of Testing Images

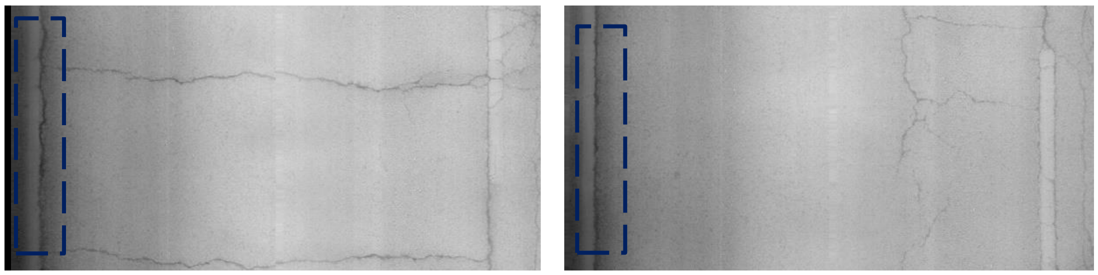


(c) Detection Results of CrackNet

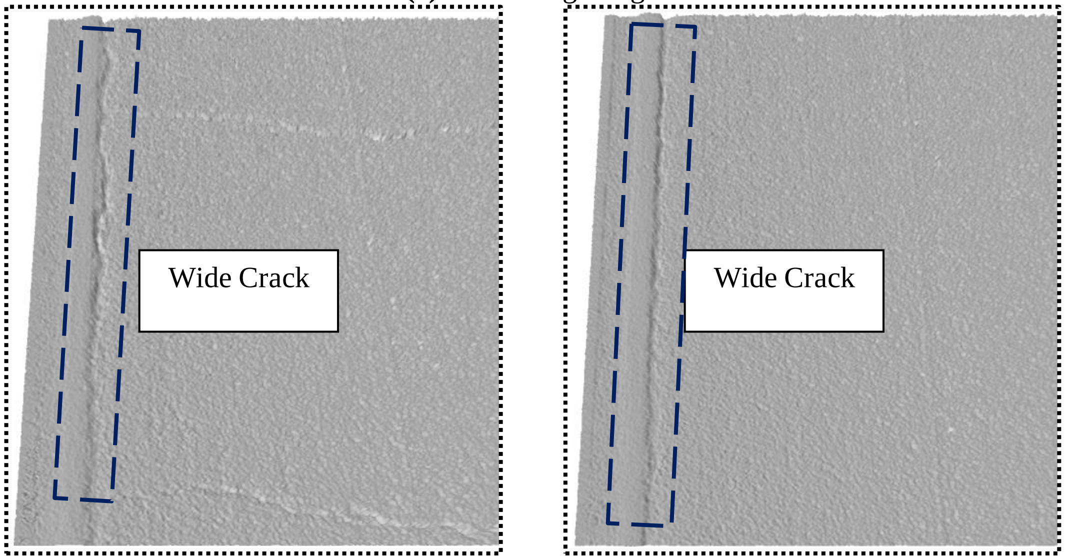


(d) Detection Results of CrackNet-V

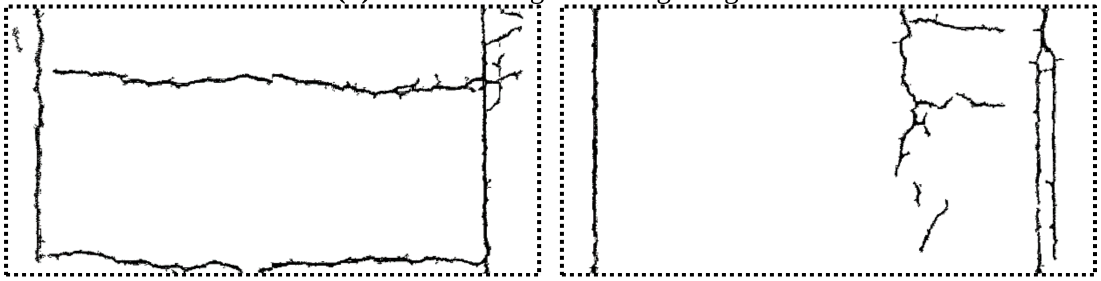
Figure 5.14. Comparison between CrackNet and CrackNet-V (Config-A) on Fine Cracks



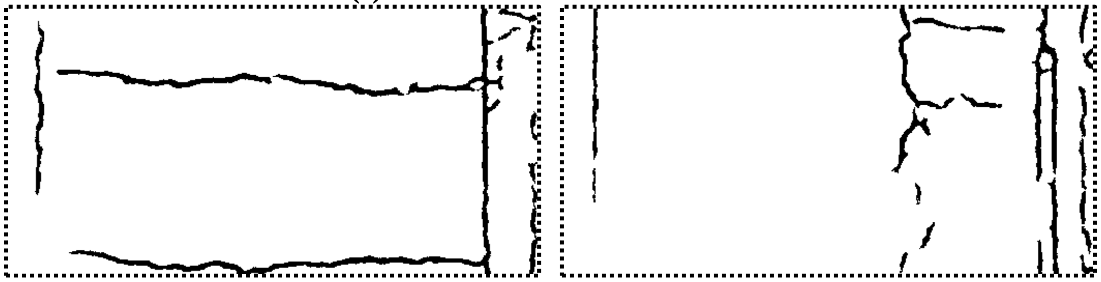
(a) 3D Testing Images



(b) 3D Rendering of Testing Images



(c) Detection Results of CrackNet



(d) Detection Results of CrackNet-V

Figure 5.15. Comparison between CrackNet and CrackNet-V (Config-A) on Wide Cracks

5.4.3 CrackNet-V vs. CrackForest (SVM)

In order to further assess the performance of CrackNet-V, a road crack detection framework called CrackForest is introduced for a comparison study on the public dataset named CFD (Shi et al., 2016). This CFD dataset consists of 118 urban road surface images with manually labeled ground truth contour. 60% of this dataset is used for training and 40% for testing. Since CFD dataset is composed of color images, the original pre-processed layer for 3D surface correction is removed from CrackNet-V architecture. Instead, the images from CFD dataset are gray-scaled and then normalized into the range from -1 to 1. During the testing process, the detected pixels which are no more than five pixels away from the manually labeled pixel are assumed as true positive pixels.

The comparison results between CrackForest (SVM) (Shi et al., 2016) and CrackNet-V (Config-A) are shown in **Fig. 5.16** and summary statistics are in **Table 5.6**. **Fig. 5.16** and **Table 5.6** show that the CrackForest (SVM) generates 89.44% recall and 82.28% precision. CrackNet-V (Config-A) yields 86.03% recall, slightly lower than that in CrackForest. However, the precision of CrackNet-V reached to 92.58%, a much higher value indicating that CrackNet-V has higher accuracy in pixel-level crack classification. As a result, the overall F1 score of CrackNet-V is 89.18% versus 85.71% of CrackForest.

Table 5.6. Comparison between Crackforest and Cracknet-V on CFD Data

Network	**Precision (%)**	**Recall (%)**	**F1 Score (%)**
CrackForest (SVM)	82.28	89.44	85.71
CrackNet-V (Config-A)	92.58	86.03	89.18

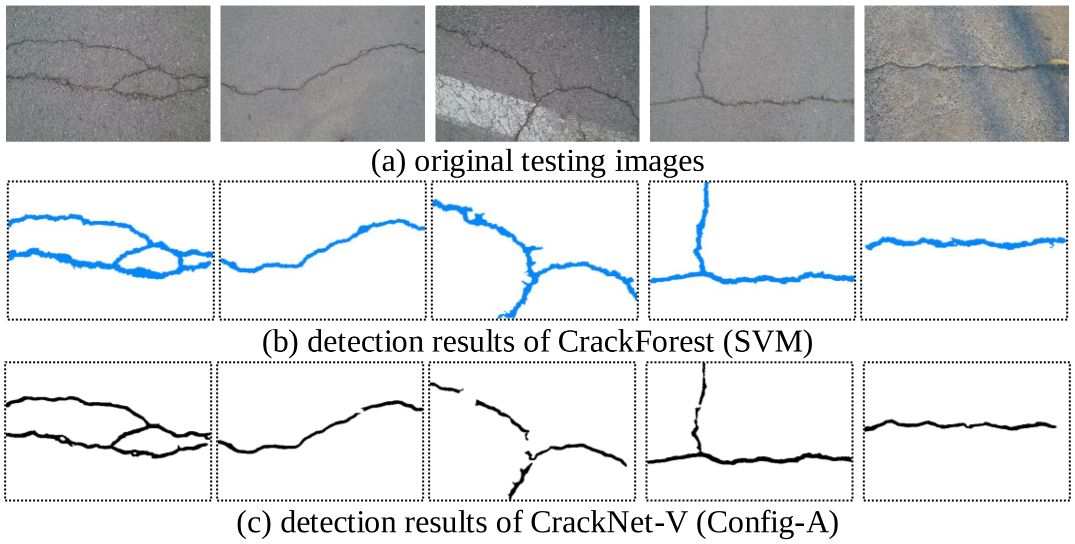

(a) original testing images
(b) detection results of CrackForest (SVM)
(c) detection results of CrackNet-V (Config-A)

Figure 5.16. Comparison between CrackForest (SVM) and CrackNet-V (Config-A) on CFD Data

5.4.4 Network Speed

A speed comparison between CrackNet and CrackNet-V is presented here to evaluate the computational efficiency of CrackNet-V. Both of these two networks are optimized based on CUDA, the parallel computing architecture developed by NVIDIA Company that enables Graphics Processing Units (GPU) to complete highly parallel computations, such as image processing or deep learning tasks. As the number of parameters is much fewer than that of CrackNet, CrackNet-V is more time-efficient.

Table 5.7 summarizes the speeds of CrackNet and CrackNet-V on a single 512×256 image. With single GPU device NVidia GeForce GTX 1080Ti, the time of forward pass and backward pass for CrackNet are 1.21 and 8.25 second respectively, and the training of CrackNet using 512×256 images normally needs four days. With the aid of the same GPU device, by contrast, the time of forward pass and backward pass for CrackNet-V are 0.33 and 2.28 second respectively. As a result, the training time takes only one day. Generally, CrackNet-V is roughly 3-4 times faster than CrackNet. Such improved computational efficiency of CrackNet-V is beneficial for developing real-time applications in the future.

Table 5.7. Average Network Speeds on Single 512×256 Image

Network	Forward-Prop Speed (sec.)	Backward-Prop Speed (sec.)	Training Time (day)
CrackNet	1.21	8.25	4
CrackNet-V	0.33	2.28	1

5.5 Summary

A CNN-based framework named "CrackNet-V" is proposed and developed for pixel-level automated crack detection on 3D asphalt pavements. CrackNet-V does not use any pooling layers so that pixel-perfect accuracy can be retained. All filters used at the first six convolutional layers are set to small size of 3×3 for effective feature extraction while decreasing trainable parameters. In addition, the successive deep convolutional layers used in CrackNet-V also introduce plenty of non-linear transformations to enhance network performance. To share the network weights at each pixel, median filter based pre-processing layer is proposed to eliminate the negative impacts caused by pavement unevenness. Moreover, the activation unit named "Leaky Rectified Tanh" is proposed to minimize the convolutional differences among cracks at varying depths. It is demonstrated that the proposed Leaky Rectified Tanh yields more accurate detection on shallow cracks than that based on the traditional Leaky ReLU. As a consequence, CrackNet-V is able to achieve 87.12% F1-score on test set, which slightly outperforms its predecessor named "CrackNet". Besides, CrackNet-V also reaches 89.18% F1-score on a public urban road data set, surpassing the traditional SVM-based machine learning approach. It is noticeable that CrackNet-V only introduces 64,113 trainable parameters, resulting in fast processing speed of 0.33 second per image. Such improved computational efficiency can enhance the application of CrackNet-V in the planned real-time processing in the future.

Although CrackNet-V is effective for most 3D asphalt images, its performance may suffer due to the limitations caused by the extremely small size of filters. Particularly, the detection accuracy on wide cracks are still not satisfactory. Future developments can be conducted to enlarge the receptive fields by stacking more 3×3 convolutional layers or using other means. In addition, since there are only two targets (i.e. cracks and non-cracks) annotated in the training images, the features of other pavement patterns (i.e. well lids, joints, markers, etc.) are not able to be learnt automatically by the network, resulting in false cracking detections caused by interferences from other pavement patterns. In the future, multiple targets need to be annotated in the standard image library, so that deep learning network can be trained for multiple surface distress features.

Chapter 6 CONCLUSIONS AND FUTURE WORK

6.1 Conclusions

With the explosive increase in pavement surface imaging data size, pavement survey based on big data becomes increasingly critical for pavement management activities. Pavement surface visualization and automated crack detection are two vital tasks for pavement data interpretation. Using 1-mm automated pavement imaging data, this dissertation investigates available automated algorithms as well as their modifications for both large-scale pavement visualization and advanced cell-based and pixel-level crack detections.

Chapter 3 proposes the Geometry Clipmap as an effective approach to visualizing long pavement surfaces in 3D environment for simulation of field survey. Geometry Clipmap utilizes nested grids based on level-of-details concept to reduce the complexity of 3D pavement model, while still retaining the highest resolution of original pavement data for detailed surface inspection. In addition, the 2D projection of view frustum is proposed as an alternative method for save computer resources by removing redundant geometrical blocks. Furthermore, vertex normal computation based on virtual information is applied to smooth visual effects for realistically reconstruction of actual pavement surface conditions. In the preliminary system based on Geometry Clipmap, a virtual ruler is developed to measure the size of surface distresses or any inspector-interested-regions. The significant conveniences of pavement survey based on large-scale 3D environment can improve the efficiency to make decisions on pavement management activities.

Chapter 4 investigates the Convolutional Neural Network (CNN) for automated pavement crack recognition using 3D pavement surface imaging data. CNN utilizes convolutional processes to achieve a variation of traditional multi-layer perceptron (MLP) for feature extractions and object classifications. In this dissertation, a standard pavement image library containing over 6,000 samples with manually labeled cracks is established. It is demonstrated that CNN is capable of automatically learning meaningful features from manually annotated images for cell-based pavement crack recognition. In addition, the performance of CNN for surface damage detection is reasonable and consistent over a large data set. However, since CNN is specially designed for cell-based image analysis, the accuracy level of automated crack detection is based on image cells, which is not able to satisfy practical requirements for pavement management activities. Thus, further modifications of CNN model are studied and implemented in Chapter 5 to explore an innovative framework that replaces the cell-based crack perception with techniques for the pixel-level.

Chapter 5 proposes a novel CNN-based model named “CrackNet-V” as a fast and effective approach for the automated pixel-level crack detection. Different from the traditional CNN concept, CrackNet-V shares network weights at each pixel to recognize cracking pixels from the input image. The sizes of all filters in CrackNet-V are set as 3×3 to extract available features while maximally reducing computational needs. In addition, a well-designed pre-processing layer is proposed to eliminate the adverse impacts caused by pavement unevenness, while retaining the meaningful crack characteristics for enhancing feature extractions. Moreover, a new activation unit called “Leaky Rectified Tanh” is created to minimize the convolutional differences between shallow cracks and deep cracks, so that cracks at varying depths can be perceived consistently. Furthermore, to prevent the gradient explosion during network training process, an effective strategy for sensitivity regularization at the output layer is recommended to balance the sample numbers of cracking and non-cracking pixels. As a result, CrackNet-V is able to achieve the robust performance of crack detection at pixel-level (87.12 in F1-score) in a short time (0.33 second per image). The performance and efficiency advantages of CrackNet-V can be further exploited for future real-time processing during automated pavement data collection.

6.2 Future Work

Geometry Clipmap is designed as a nested structure for rendering quadrate shape of terrain instead of rectangular pavement surface. In other words, redundant geometrical regions outside from long pavement surfaces based on Geometry Clipmap can be removed by other methods for saving more computational resources. In addition, the preparation of pyramidal data consumes huge amount of time before 3D rendering implementation. There should be some rooms for improvement in this type of hierarchical data preparation. Further, other terrain visualization techniques (e.g. Quadtree, Chunked LOD, GPU Terrain, etc.) need to be investigated to seek the best algorithm for long pavement rendering.

Even though CrackNet-V reveals remarkable potentials of deep learning on pixel-level asphalt crack detection, the performance on other types of pavement surfaces, especially on concrete pavements with joints, still requires further development and improvements. In addition, CrackNet-V loses a certain of accuracy on wide crack detection due to a series of extremely small filter sizes. Furthermore, due to only two targets annotated in pavement surface images, the features of other surface patterns (e.g. joints, well lids, potholes, etc.) are not able to be learnt automatically and effectively, resulting in lack of distinctions between cracks and other surface patterns. In other words, the standard pavement image library may require multiple clean labels (cracking and other distresses) for multiple objects recognition, so that the false-detections of pavement cracks caused by other surface patterns can be reduced.

REFERENCES

Abas, F.S. and Martinez, K. (2003) “Classification of painting cracks for content-based analysis.” In SPIE, Annual Symposium Electronic Imaging: Machine Vision Applications in Industrial Inspection XI, Vol.5011, pp.149–160.

Abdelaty, A., Jeong, H.D., Smadi, O.G., and Gransberg, D.D. (2015). “Iowa Pavement Asset Management Decision-Making Framework.” InTrans Project Reports, 143. Online available at: http://lib.dr.iastate.edu/cgi/viewcontent.cgi?article=1142&context=intrans_re¬p¬ort¬s. Accessed Dec 22, 2017.

American Association of State Highway and Transportation Officials (AASHTO). (1985). “Guidelines on Pavement Management.” American Association of State Highway and Transportation Officials, Washington D.C.

Assarsson U., and Moller T. (2000). “Optimized View Frustum Culling Algorithms for Bounding Boxes.” Journal of Graphic Tools, 5(1), 9-22.

Attoh-Okine, N.O. and Adarkwa, O. (2013). “Pavement Condition Surveys – Overview of Current Practices.” Project Report for Delaware Department of Transportation.

Brettell, N., and Mukundan, R. (2005). “Terrain Rendering Using Geometry Clipmaps.” Technical Report, University of Canterbury, NZ.

Bureau of Transportation Statistics. (2012). “Transportation Statistics Annual Report 2012.” Research and Innovative Technology Administration, U.S. Department of Transportation.

California Department of Transportation (CDOT). (2013). “2013 State of the Pavement Report based on the 2013 Pavement Condition Survey.” California Department of Transportation, Division of Maintenance, Pavement Program.

Carvalho, R., Stubstad, R., Briggs, R., Selezneva, O., Mustafa, E., and Ramachandran, A. (2012). “Simplified Techniques for Evaluation and Interpretation of Pavement Deflections for Network-Level Analysis – Chapter 2.” US Department of Transportation, Federal Highway Administration (FHWA). Online available at: https://www.fhwa.dot.gov/publications/research/infrastructure/pavements/ltpp/12023/002.cfm. Accessed Dec 14, 2017.

Cha Y.-J., Choi, W., and Oral, B. (2017). “Deep Learning-Based Crack Damage Detection Using Convolutional Neural Networks.” Computer-Aided Civil and Infrastructure Engineering, vol.32, pp.361-378.

Chambon, S., Subirats, P., and Dumoulin, J. (2009). “Introduction of A Wavelet Transform Based on 2d Matched Filter in A Markov Random Field for Fine Structure Extraction: Application on Road Crack Detection.” In IS&T/SPIE Electronic Imaging-Image Processing: Machine Vision Applications II.

Chaudhuri, S., Chatterjee, S., Katz, N., Nelson, M., and Goldbaum, M. (1989). “Detection of Blood Vessels in Retinal Images Using Two-dimensional Matched Filters.” IEEE Transactions on Medical Imaging, 8(3):263–269.

Chou, J. and Cheng, H.D. (1994). “Pavement Distress Classification Using Neural Networks.” In IEEE Transactions on Systems, Man and Cybernetics, vol.1, pp.397–401.

Clark, J. H. (1976). “Hierarchical geometric models for visible surface algorithms” Communications of the ACM. New York, USA. 19(10), pp 547-554.

Coenen, T.B.J. and Golroo, A. (2017). “A Review on Automated Pavement Distress Detection Methods.” Journal of Cogent Engineering, 4(1).

Coster, M. and Chermant, J.-L. (2001). “Image analysis and mathematical morphology for civil engineering materials.” Cement and Concrete Composites, 23(2), pp.133–151.

Cybenko, G.V. (2006). "Approximation by Superpositions of a Sigmoidal function". In van Schuppen, Jan H. Mathematics of Control, Signals, and Systems. Springer International. pp. 303–314.

Daniel, A., and Preeja, V. (2014). “A novel technique for automatic road distress detection and analysis.” Int. J. Comput. Appl., 101(10), 0975–8887.

Daugman, J. G. (1985). "Uncertainty relation for resolution in space, spatial frequency, and orientation optimized by two-dimensional visual cortical filters." Journal of the Optical Society of America A, vol. 2, no. 7, pp. 1160–1169.

Dicker, G.R. (2003). “The Future of 3D Graphics Technology: Will the Movies Maintain Their Lead on the Desktop?” Stanford University. Online available at: https://web.stanford.edu/group/htgg/sts145papers/gdicker_2003_1.pdf. Accessed Jan 2, 2018.

Duan, Y., Lv, Y., Liu, Y.L., and Wang, F.Y. (2016). “An Efficient Realization of Deep Learning for Traffic Data Imputation.” Transportation Research Part C: Emerging Technologies, vol.72, pp.168-181.

Elbehiery, H., Hefnaway, A., and Elewa, M. (2005). "Surface Defects Detection for Ceramic Tiles Using Image Processing and Morphological Techniques." Proceedings of World Academy of Science, Engineering and Technology (PWASET), vol.5, pp.158-162.

Fawcett, T. (2006). "An Introduction to ROC Analysis." Pattern Recognition Letter, 27(8), 861-874.

Federal Highway Administration (FHWA). (2006). "Long-Term Pavement Performance Maintenance and Rehabilitation Data Collection Guide." Turner-Fairbank Highway Research Center, Public No. FHWA-HRT-06-068. Online available at: https://www.fhwa.dot.gov/publications/research/infrastructure/pavements/ltpp/06068/06068.pdf. Accessed Dec 22, 2017.

Federal Highway Administration (FHWA). (2017). "Employment Impacts of Highway Infrastructure Investment." Online available at: https://www.fhwa.dot.gov/policy/otps/pubs/impacts. Accessed Dec 20, 2017.

Garmin Ltd. "What is 3D Terrain View?" Online available at: https://support.garmin.com/support/searchSupport/case.faces?caseId=%7B2a5c7c30abc7-11df-55a0-0000000000¬00¬%¬7D. Accessed Jan 2, 2018.

Georgieva, K., Koch, C., and Konig, M. (2015). "Wavelet transform on multi-GPU for real-time pavement distress detection." Comput. Civ. Eng., pp.99–106.

Glorot, X. and Bengio, Y. (2010). "Understanding the Difficulty of Training Deep Feedforward Neural Networks." In Proceedings of the 13th International Conference on Artificial Intelligence and Statistics (AISTATS), pp.249-256.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). "Deep Learning." Published by MIT Press. Online available at: http://www.deeplearningbook.org.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). Deep Learning, Published by MIT Press, online available at: http://www.deeplearningbook.org.

Google Inc. “Google Earth Learn Page.” Online available at: http://www.google.com/earth/learn/. Accessed Jan 2, 2018.

Hagan, M.T., Demuth, H.B., Beale, M.H., and Jesús, O.D. (2014). “Neural Network Design 2nd Edition.” Published by Martin Hagan, United States, ISBN 10: 0971732116 / ISBN 13:9780971732117.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). “Deep Residual Learning for Image Recognition.” The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), online available at: https://arxiv.org/abs/1512.03385.

Highway Research Board. (1972). “NCHRP Synthesis of Highway Practice 14: Skid Resistance.” Highway Research Board, National Research Council, Washington, D.C.

Hinton, G.E., Osindero, S. and Teh, Y.-W. (2006). “A fast learning algorithm for deep belief nets.” Journal of Neural Computation vol.18, pp.1527–1554.

Hsu, C.-J., Chen, C.-F., Lee, C., and Huang, S.-M. (2001). “Airport pavement distress image classification using moment invariant neural network.” In Asian Conference on Remote Sensing, vol.1, pp.216–220.

Huang, T., Yang, G., and Tang, G. (1979). “A Fast Two-Dimensional Median Filtering Algorithm.” Journal of IEEE Trans. Acoust., Speech, Signal Processing, vol. 27, no. 1, pp. 13-18.

Iyer, S. and Sinha, S.K. (2005). “A Robust Approach for Automatic Detection and Segmentation of Cracks in Underground Pipeline Images.” Image and Vision Computing, 23(10), pp.921–933.

Jean, S., Cho, K., Memisevic, R., and Bengio, Y. (2015). “On using very large target vocabulary for neural machine translation.” Proc., the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing, pp.1-10.

Kaseko, M.S. and Ritchie, S.G. (1993). “A Neural Network-based Methodology for Pavement Crack Detection and Classification.” Annual Meeting of Transportation Research Record, 1(1), pp.275–291.

Katakam, N. (2009). “Pavement Crack Detection System through Localized Thresholing.” M.S. Thesis, Univerisity of Toledo, Toledo, OH.

Ke, J., Zheng, H., Yang, H., and Chen, X. (2017). “Short-term Forecasting of Passenger Demand under On-demand Ride Services: A Spatio-Temporal Deep Learning Approach.” Transportation Research Part C: Emerging Technologies, vol.85, pp.591-608.

Kim, D. (2004). “3D Visual Urban Simulation: Methods and Applications.” Accepted for Korean Local Administration Review. Online available at: https://pdfs.semanticscholar.org/14b5¬/¬¬31798a4e9cbb1f23c864c290cec8e57966b9.pdf. Accessed Jan 2, 2018.

Koch, C. and Brilakis, I. (2011). “Pothole Detection in Asphalt Pavement Images.” Journal of Advanced Engineering Informatics, 25(3), pp.507-515.

Koutsopoulos, H.N. and Downey, A.B. (1993). “Primitive-Based Classification of Pavement Cracking Images.” Journal of Transportation Engineering, 119(3), pp.402-418.

Kreinovich, V.Y. (1991). “Arbitrary Nonlinearity Is Sufficient to Represent All Functions by Neural Networks: A Theorem.” Journal of Neural Networks, vol. 4, no. 3, pp. 381-383.

Krizhevsky, A., Sutskever, I., and Hinton, G. (2012). “ImageNet classification with deep convolutional neural networks.” Advances in Neural Information Processing Systems, 25, pp.1097-1105.

Krogh, A., and Hertz, J.A. (1992). “A Simple Weight Decay Can Improve Generalization.” Advances in Neural Information Processing Systems, vol.4, pp.951-957.

Laurent, J., Hebert, J.F., Lefebvre, D., and Savard, Y. (2012). “Using 3D Laser Profiling Sensors for the Automated Measurement of Road Surface Conditions.” 7th RILEM International Conference on Cracking in Pavement, pp.157-167.

Laurent, J., Lefebvre, D., and Samson, E. (2008). “Development of A New 3D Transverse Laser Profiling System for the Automatic Measurement of Road Cracks.” Symposium on Pavement Surface Characteristics, 6th, Portoroz, Slovenia.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). “Deep Learning.” Journal of Nature, 521(7553), 436-444.

LeCun, Y., Boser, B., Denker, J.S., Henderson, D., Howard, R.E., Hubbard, W., and Jackel, L.D. (1990). “Handwritten Digit Recognition with a Back-Propagation Network.” Journal of Advances in Neural Information Processing Systems (NIPS), vol.2, pp.396-404.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998a). “Gradient-Based Learning Applied to Document Recognition.” Proc., IEEE, 86(11), 2278-2324.

LeCun, Y., Bottou, L., Orr, G., and Müller, K. (1998b). “Efficient BackProp.” pp.9-50 from Book of Neural Networks: Tricks of the Trade, Published by Springer-Verlag, London, UK.

Ledig, C., Theis, L., Huszar, F., Caballero, J., Cunningham, A., Acosta, A., Aitken, A., Tejani, A., Totz, j., Wang, Z., and Shi, W. (2016). “Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network.” Journal of CoRR, online available at: https://arxiv.org/abs/1609.04802.

Lei, J., Wang, E., Zeng, J., Wang, W., and Wu, J. (2014). "Research of acquisition method for pavement surface texture based on photometric stereo techniques." 17th IEEE International Conference on Intelligent Transportation Systems, ITSC 2014, pp.1596-1601.

Liang, S.Q. and Sun, B.C. (2010). “Using wavelet technology for pavement crack detection.” Proc., 2010 Int. Conf. of Logistics Engineering and Management, vol.387, ASCE, Reston, VA, pp.2482–2487.

Losasso, F., and Hoppe, H. (2004). “Geometry Clipmaps: Terrain Rendering Using Nested Regular Grids.” ACM Trans. Graphics (Proceedings of SIGGRAPH 2004). New York, USA. 23(3), pp 769–776.

Luebke, D., and Hallen, B. (2001). “Perceptually Driven Simplification for Interactive Rendering.” In Proceedings of the 12th Eurographics Workshop on Rendering Techniques. London, United Kingdom. pp 223-234.

Luebke, D., Reddy, M., Cohen, J. D., Varshney, A., Watson, B., Huebner, R. (2003). Level of Detail for 3D Graphics, 1st Ed., Morgan Kaufmann, San Francisco.

Maas, A.L., Hannun, A.Y., and Ng, A.Y. (2013). “Rectifier Nonlinearities Improve Neural Network Acoustic Models.” in Proceedings of the 30th ICML, vol.28.

Mancini, A., Malinverni, E.S., Frontoni, E., and Zingaretti, P. (2013). “Road Pavement Crack Automatic Detection by MMS Images.” 21st Mediterranean Conference on Control and Automation, pp. 25-28, June 2013.

Marčelja, S. (1980). "Mathematical Description of the Responses of Simple Cortical Cells." Journal of the Optical Society of America. vol. 70, no. 11, pp. 1297–1300.

Mathavan, S., Kamal, K., and Rahman, M. (2015). “A Review of Three-Dimensional Imaging Technologies for Pavement Distress Detection and Measurements.” IEEE Transactions on Intelligent Transportation Systems, 16(5), pp.2353-2362.

Max N. (1999). “Weights for Computing Vertex Normals from Facet Normals.” Journal of Graphics Tools, 18(1), 1-6.

McKesson, J. L. (2012). “Learning Modern 3D Graphics Programming.” Online available at: https://paroj.github.io/gltut/. Accessed Jan 25, 2018.

Metro Nashville. (2018). “Long Range Paving Plan – Chapter 3: Pavement Management Data.” Metropolitan Government of Nashville & Davidson County, Tennessee.

Minsky, M. and Papert, S. (1969). “Perceptrons: An Introduction to Computational Geometery, 1st Ed.” Published by MIT Press, Cambridge MA, USA. ISBN: 0-262-63022-2.

Moreno, F.A., Gonzalez-Jimenez, J., Blanco, J.L., and Esteban, A. (2013). “An Instrumented Vehicle for Efficient and Accurate 3D Mapping of Roads.” Journal of Computer-Aided Civil and Infrastructure Engineering, 28(6).

Moussa, G., and Hussain, K. (2011). “A new technique for automatic detection and parameters estimation of pavement crack.” Proc., 4th Int. Multi-Conf. on Engineering and Technological Innovation, Orlando, FL, pp.11–16.

National Research Council (U.S.). (1970). Standard Nomenclature and Definitions for Pavement Components and Deficiencies. Published by Highway Research Board, National Academy of Sciences, Washington DC, ISBN 10: 0309018021.

Nguyen, T. S., Avila, M., and Begot, S. (2009). “Automatic detection and classification of defect on road pavement using anisotropy measure.” Proc., 17th European Signal Processing Conf., IEEE, New York, 617–621.

Oliveira, H. and Correia, P.L. (2009). “Automatic road crack segmentation using entropy and image dynamic thresholding.” In European Signal Processing Conference.

Oregon Department of Transportation (ODOT). (2010). “Pavement Distress Survey Manual.” Oregon Department of Transportation. 355 Capitol Street NE. MS 11, Salem, OR, 97301-3871.

PaddlePaddle. (2014). ‘Image Classification.’ Online available at: http://paddlepaddle.org/docs/develop/book/03.image_classification/index.html. Accessed Jan 26, 2018.

Pierce, L.M., McGovern, G., Zimmerman, K.A. (2013). “Practical Guide for Quality Management of Pavement Condition Data Collection.” US Department of Transportation, Federal Highway Administration (FHWA).

Polson, N.G., and Sokolov, V.O. (2017). “Deep Learning for Short-Term Traffic Flow Prediction.” Transportation Research Part C: Emerging Technologies, vol.79, pp.1-17.

Pulli, K., Aarnio, T., Miettinen, V., Roimela, K., and Vaarala, J. (2008). Mobile 3D Graphics with OpenGL ES and M3G. Published by Morgan Kaufmann, ISBN: 978-0-12-373727-4.

Rosenblatt, F. (1958). “The perceptron: a probabilistic model for information storage and organization in the brain.” Journal of Psychological Review, vol.65, pp.386-408.

Rumelhart, D.E., Hinton, G.E. and Williams, R.J. (1986). “Learning representations by back-propagating errors.” Journal of Nature, vol.323, pp.533–536.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A.C., and Fei-Fei, L. (2014). “Imagenet large scale visual recognition challenge.” International Journal of Computer Vision (IJCV), online available at: https://arxiv.org/abs/1409.0575.

Ruzinoor, C. M., Shariff, A. R. M., Pradhan, B., Ahmad, M. R., Rahim, M. S. M. (2012). “A review on 3D terrain visualization of GIS data: techniques and software.” Geo-spatial Information Science, 15(2), 105-115.

Saleh, M., Abbott, A.L., and Flintsch, G.W. (2016). “3D Pavement Surface Spherical Representation and Reconstruction.” Transportation Research Board 95th Annual Meeting, Washington DC, United States.

Sayers, M.W., Gillespie, T.D., and Paterson, D.O. (1986). “Guidelines for the Conduct and Calibration of Road Roughness Measurements.” World Bank Technical Paper No.46, the World Bank, Washington DC.

Sayers, M.W., Gillespie, T.D., and Queiroz, C.A.V. (1986). “The International Road Roughness Experiment: Establishing Correlation and a Calibration Standard for Measurements.” World Bank Technical Paper No.45, the World Bank, Washington DC.

Shallue, C.J. and Vanderburg, A. (2017). "Identifying Exoplanets with Deep Learning: A Five Planet Resonant Chain around Kepler-80 and an Eighth Planet around Kepler-90." Online available at: https://arxiv.org/abs/1712.05044.

Shi, Y., Cui, L., Qi, Z., Meng, F., and Chen, Z. (2016). "Automatic Road Crack Detection Using Random Structured Forests." Journal of IEEE Transactions on Intelligent Transportation Systems, vol. 17, no. 12, pp. 3434-3445.

Shrivastava, A., Pfister, T., Tuzel, O., Susskind, J., and Wang, W., and Webb, R. (2016). "Learning from Simulated and Unsupervised Images through Adversarial Training." CoRR, online available at: https://arxiv.org/abs/1612.07828.

Silver, D., Huang, A., Maddison, C.J., Guez, A., Sifre, L., Driessche, G.v.d., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, and T., Hassabis, D. (2016). Mastering the game of Go with deep neural networks and tree search. Nature Journal, Vol. 529(7587), pp.484-489.

Simonyan, K., and Zisserman, A. (2014). "Very deep convolutional networks for large-scale image recognition." Proc., International Conference on Learning Representations, online available at: https://arxiv.org/abs/1409.1556.

Sivaneswaran, N., Pierce, L.M., and Mahoney, J.P. (2004). "Transition from Manual to Automated Pavement Condition Surveys: Washington State's Experience." 6th International Conference on Managing Pavements.

Solla, S.A., Levin, E., and Fleisher, M. (1988). "Accelerated Learning in layered neural networks." Complex Systems, vol.2, pp.625-640.

South Dakota Department of Transportation (SD DOT). (2017). “Visual Distress Survey Manual.” South Dakota Department of Transportation. Becker-Hansen Building 700 E. Broadway Ave., Pierre, SD 57501.

Subirats, P., Dumoulin, J., Legeay, V., and Barba, D. (2006). “Automation of Pavement Surface Crack Detection with a Matched Filtering to Define the Mother Wavelet Function Used.” In European Signal Processing Conference.

Sullivan, D.E. (2006). “Materials in Use in U.S. Interstate Highways.” Fact Sheet 2006-3127 from U.S. Department of Interior, U.S. Geological Survey. Online available at: https://pubs.usgs.gov/fs/2006/3127/2006-3127.pdf. Accessed Nov 30, 2017.

Sutskever, I., Vinyals, O., and Le. Q.V. (2014). “Sequence to sequence learning with neural networks.” Proc., Advances in Neural Information Processing System, Vol.27, pp.3104-3112.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2014). “Going deeper with convolutions.” Journal of CoRR, online available at: https://arxiv.org/abs/1409.4842.

Tanaka, N. and Uematsu, K. (1998). “A Crack Detection Method in Road Surface Images Using Morphology.” In Workshop on Machine Vision Applications, pp.154–157.

Teomete, E., Amin, V.R., Ceylan, H., and Smadi, O. (2005). “Digital Image Processing for Pavement Distress Analyses.” Proc., the 2005 Mid-Continent Transportation Research Symposium.

Timm, D.H. and McQueen, J.M. (2004). “A Study of Manual vs. Automated Pavement Condition Surveys.” Alabama Department of Transportation, Montgomery, AL, Highway Research Center, Auburn University, Alabama 36849-5337.

U.S. Army. (1982). “Pavement Maintenance Management, Technical Manual TM 5-623.” Headquarters, Department of the Army, Washington, D.C.

Wang, K. C. P., Gong, W., Elliott, R. P. (2004). “A Feasibility Study on Data Automation for Comprehensive Pavement Condition Survey.” In proceedings of the 6th International Conference on Managing Pavements: The Lessons, The Challenges, The Way Ahead. Brisbane, Queensland, Australia.

Wang, K.C.P. and Smadi, O. (2011). “Automated Imaging Technologies for Pavement Distress Surveys.” Transportation Research Board of the National Academies, Washington, D.C.

Wang, K.C.P., Gong, W., Tracy, T., and Nguyen, V. (2011). “Automated Survey of Pavement Distress Based on 2D and 3D Laser Images.” Mack-Blackwell Transportation Center, University of Arkansas, Fayetteville.

Wang, K.C.P., Li, G., and Gong, W. (2007). “Wavelet-based Pavement Distress Image Edge Detection with A Trous Algorithm.” In Annual Meeting of Transportation Research Record, vol.2024, pp.73–81.

Wang, K.C.P., Li, Q.J., Yang, G., Zhan, Y., and Qiu, Y. (2015). “Network Level Pavement Evaluation with 1 mm 3D Survey System.” Journal of Traffic and Transportation Engineering, 2(6), pp.391-398.

Wang, K.C.P., Li, Q.J., Yang, G., Zhan, Y., and Qiu, Y. (2015). “Network Level Pavement Evaluation with 1 mm 3D Survey System.” Journal of Traffic and Transportation Engineering, 2(6), pp.391-398.

Wolff D. (2013). OpenGL 4 Shading Language Cookbook, 2nd Ed., Packt Publishing Ltd, Birmingham.

Zhang, A., Li, Q., Wang, K.C.P., and Qiu, S. (2013). “Matched Filtering Algorithm for Pavement Cracking Detection.” Transp. Res. Rec., vol.2367, pp.30–42.

Zhang, A., Wang, K. C. P. & Ai, C. (2017a). “3D shadow modeling for detection of descended patterns on 3D pavement surface.” Journal of Computing in Civil Engineering, 31(4), 04017019.1-04017019.13.

Zhang, A., Wang, K.C.P., Ji, R. and Li, Q. (2016a). “Efficient system of cracking-detection algorithms with 1-mm 3D-surface models and performance measures.” Journal of Computing in Civil Engineering, 30(6), 04016020.1-04016020.16.

Zhang, A., Wang, K.C.P., Li, B., Yang, E., Dai, X., Peng, Y., Fei, Y., Liu, Y., Li, J.Q., Chen, C. (2017b). “Automated Pixel-Level Pavement Crack Detection on 3D Asphalt Surfaces Using a Deep-Learning Network.” Computer-Aided Civil and Infrastructure Engineering, 32(10), pp.805-819.

Zhang, C. and Elaksher, A. (2012). “An Unmanned Aerial Vehicle-based Imaging System for 3D Measurement of Unpaved Road Surface Distresses.” Journal of Comput.-Aided Civil Infrastruct. Eng., 27(2), pp.118-129.

Zhang, L., Yang, F., Zhang, Y.D., and Zhu, Y.J. (2016b). “Road crack detection using deep convolutional neural network.” IEEE Conference, ISSN: 2381-8549.

Zhou, J., Huang, P., and Chiang, F.-P. (2005). “Wavelet-based Pavement Distress Classification.” Journal of the Transportation Research Board (Transportation Research Record), vol.1940, pp.89–98.

VITA

YUE FEI

Candidate for the Degree of

Doctor of Philosophy

DISSERTATION: VISUALIZATION AND INTELLIGENT SOLUTIONS FOR BIG PAVEMENT DATA

Major Field: Civil Engineering

Biographical:

Education:

Completed the requirements for the Doctor of Philosophy in Civil Engineering at Oklahoma State University, Stillwater, Oklahoma in July, 2018.
Completed the requirements for the Master of Science in Civil Engineering at Oklahoma State University, Stillwater, Oklahoma in May, 2015.
Completed the requirements for the Bachelor of Science in Computer and Communication Engineering at Southwest Jiaotong University, Chengdu, China in June, 2011.

Experience:
May 2013-present: Research Assistant, Oklahoma State University