DEVELOPING A CRACK INSPECTION ROBOT FOR BRIDGE DECK MAINTENANCE

By

RONNY SALIM LIM

Bachelor's Degree
Pelita Harapan University
Jakarta, Indonesia
2008

Submitted to the Faculty of the
Graduate College of
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
July, 2011

DEVELOPING A CRACK INSPECTION ROBOT FOR BRIDGE DECK MAINTENANCE

Thesis Approved:

Dr. Weihua Sheng

Thesis Advisor

Dr. Gary G. Yen

Committee Member

Dr. Damon Chandler

Committee Member

Dr. Mark E. Payton

Dean of the Graduate College

ACKNOWLEDGMENTS

First and foremost, I would like to thank my advisor Dr. Weihua Sheng, who has supported me throughout my master thesis with his patience and knowledge. I appreciate his guidance, inspiration, motivation and encouragement. I would have been lost without him. Besides my advisor, I would like to thank my thesis committee member: Dr. Gary G. Yen and Dr. Damon Chandler, for their comments and questions to improve this master thesis.

I gratefully acknowledge Hung M. La for his advice and ideas, which give a lot of contributions in this research. Since the first time I started my master studies, he has given me a lot of inspiration and motivation in various ways for this research. Hung, thank you in every possible way and I hope to keep up our collaboration in the future.

I gratefully thank to my fellow lab members, Chun Zhu, Gang Li and Anand Thobbi. They provided me with many tools to aid in the completion of this thesis. I am much indebted to them for their support. Many thanks to Shawn Harmon, Jianhao Du, and Jeremy Paul Evert for such valuable comments.

Lastly, and most importantly, I wish to thank my mother and my sister, Sung Fu Sing and Rosalina Salim. They supported me and loved me. To them I dedicate this thesis.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

This chapter introduces one of the important problems in the public transportation system-bridge deck inspection. Currently, the bridge deck inspection is performed by a human inspector. We propose a system which consists of a mobile robot to replace the human inspector because of the capability of the mobile robot is better than the human inspector. To implement this, we discuss several challenges that we need to address in this research. We also discuss some literature on crack inspection that are related to our work.

1.1 Motivation

For many engineered transportation structures, including civil, mechanical and aerospace structures, timely awareness of their structural health can prevent functional failures which may lead to catastrophic consequences. On August 1, 2007, the collapse of the I-35W Mississippi river bridge (See Figure 1.1) that carried Interstate 35W across the Mississippi river in Minneapolis left 13 dead and more than 100 injured [6], not to mention its big impact on

Figure 1.1: I-35W bridge over the Mississippi river in Minneapolis collapsed [1].

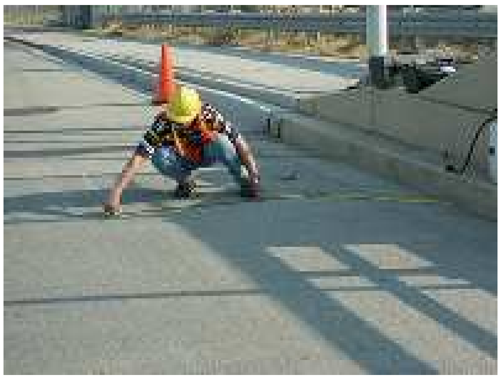


Figure 1.2: A human inspector measures cracks for bridge deck maintenance.

the traffic and businesses in the surrounding areas. This accident has clearly demonstrated the catastrophic results of structural failures and the importance of timely awareness of structural health. Bridge decks are typically the elements of first maintenance on a bridge. Since the surface of a bridge deck is exposed to the environment, direct loading from vehicles and exposure to deicing chemicals, constant maintenance is a must. Therefore bridge deck inspection is helpful and can provide owners warning to the future deterioration of the bridge deck. Currently, bridge decks are inspected with very rudimentary methods in the form of visual inspection by a trained engineer as shown in Figure 1.2. While these inspections are useful, they are only useful if the deterioration of the bridge deck is significant. The inspectors usually walk though the bridges and measure the crack locations and crack sizes. This kind of manual approach has the following disadvantages:

1. It is prone to human errors.
2. It has very limited accuracy due to the limited visual capability of human inspectors.
3. It cannot guarantee the full coverage of the whole bridge deck.

Additionally, conducting visual crack inspection of a bridge deck is a dangerous job with passing traffic. Therefore, we need to develop a robotic crack inspection and mapping (ROCIM) system which can conduct accurate assessment of cracking on bridge decks. The ROCIM system will have the following features which outperform the human inspectors:

1. A high resolution camera will be mounted on the mobile robot to achieve high accuracy of crack detection.
2. The crack locations can be accurately determined since the robot can localize itself precisely.
3. Using mobile robots will reduce the chance of the loss of human life.

1.2 Challenges

In the ROCIM system, a mobile robot is utilized to create a two dimensional (2D) map of the bridge deck using a laser sensor while a camera is used to collect images of the bridge deck surface. The collected images are then processed using image processing techniques to detect the cracks. We store the crack locations in the 2D map, which then becomes a crack map. The crack map is useful for bridge deck maintenance in terms of measuring, classifying and monitoring cracks periodically. In order to implement the ROCIM system, there are several challenging problems that we need to address which are listed below:

1. The coordinate transformation. To create a crack map, the crack location has to be plotted in the global coordinate system which is the same coordinate system as the 2D map of the bridge deck. Since the cracks are detected in the image coordinate system, we have to map the crack locations from the image coordinate system to the global coordinate system.
2. The path planning of mobile robot. The path planning should ensure the mobile robot can inspect the whole area of the bridge deck surface in an efficient way. Unlike the typical complete coverage path planning of a vacuum cleaner robot, which uses the robot body as the coverage, the mobile robot in ROCIM system is interested in the union of camera field of view as the coverage, which poses some difficulties in the path planning.

3. The mobile robot localization. To create the 2D map of the bridge deck, the mobile robot needs to know its current position. Robot localization is a challenging problem in robotic research.

1.3 Literature review

Recent years have witnessed growing research interests in structural health monitoring (SHM) [7],[8],[9],[10],[11] for bridges, buildings and other civil infrastructures. Research in structure inspection using robotic devices has resulted in several prototypes. Yu *et al.* [5] presented an auto inspection system using a mobile robot for detecting concrete cracks in a tunnel. Their system consists of a mobile robot, a data storage system and a crack detector. The crack is detected using image processing and the crack information is extracted using Sobel and Laplacian operators [5]. In addition to their system, an illuminator is used to distinguish the crack and non-crack areas. Their mobile robot system has a complex imaging system to reduce noise resulted from the mobility of mobile robot. Sinha *et al.* [12] developed a statistical filter for crack detection in pipes. A two step algorithm is proposed. First, crack features are extracted from the segmented image. Two filter detectors are applied to extract the crack information. Then, both responses are merged to obtain a unique response. Second, apply cleaning and linking operation to form the global cracks. Tung *et al.* [13] proposed the development of a mobile manipulator imaging system for bridge crack inspection. The manipulator system is equipped with a binocular charge coupled devices (CCD) camera. Their system aims to replace human inspectors. Lee *et al.* [14] and Oh *et al.* [15] proposed a bridge inspection system which consists of a specially designed car, a robotic mechanism and a control system for automatic crack detection. Their system measures the length and width of the cracks. The measurement result is stored in the database which is utilized in the bridge management system. Sohn *et al.* [16] proposed an automatic procedure for monitoring crack growth on concrete structures. Their system focuses on quantifying the change of cracks from multi temporal images during the monitoring period. A modified iterated Hough transform (MIHT) is developed to solve a 2D projection between 2D concrete surface and 2D digital image [17]. Ito *et al.* [18] demonstrated an automatic measurement system for concrete block inspection by means of fine crack extraction. Most of these studies classify, measure, and detect cracks. However, none of these crack inspection studies finds the global location of cracks. In this thesis, we develop an overall system for robotic crack inspection. We detect cracks using a high resolution camera. Then we find the location of cracks in the 2D map. To ensure the completeness of the crack map, the inspection has to cover the whole area of the bridge deck surface. In other words, the complete coverage path planning must be applied for the ROCIM system.

There are several related works for complete coverage path planning using mobile robots. Choset *et al.* [19] [20] developed a complete coverage path planning using boustrophedron motion. The environment is decomposed into cells and the back and forth motion is applied for each cell. Two types of decomposition are discussed: boustrophedron decomposition and trapezoidal decomposition. The boustrophedron decomposition gives more efficient coverage path planning than trapezoidal decomposition because it has fewer cells to be covered. Muzaffer *et al.* [21] investigated a genetic algorithm to coverage path planning for mobile robots. The coverage is modeled as a disk which represents the range of sensing devices. The genetic algorithm is used to find an optimum or near optimum path in order to cover the disk at least once. Paulo *et al.* [22] utilized a genetic algorithm to find the most efficient coverage path in a static environment. The proposed algorithm decomposes the region into sub regions to separate the obstacle and non obstacle area. Their algorithm considers two common templates, zigzag and windowing, to cover in the sub region area. The fitness function of the genetic algorithm is defined based on distance and time. Simon *et al.* [23] developed a neural network approach to complete coverage path planning. The dynamic neural activity based on Hodgkin and Huxley's [24] shunting equation is used to develop the complete coverage path planning. The idea is unclean area attracts the mobile robot and the obstacle pushes the mobile robot away. The result shows that the number of robot turns and traveling distance are minimized. Most of these works consider the robot body as the coverage and the camera field of view (FoV) is not studied. In the ROCIM system we are interested in covering the camera FoV. There are challenges in the planning problem due to the configuration of the camera and the mobility of the robot. In addition, we present a solution to this problem based on genetic algorithm. The solution of complete coverage path planning (CCPP) is developed under the condition that the mobile robot knows their position. To solve this localization problem, we use Monte Carlo localization algorithm [25]. We utilize an advanced range navigation laser (ARNL) from the Mobilerobot Inc. [2] to implement the MCL algorithm.

1.4 Organization of the thesis

This thesis is organized as follows. Chapter 2 presents overview of the ROCIM system. The camera calibration is discussed in Chapter 3, which finds the intrinsic and extrinsic parameters that describe the geometric mapping between the 3D world and the 2D image. Chapter 4 presents the algorithm for crack detection. Chapter 5 discusses the complete coverage path planning for the ROCIM system. The path planning is developed using genetic algorithm to ensure the complete coverage of camera field of view (FoV) and to minimize the traveling distance and the number of robot turns. The experiment results for the overall ROCIM system and the complete coverage path planning are discussed in Chapter 6. Finally, conclusions and future work are given in Chapter 7.

CHAPTER 2

ROCIM SYSTEM OVERVIEW

In this chapter, we first introduce the hardware setup of our ROCIM system. The main hardware for the ROCIM system is a mobile robot, which travels in the inspection area and collects images of the bridge deck surface. Then, the main principle of the ROCIM system is presented as follows: navigation, data collection and crack map generation.

2.1 Hardware setup of the ROCIM system

Figure 2.1 shows the overall hardware setup of the ROCIM system which consists of:

- one Pioneer3-DX mobile robot
- one LMS-200 laser sensor
- one Pan-tilt-zoom (PTZ) Canon VC-C50i camera
- one laptop

Inside the mobile robot, there is an on-board computer with limited memory and processor power. Therefore, we utilize a laptop computer to implement complicated algorithms such as cracks detection, localization, and path planning.

The Pioneer3-DX is a two-wheel drive mobile robot and has an on-board computer. The processor of the on-board computer is a Pentium III 533 Mhz with 128 MB RAM and it also has a 44.2368 MHz Renesas SH2 32-bit RISC microprocessor with 32K RAM. The communication between the two processors on this mobile robot is based on a client-server architecture.

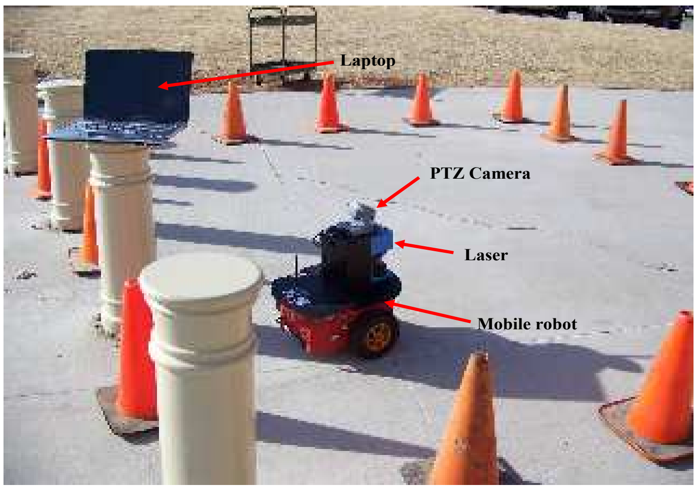


Figure 2.1: ROCIM system in action.

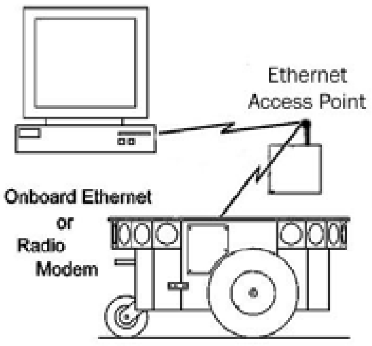


Figure 2.2: Connection setup between the laptop and the mobile robot [2].

The laptop is an HP G62-140US which has a 2.13GHz Intel Core i3-330 Processor and 4GB DDR3 system memory. The laptop and the mobile robot are communicating via wireless technology through Transmission Control Protocol (TCP). The laptop controls the microprocessor by sending TCP data packet to the on-board computer. This TCP data packet is converted into serial data to communicate with the microprocessor. The illustration of the wireless communication is shown in Figure 2.2. In addition, the ethernet access point can also be removed by using a peer to peer wireless ethernet network between the laptop and the on-board computer.

The LMS-200 laser sensor is used to create the 2D map of the bridge deck and localize the robot during the inspection. This sensor has 180^0 bearing angle and is able to estimate a range up to 15 meters [2]. This device connects to the on-board computer through serial communication. Another important device is the PTZ Canon VC-C50i camera. It is a color camera with a resolution of 860×640 and optical-zoom-in of 26. The orientation of this camera can be changed and the ranges of the pan and tilt angles are -100^0 to 100^0, 30^0 to 90^0, respectively. This camera produces high resolution images which can be used for crack detection in concrete environments.

2.2 Principle of the ROCIM system

To conduct bridge deck crack inspection and mapping, as illustrated in Figure 2.3, we will block half of the bridge. The ROCIM system will then be deployed to inspect the blocked half of the bridge. Once completed, the traffic will be switched to the completed half and the other half will be inspected. During the ROCIM operation, the images of the mobile robot will get blurred due to the vibration caused by the passing traffic. To solve this problem, an anti-shock device can be used underneath the camera. Moreover, when the mobile robot collects the images, we assume the mobile robot completely stops in order to capture a clear image.

There are three steps in crack inspection and mapping.

1. Navigation map building: A 2D bridge deck map will be created first, which will be

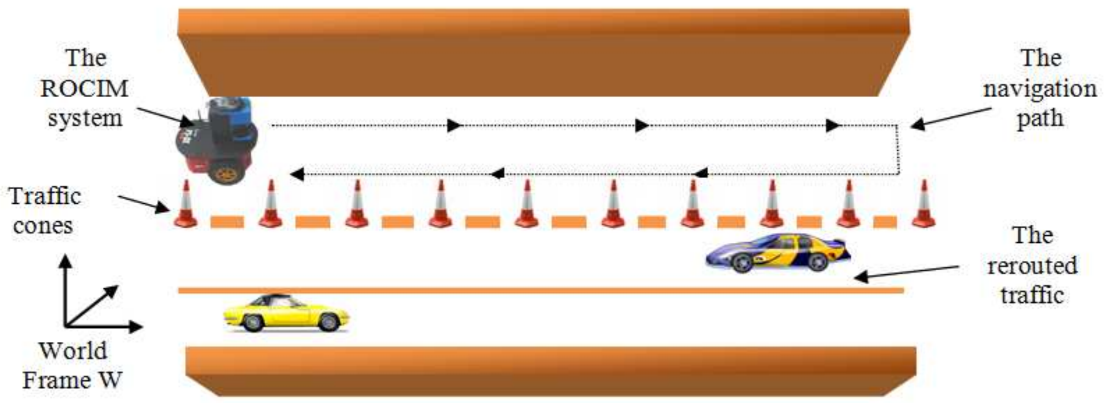


Figure 2.3: Scenario of crack inspection and mapping using the ROCIM system.

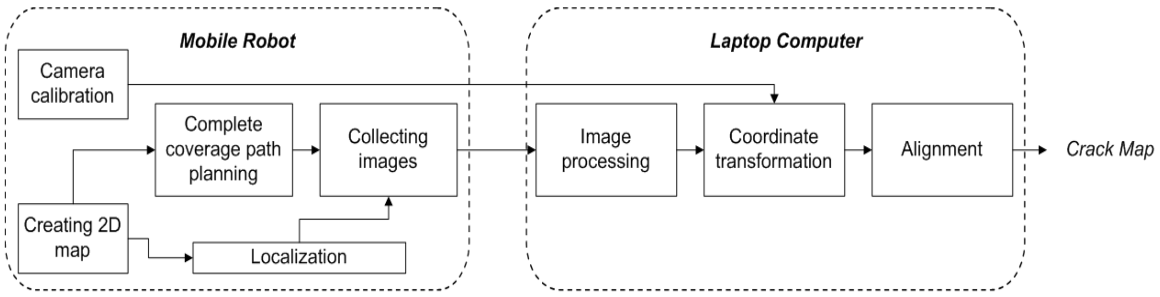


Figure 2.4: Principle of the ROCIM system.

used to localize the robot during the data collection step.

2. Data collection: The robot will navigate on the bridge deck to collect the image data at different locations. The raw image data can be stored in the on-board computer or transferred to a nearby laptop computer using wireless connection.

3. Crack detection and crack map generation: Cracks will be detected through image processing. The crack map will be created by piecing together multiple local crack maps. This step can be performed off-line on the laptop computer.

The proposed crack inspection system works according to Figure 2.4. To create a navigation map, we use a simultaneous localization and mapping (SLAM) algorithm [26]. The SLAM algorithm estimates the robot locations and creates the 2D map at the same time. We utilize the Mapper3 software [2] to create the 2D map. After the map is created, the mobile robot has a prior knowledge of the environment and the robot can localize itself based on that map using a Monte Carlo localization (MCL) algorithm [25].

Camera calibration is used to find the relationship between the image coordinate system and the robot coordinate system. Using the camera mounted on top of the robot, the position and orientation of the camera are assumed to be static. In other words, the relationship between the camera and robot coordinate system is assumed to be fixed once the camera orientation is fixed. We denote this relationship as a camera-robot transformation. The crack locations in the image coordinate system can be mapped to the robot coordinate system through this transformation. During the inspection, the robot will travel on the bridge deck and localize itself according to the 2D map. Therefore, the relationship between the robot coordinate system and the global coordinate system can be found using the MCL algorithm. We denote this relationship as a robot-global coordinate system transformation. Another transformation is an image-camera coordinate system transformation that can be found from camera calibration. In order to create the crack map, we derive an image-global coordinate system transformation which is a multiplication of image-camera, camera-robot and robot-global coordinate system transformation.

After all images are collected, they will be processed using the Laplacian of Gaussian (LoG) algorithm [27] [28] to detect the cracks. The LoG algorithm finds a zero-crossing as an edge in the second differential of the image intensity. Since the cracks are detected using the LoG algorithm, the crack locations are found in the image coordinate system. In order to create the crack map. We apply the image-global coordinate transformation to map these crack locations to the global coordinate system.

To ensure the mobile robot collects all the images on the bridge deck in an efficient manner, we explore using a complete coverage path planning (CCPP) algorithm for the ROCIM system. We propose a robotic inspection path planning based on the genetic algorithm (RIP-GA) to solve this CCPP problem.The RIP-GA algorithm will run off-line after the 2D map of the bridge deck is generated. The output of RIP-GA is a path which consists of a sequence of robot locations in the 2D map with a particular robot and camera pose. To implement this algorithm, the mobile robot has to travel according to this path and collects an image at each location.

In the last part of the ROCIM system, there is an alignment process to connect the continuous crack in different images. Since the calibration and localization have certain error due to distortion and noise, this alignment process can be used to improve the accuracy of the crack map. In this thesis, we mainly focus on developing the overall solution for crack detection and the path planning strategy for the ROCIM system. We will solve the alignment problem as part of our future work.

CHAPTER 3

CAMERA CALIBRATION

In the ROCIM system, a mobile robot detects cracks in the image coordinate system while a crack map should be created in the global coordinate system. Hence, camera calibration is needed to map the crack locations from the image coordinate system to the global coordinate system. In other words, we need to find a transformation matrix to map these crack locations. In this chapter, first, we discuss a pin hole camera model for the camera calibration. Second, a Matlab camera calibration toolbox is introduced to solve the camera calibration [4]. Last, we discuss an overall coordinate transformation to create the crack map.

3.1 Camera model

Camera calibration is a process of geometric mapping between the 3D world and 2D images using a set of points. The geometric mapping consists of extrinsic and intrinsic parameters. Figure 3.1 illustrates the pure perspective projection of the pin hole camera model. Here, we have a point P in the global coordinate system and the center of the projection is at the origin O of the camera coordinate system C. The image plane Π is parallel to the xy plane of the camera coordinate system with displacement of focal length (f) along the z axis. The principal point is an intersection of plane Π and the z axis. The coordinates of the principal point in the image coordinate system I are $[u_0, v_0]^T$.

The coordinate of P in the global coordinate system is $[X, Y, Z]^T$. The coordinate of p in the image coordinate system is $[u, v]^T$. The following homogeneous equation can be used to express the transformation of these coordinate systems.

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \alpha \begin{bmatrix} \lambda u \\ \lambda v \\ \lambda \end{bmatrix} = \mathbf{F} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \mathbf{LM} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{3.1}$$

where **F** is the perspective transformation, λ is a scale factor, **L** describes the intrinsic matrix, and **M** denotes the mapping from the global coordinate system to the camera coordinate system which consists of rotation **R** and translation **t**. The decomposition of matrix **L** and **M** are shown in Equation 3.2 and Equation 3.3, respectively.

$$\mathbf{L} = \begin{bmatrix} sf & 0 & \mu_0 & 0 \\ 0 & f & \nu_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{3.2}$$

$$\mathbf{M} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} \tag{3.3}$$

where $\mathbf{t} = [t_x, t_y, t_z]^T$ and **R** can be defined by the three Euler angles θ, ϕ, and ψ. These

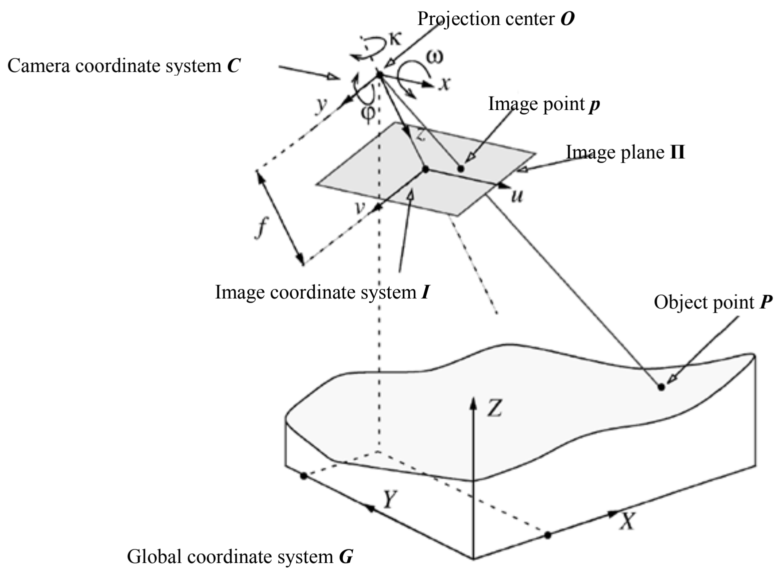

Projection center *O*
Camera coordinate system *C*
κ
ω
x
y
φ
z
Image point *p*
Image plane **Π**
u
v
f
Image coordinate system *I*
Object point *P*
Z
Y
X
Global coordinate system *G*

Figure 3.1: Illustration of pinhole camera model [3].

angles can be computed using the rotation matrix **R**.

$$\theta = sin^{-1} r_{31} \tag{3.4}$$

$$\phi = atan2(-\frac{r_{32}}{\cos\theta}, \frac{r_{33}}{\cos\theta}) \tag{3.5}$$

$$\psi = atan2(-\frac{r_{21}}{\cos\theta}, \frac{r_{11}}{\cos\theta}) \tag{3.6}$$

The variables $t_x, t_y, t_z, \theta, \phi$, and ψ are called extrinsic parameters and s, f, u_0, and v_0 are denoted as intrinsic parameters.

Figure 3.1 shows the ideal projection of a point in the global coordinate system to the image coordinate system. In practice, there will be an error (distortion) caused by the lens system. Many research works have been proposed to address with this problem. The main approach is to decompose the distortion into radial and decentering components [3]. Here, we utilize the Matlab camera calibration toolbox to conduct the camera calibration. The interface of this toolbox is shown in Figure 3.2.

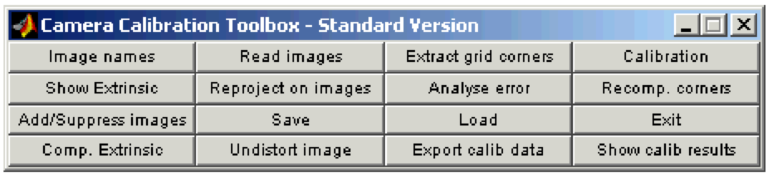


Figure 3.2: Matlab camera calibration toolbox [4].

3.2 Camera calibration using Matlab toolbox

In this section, we briefly present the steps of using Matlab toolbox to solve the camera calibration problem for the ROCIM system. The input of this toolbox is a set of n points in the image and the global coordinate system where $n > 5$. The output is the intrinsic and extrinsic parameters of the camera calibration. The procedure of the calibration in our experiment is described as follows:

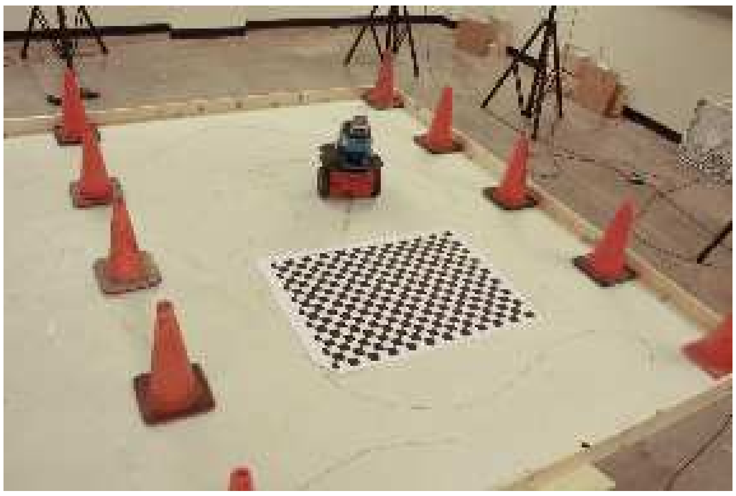
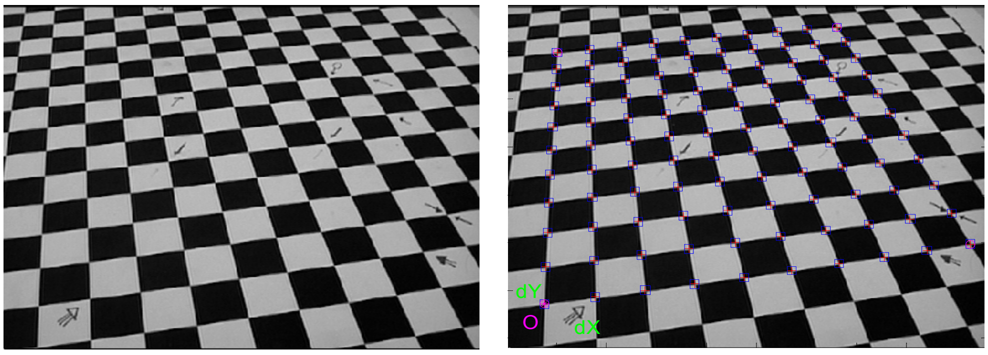


Figure 3.3: Camera calibration for the ROCIM system.



(a) Image for camera calibration (b) Corner extraction result

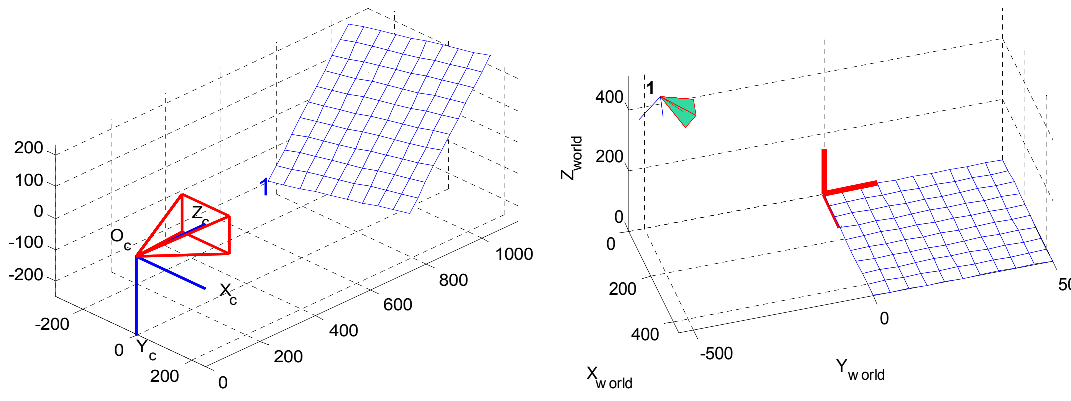
Figure 3.4: A chessboard is used for camera calibration.

1. Setup the experiment as shown in Figure 3.3. A chessboard is used as the object for the calibration. Since the grid has the same length but different colors for neighboring grid cells, the location of the corner points can be estimated accurately.

2. Capture an image of the chessboard using the mobile robot camera as shown in Figure 3.4(a). This image will be used as the input for the toolbox.

3. Extract the grid corners to find the set of n points in the image and the global coordinate system. Figure 3.4(b) shows the result of the corner extraction.

4. Run the calibration. The results of the calibration are the intrinsic parameters and the



(a) camera-oriented view (b) global-oriented view

Figure 3.5: The result of camera calibration using Matlab toolbox.

extrinsic parameters which are shown in Figure 3.5(a) and Figure 3.5(b).

3.3 Coordinate transformation

In this section we discuss the coordinate transformation to map the crack location from the image coordinate system to the global coordinate system. As we mentioned before that the mobile robot travels in the inspection area and collects the image of the bridge deck surface. The location and orientation of the mobile robot can be estimated using Monte Carlo localization (MCL) algorithm [29] [25]. The MCL algorithm is a sampling-based method to approximate a probability density distribution of the robot locations. Each sample consists of a possible robot location and a probability that the robot currently is at that location. We use advanced range navigation laser (ARNL) library [2] to implement the MCL algorithm. Figure 3.6 shows a graphic user interface (GUI) of the mobile eyes software where the MCL result can be visualized [2]. The red circle represents the mobile robot body. The green and blue dots denote previous and current samples, respectively. These samples are measured using the laser sensor. The information of the current position, orientation and the probability for the current location can be seen in this GUI.

Camera calibration finds the extrinsic and the intrinsic parameters of the camera. Subsequently, we utilize those parameters to map all crack locations to the 2D map by assuming

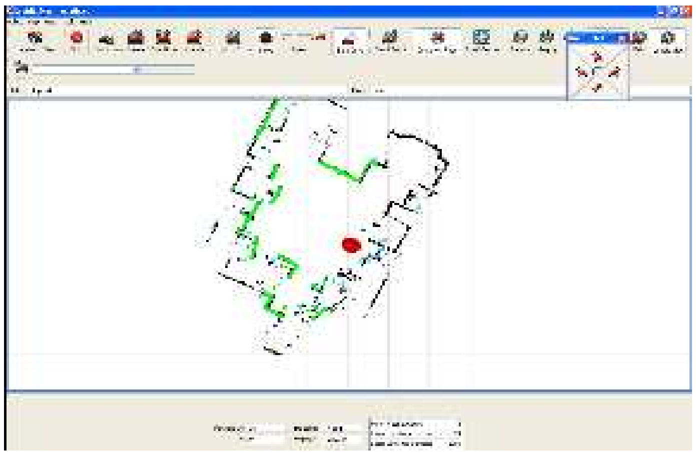


Figure 3.6: Graphic user interface for mobile eyes [2].

that the transformation between the camera coordinate system and the robot coordinate system is fixed, for a given camera orientation.

The ROCIM system involves five coordinate systems as shown in Figure 3.7. They are: image coordinate system (F_I), camera coordinate system (F_C), local coordinate system (F_L), robot coordinate system (F_R) and global coordinate system (F_G). After we do the camera calibration, we have the intrinsic and the extrinsic parameters. The intrinsic parameters are focal length (f), skew value (s) and the origin of image coordinate system (μ_0, ν_0)

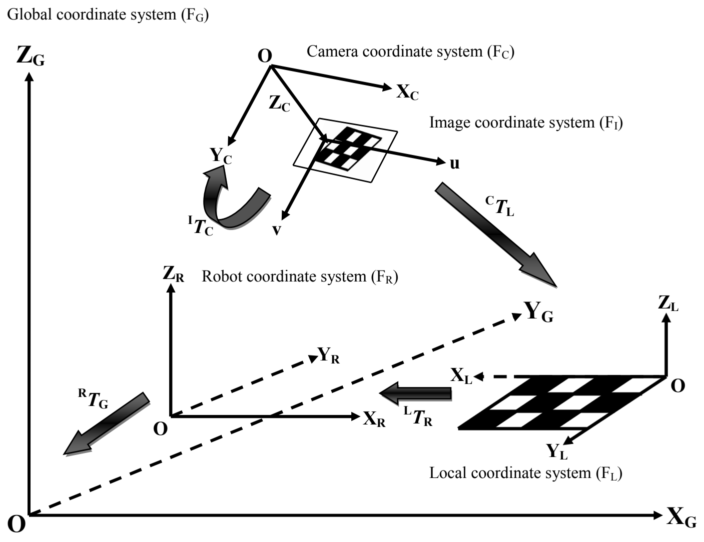

Global coordinate system (F_G)
Z_G
O
Camera coordinate system (F_C)
X_C
Z_C
Image coordinate system (F_I)
u
Y_C
$^{C}T_L$
$^{I}T_C$
v
Z_R
Robot coordinate system (F_R)
Y_G
Z_L
Y_R
X_L
O
$^{R}T_G$
O
X_R
$^{L}T_R$
Y_L
Local coordinate system (F_L)
O
X_G

Figure 3.7: Coordinate systems in the ROCIM system.

as described in Equation (3.2). The extrinsic parameters are rotation (**R**) and translation (**t**) matrices as in Equation (3.3) [3]. These parameters define the transformation matrix ${}^{I}T_{C}$ and ${}^{C}T_{L}$, respectively. In the ROCIM system as the robot collects the images, it saves its current location in the global coordinate system. After we detect the crack in each image, we should plot all the crack locations in the global coordinate system. To do this, we have the crack locations in the image coordinate system (u, v). We map the crack locations to the robot coordinate system (X_R, Y_R) using the camera-robot transformation ${}^{C}T_{R}$, which is derived from the multiplication of transformation matrices as below.

$$ {}^{C}T_{R} = {}^{C}T_{L}\,{}^{L}T_{R} \tag{3.7}$$

The transformation ${}^{L}T_{R}$ can be calculated using pseudo-inverse as in Equation 3.8. Here, D_L and D_R are a set of point locations in the local coordinate system and the robot coordinate system, respectively. We utilize an optical tracking system to find the location of these points [30].

$$ {}^{L}T_{R} = (D_L^T D_L)^{-1} D_L^T D_R \tag{3.8}$$

We apply another transformation from the robot coordinate system to the global coordinate system using the transformation ${}^{R}T_{G}$, which can be calculated from the robot location (x, y, θ). Therefore from the image coordinate system to the global coordinate system, we have the following transformation:

$$ {}^{I}T_{G} = {}^{I}T_{C}\,{}^{C}T_{R}\,{}^{R}T_{G} \tag{3.9}$$

CHAPTER 4

CRACK DETECTION

In this chapter, we discuss Laplacian of Gaussian (LoG) as the crack detection algorithm in the ROCIM system. We present how to apply this algorithm to detect cracks in each image captured by the mobile robot. We also discuss our graphic user interface (GUI) to detect the cracks.

4.1 Crack detection algorithm

The main idea of detecting cracks is to find out the edge points in an image. These edge points can be detected by finding the zero-crossings of the second derivative of the image intensity. However, calculating the second derivative is very sensitive to noise. Hence, this noise should be filtered out. To achieve this, the Laplacian of Gaussian (LoG) algorithm [31] is used. This method combines Gaussian filtering with the Laplacian for edge detection. In the LoG edge detection, there are mainly three phases: filtering, enhancement, and detection. Especially, Gaussian filter is used for smoothing and its second derivative

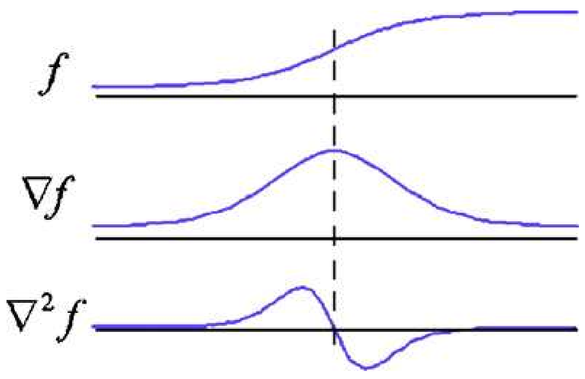


Figure 4.1: First and second order derivative of one dimension signal [5].

is used for enhancement. The detection criteria is the presence of a zero-crossing in the second derivative with the corresponding large peak in the first derivative as shown in Figure 4.1.

In this approach, the noise is firstly reduced by convoluting the image with a Gaussian filter. Then isolated noise points and small structures are filtered out. With smoothing, edges are spread out. Those pixels that have locally maximum gradient are considered as edges by the edge detector in which zero-crossings of the second derivative are used. To avoid detecting insignificant edges, only the zero-crossings with corresponding first derivative above some thresholds are selected as edge points. The edge direction is obtained by using the direction in which zero-crossing occurs.

The Laplacian is a 2D isotropic measure of the 2nd spatial derivative of an image. The Laplacian of an image highlights regions of fast intensity change and is often used for edge detection. We define a 2-D second order derivative as the Laplacian function which is isotropic [31].

$$(\nabla^2 f)(x,y) = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \tag{4.1}$$

The Laplacian is often applied to an image that has first been smoothed with a Gaussian smoothing filter in order to reduce its sensitivity to noise. Hence, the two variants will be described together here. The operator normally takes a single gray level image as input and produces another gray level image as output. The Laplacian L(x,y) of an image with pixel intensity values I(x,y) is given by:

$$L(x,y) = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2} \tag{4.2}$$

Since the input image is represented as a set of discrete pixels, we have to find a discrete convolution kernel that can approximate the second derivatives in the definition of the Laplacian.

Here we use two kernels with sizes 15×15 and 17×17 in combination with the changing thresholds and variances to check the effect of edges in the images. Figure 4.2 shows the

LoG kernel with kernel size equal to 17×17. Using these kernels, the Laplacian can be calculated using standard convolution methods. Because these kernels are approximating a second derivative measurement on the image, they are very sensitive to noise. To counter this, the image is often smoothed using Gaussian filter before the Laplacian filter is applied. This pre-processing step reduces the high frequency noise components prior to the differentiation step. Because the convolution operation is associative, we can convolve the Gaussian smoothing filter with the Laplacian filter, and then convolve this hybrid filter with the image to achieve the required result. Doing things this way has two advantages:

- Since both the Gaussian filter and the Laplacian kernels are usually much smaller than the image, this method usually requires far fewer arithmetic operations.
- The LoG kernel can be pre-calculated in advance so only one convolution needs to be performed at run-time on the image.

The 2D LoG function centered on zero and with Gaussian standard deviation has the form:

$$LoG(x,y) = -\frac{1}{\pi\sigma^4}\left[1 - \frac{x^2+y^2}{2\sigma^2}\right]e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{4.3}$$

If the image is pre-smoothed by a Gaussian low-pass filter, then we have the LoG operation that is defined as

$$(K_{\nabla^2} * *(G_\sigma * *I)) = (K_{\nabla^2} * *G_\sigma) * *I = (\nabla^2 G_\sigma) * *I \tag{4.4}$$

where

$$\nabla^2 G_\sigma(x,y) = \frac{1}{2\pi\sigma^4}\left[\frac{x^2+y^2}{\sigma^2} - 2\right]e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{4.5}$$

To find the places where the value of the Laplacian passes through zero points and also changes sign, we use the zero-crossing detector. We know that such points often occur at edges where the intensity of the image changes rapidly, but they also occur at places that are not easy to associate with edges. It is better to think of the zero-crossing detector as some sort of feature detector rather than as a specific edge detector. Zero-crossings always

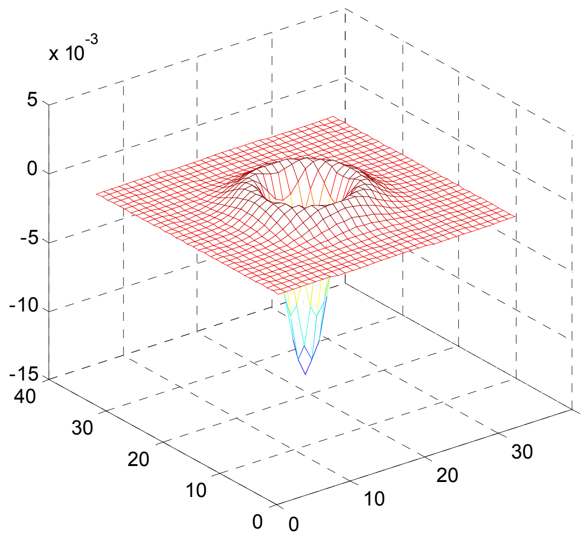


Figure 4.2: LoG kernel.

lie on closed contours, and so the output from the zero-crossing detector is usually a binary image with single-pixel lines showing the positions of the zero-crossing points.

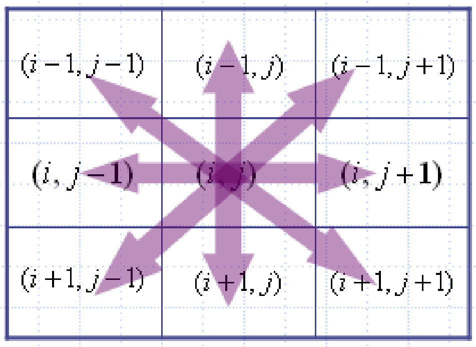


Figure 4.3: Zero-crossing.

The initial input for the zero-crossing detector is an image which has been filtered using the Laplacian of Gaussian filter. The resulting zero-crossings are strongly affected by the size of the Gaussian used for the smoothing stage of this operator. As the smoothing is increased, then fewer and fewer zero-crossing contours will be found, and those that remain will correspond to features of larger and larger scale in the image. The summary of the crack detection algorithm is shown in Algorithm 1.

```
Algorithm 1: Crack Detection
    Step 1. Apply the LoG to the image.
    Step 2. Apply the zero-crossing detector in the image.
    Step 3. Filter the zero-crossings to keep only those strong ones (large difference
    between the positive maximum and the negative minimum).
    Step 4. Suppress the weak zero-crossings most likely caused by noise.
```

4.2 Operation interface for crack detection

In this subsection, we discuss the operation interface for crack detection by using a graphical user interface (GUI) as shown in Figure 4.4. The input images captured by the robot are collected and stored on the computer. There are two modes of operation, first, we can select the image in **Browse**, then the **Manual-Run** button will start the crack detection algorithm and the crack results (white lines) are superimposed on the output image. Second, tn the **Auto-Run** mode, it allows us to detect the cracks on all images automatically. After the cracks on each image is detected, the crack locations in the image coordinate system are mapped to the 2D map.

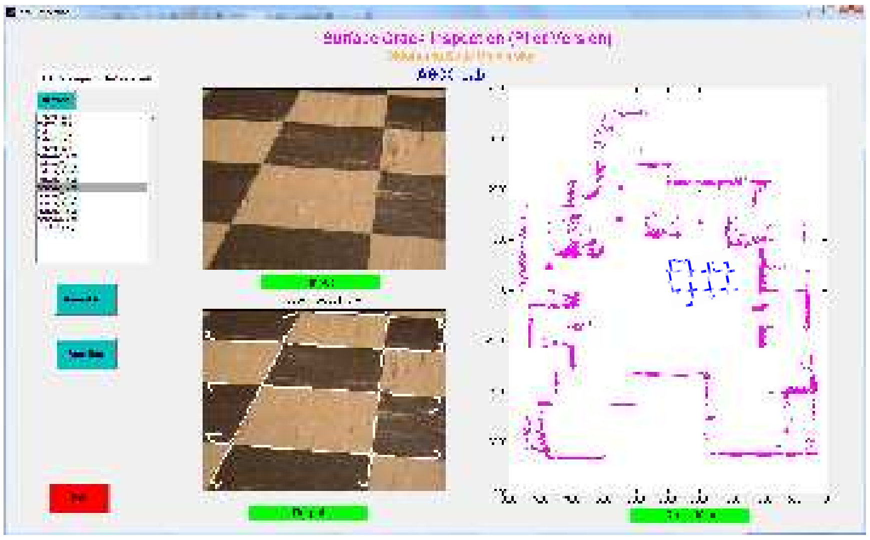


Figure 4.4: GUI for crack detection and mapping.

CHAPTER 5

COMPLETE COVERAGE PATH PLANNING

In this chapter, we present a new path planning algorithm that allows the robot to travel efficiently during the inspection. The path planning is important for the ROCIM system to save time and energy. We first formulate the problem, then we propose two approaches to solve this problem.

5.1 Problem formulation

The purpose of complete coverage path planning (CCPP) is to ensure the union of camera field of views (FoV) covers the whole bridge deck surface. There are several issues that we need to address in this CCPP problem. First, a typical camera installation is mounted on top of the mobile robot as shown in Figure 5.1. Therefore, there is a blind spot which is caused by the offset distance between the center location of the robot body and the center location

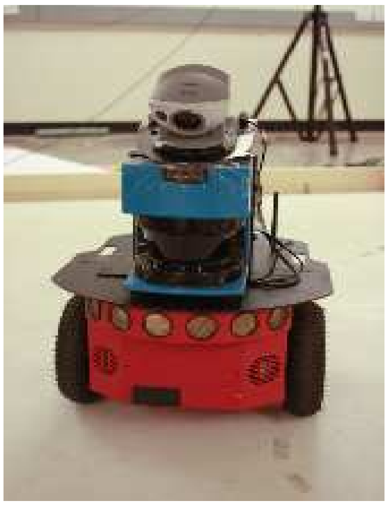


Figure 5.1: Pioneer3-DX with the camera on top of it.

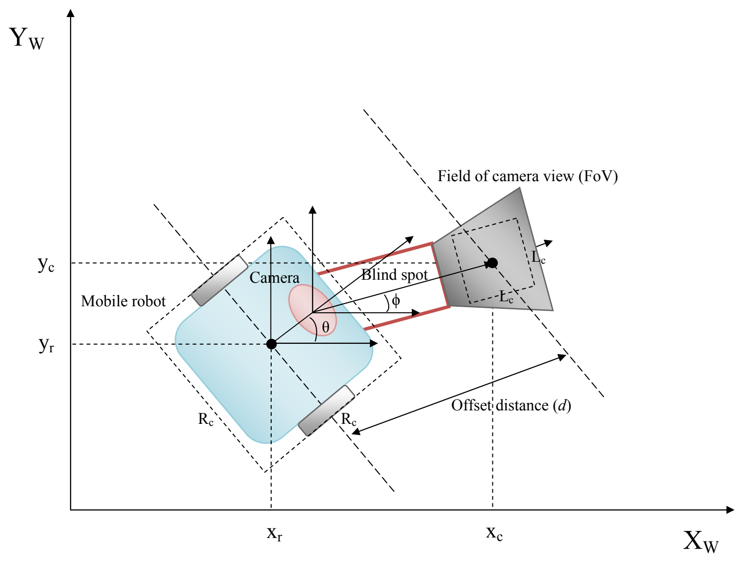


Figure 5.2: Projection of mobile robot and field of camera view (FoV) to an XY plane.

of the FoV. Second, to allow the mobile robot to detect small cracks, the camera should zoom in. Hence, it makes the size of FoV smaller than the size of robot body. To illustrate these issues, we project the robot body and FoV to an XY-plane as shown in Figure 5.2. To simplify the problem, the area of interest is partitioned into cells which have the same size as the FoV. We define several concepts to develop the solution for this problem. These definitions are described as follows:

Definition 1: Configuration of Inspection

Let $(x_r, y_r) \in \mathbf{R}^2$ be the center location of the mobile robot; $\theta \in [0, 2\pi]$ be the orientation of the mobile robot; and $\phi \in [-\pi/2, \pi/2]$ be the pan angle of the camera. Then, a *configuration of inspection* (CoI^i) is defined as:

$\mathrm{CoI}^i \triangleq (x_r, y_r, \theta, \phi)$ where i denotes a cell.

Definition 2: Motion Template

A motion template, MT(i, j), is the mobile robot and camera motion plan that transitions from CoI^i to CoI^j:

MT(i, j) $\triangleq \mathrm{CoI}^i \rightarrow \mathrm{CoI}^j$, $j \in \mathbf{N}(i)$, here $\mathbf{N}$ is the neighborhood of cell i as shown in Figure 5.4(a).

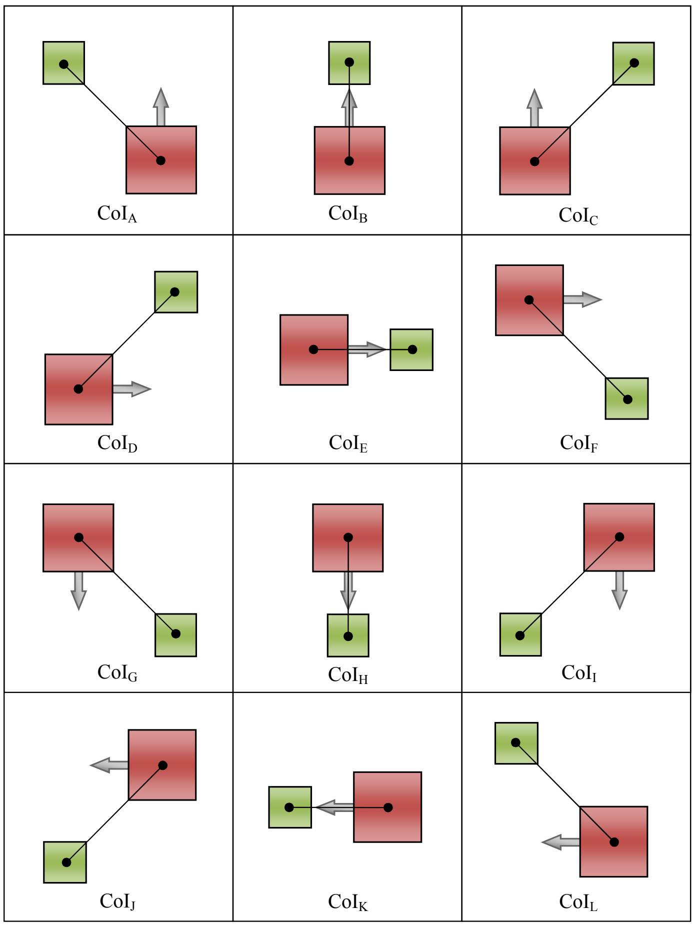


Figure 5.3: Twelve configuration of inspections (CoI).

Definition 3: Inspection Path

An inspection path, IP, is a sequence of motion templates that ensure the union of camera FoVs cover the whole area of interest. An *inspection path* is defined as:

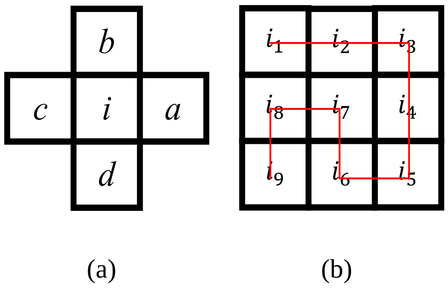


Figure 5.4: (a) Four directions of camera path defines four neighbor cells. (b) An example of camera path to cover the whole area.

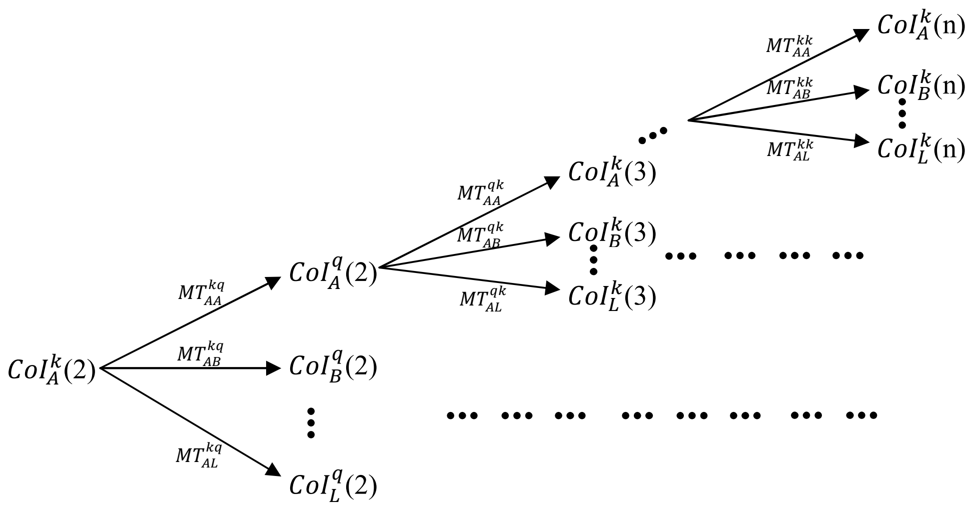


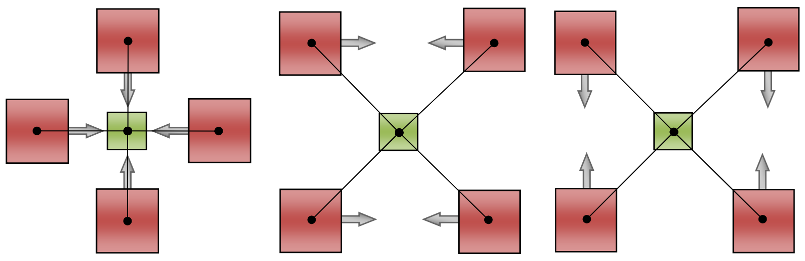
Figure 5.5: Relationship between configuration of inspection (CoI), motion templates (MT) and inspection path (IP).

IP ≜ [MT(i_1, i_2), MT(i_2, i_3),..., MT$^($$i_{n-1}i_n$)] where ($i_1, i_2, ..i_n$) is the camera path (a sequence of all the cells). An example of camera path is shown in Figure 5.4(b).

In Figure 5.3, we show the twelve CoIs where each CoI has a different robot orientation (θ) and camera pan angle (ϕ) where θ = (0^0, 90^0, 180^0, 270^0) and ϕ = (-45^0, 0^0, 45^0). We use twelve CoIs because the robot heading (θ) is discretized into four values and the camera pan angle (ϕ) is discretized into three values. We encode the CoIs using alphabet from A to L: **S** = (A, B, C, D, E, F, G, H, I, J, K, L) as shown in Figure 5.3. The red square (darker square) is the mobile robot and the green square (lighter square) is the FoV. The relationship between the three definitions is shown in Figure 5.5. In a free space, all the twelve CoIs can be applied to cover a cell i as shown in Figure 5.6.

In Figure 5.4(a) we show that cell i has four neighbors (a, b, c, d) because we assume the camera FoV moves only in four directions such as up, down, left and right. Therefore, there are 512 (12×12×4) different motion templates base on twelve CoIs and four camera path directions.

The problem of complete coverage path planning for the ROCIM system is how the mobile robot plans a path (inspection path) in order to have complete coverage of the cells in an



(a) (b) (c)

Figure 5.6: In free space, to inspect a cell, there are twelves CoIs that can achieve it.

efficient manner. This can be done by selecting the combination of CoIs which minimizes a fitness function. The fitness function consists of three components: traveling distance, robot turns, and camera turns. The robotic inspection path planning based on genetic algorithm (RIP-GA) is proposed to minimize the number of robot turns and traveling distance. The last component (camera turns) is not considered because the camera works in parallel to the mobile robot. In summary, the CCPP problem can be stated as follows:

Given an area of interest, which is partitioned into cells, find an inspection path that covers all the cells while minimizing the numbers of robot turns and traveling distance.

In this thesis, we are interested in finding the sub-optimal solution because finding the optimal solution is an NP-hard problem and the computation increases exponentially with the number of cells.

5.2 Two approaches to complete coverage path planning

Many existing approaches cannot be directly applied to solve the CCPP problem because in these approaches, the robot body is used for the coverage [19] [21] [22] [32] [33]. In this section, we discuss two different approaches to the CCPP problem. First, we present a heuristic approach which provides a full coverage of cells, however the traveling distance and the number of robot turns are not optimized. Second, we present a RIP-GA approach which gives a better solution than the heuristic approach.

5.2.1 Heuristic Approach

In this approach, the robot plans the inspection path based on the robot body movement. Typical motion planning such as boustrophedron motion [19], wall-following motion, and spiral motion can be used to plan the path. After the robot covers the entire area using the robot body, the coverage of FoV is considered. If the complete coverage is not achieved, the robot needs to inspect the uncovered cells by adding more movements. Pseudo-code of this approach is shown in Algorithm 1. We show some examples using this approach that can give a full coverage of the cells however the number of robot turns and traveling distance are not minimized. The examples are presented as follows:

Example 1

We consider an empty rectangular map without any obstacle with size 80×120 as in Figure 5.7(a). The map is discretized into square cells with the same size as the FoV. The boustrophedron motion from top to bottom is applied to the mobile robot with camera facing forward, i.e., CoI_B. The mobile robot starts moving from $(-50, 30)$ to $(50, -30)$. These are missing areas especially along the edges as shown in Figure 5.7(a). The trajectory of this robot motion is shown in Figure 5.7(b). To cover the missing areas, the robot needs to re-plan the motion which results in a lot of overlapping path. Therefore, the solution is not very good.

Algorithm 2: Heuristic Approach

```
Step 1. Generate a mobile robot path to cover the area.
if there is any uncovered cell then
    Step 2. Find out the uncover cells and let the robot go back to cover these cells.
else
    Step 3. Finish.
```

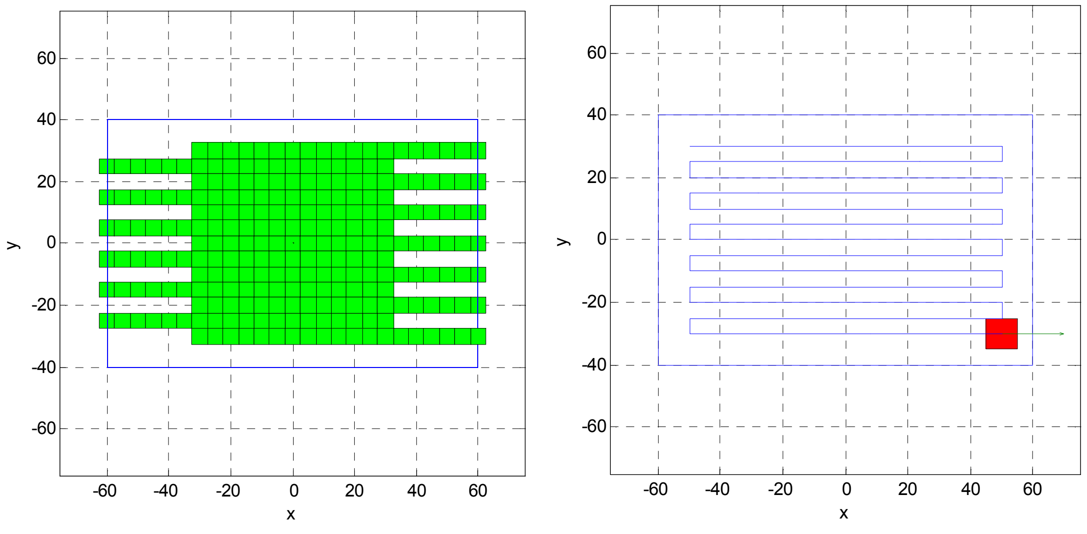


Figure 5.7: Mobile robot trajectory with 0^0 camera orientation.

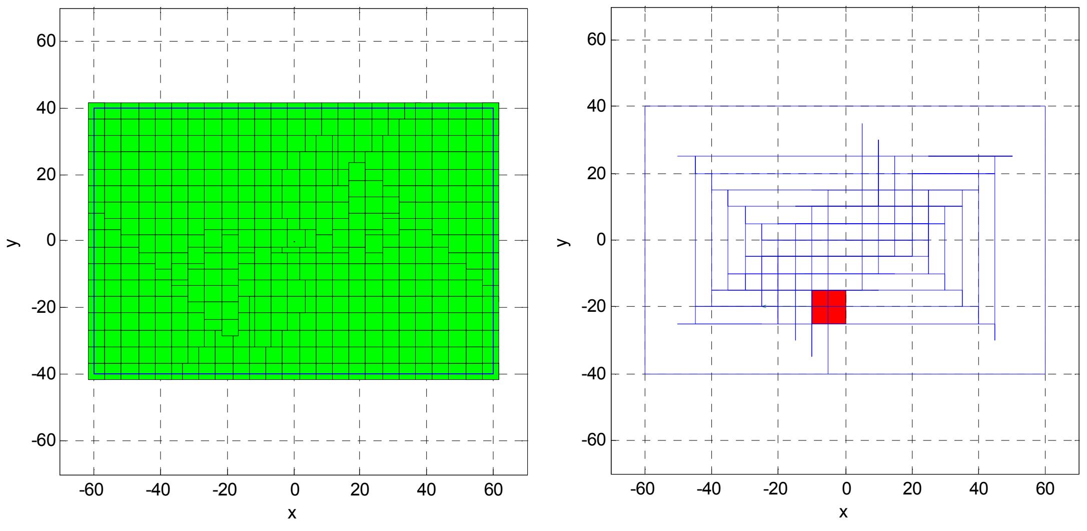


Figure 5.8: Trajectory of the wall following mobile robot motion with 45^o camera orientation.

Example 2

In this example we change the pan-orientation of camera to 45^0 (CoI_C) which allows the robot to inspect the cells along the edges. Then, we apply wall-following motion for the mobile robot to cover the entire area [28]. The coverage of this motion planning is shown in Figure 5.8(a) and the trajectory of the robot motion is shown in Figure 5.8(b). The result shows that full coverage is achieved. However there is unnecessary overlap in the coverage and robot paths. Therefore the traveling distance and the number of robot turns are not satisfying. As the result, this heuristic approach does not provide a good solution to the CCPP problem.

5.2.2 RIP approach

The RIP approach finds an inspection path (IP) that minimizes the total traveling distance and the number of robot turns by selecting the Configuration of Inspection (CoI). The proposed algorithm extends the idea of the proposed RPP-CP (robotic path planning based on the camera path) [28]. The RPP-CP and RIP approaches assume that the camera path is planned in the first step in order to have complete coverage of FoV. Then, the robot and camera motion are planned according to the camera path.

To select the CoI, we utilize a genetic algorithm as a stochastic method and greedy algorithm as a deterministic method. Here, we denote RIP-greedy as the robotic inspection path planning based on greedy algorithm. The genetic algorithm is an optimization method which mimics the natural evolution. The solution is generated using techniques such as mutation, selection and crossover [34]. We model some parameters to fit the genetic algorithm as follows:

- The gene is defined as the CoI.
- The chromosome (C_r^I) is a sequence of CoIs, $C_r^I = (CoI^{i_1}, CoI^{i_2}, CoI^{i_3}, ..., CoI^{i_n})$ and the length of C_r^I is the same as the number of cells (n).

Figure 5.10(a) shows the illustration of the chromosome and it represents the solution to this CCPP problem. For each generation the genetic algorithm tries to find a better chromosome by minimizing the fitness function.

$$F = \Sigma_{j=1}^{n}(\alpha||q(i_j, i_{j+1})|| + \beta||u(i_j, i_{j+1})||) \tag{5.1}$$

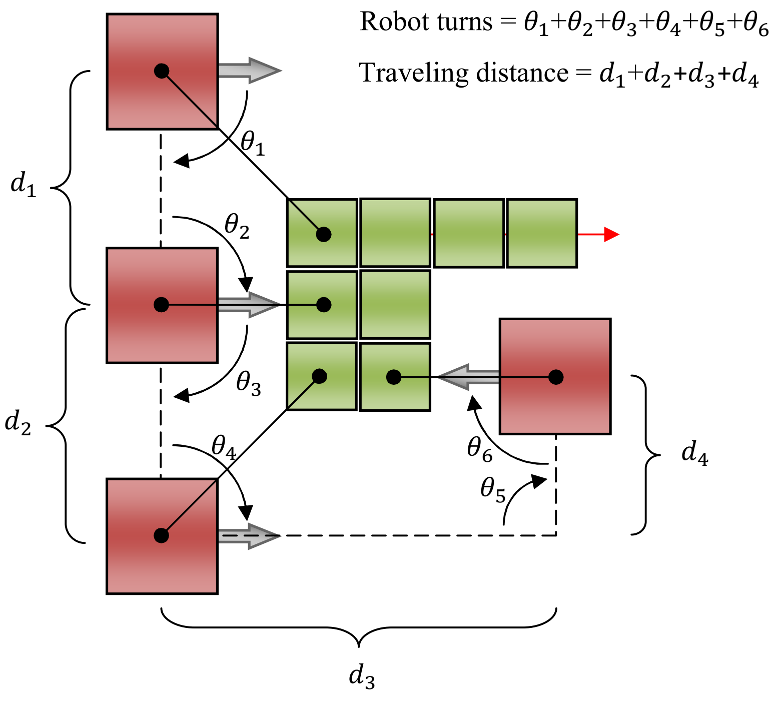

Robot turns = $\theta_1+\theta_2+\theta_3+\theta_4+\theta_5+\theta_6$
Traveling distance = $d_1+d_2+d_3+d_4$
θ_1
θ_2
θ_3
θ_4
θ_5
θ_6
d_1
d_2
d_3
d_4

Figure 5.9: Calculation of the fitness function.

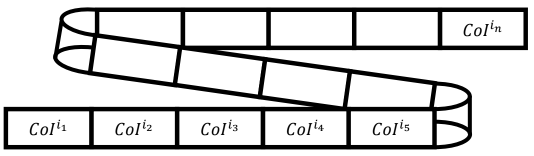
Here, the functions q, u basically are the traveling distance and robot turns, respectively. These functions are defined based on the path planning algorithm of the mobile robot. The variables α, β are the weight value for distance and time. In Figure 5.9 we show an example of fitness function calculation from one CoI to another CoI.

The pseudo-code of the RIP-GA algorithm is shown in Algorithm 2. In the step 4 of Algorithm 2, the multiple cross-over operation is applied to mate the nearest pair. This operation is used to produce offspring (ω^I, ω^{I+1}).

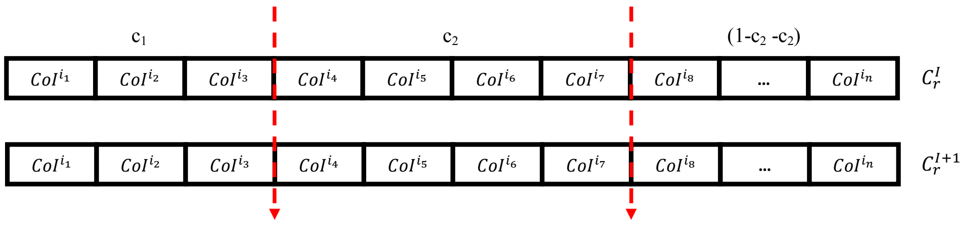
$$\omega^I = c_1C_r^I + c_2C_r^{I+1} + (1 - c_1 - c_2)C_r^I \tag{5.2}$$

$$\omega^{I+1} = c_1C_r^{I+1} + c_2C_r^I + (1 - c_1 - c_2)C_r^{I+1} \tag{5.3}$$

Equation 5.2 and 5.3 have to satisfy the following conditions as follows: $c_1, c_2 \in [0,1]$ and $c_1 + c_2 \leq 1$. Here, c_1, c_2 is a percentage of the CoI from one chromosome. For example, let $c_1 = 0.2$, $c_2 = 0.3$ then the chromosome ω^I takes 70% genes from C_r^I to mate with 30% of the C_r^{I+1}. The illustration of the cross-over operation is shown in Figure 5.10(b). Then, the mutation operation is applied in order to avoid the genetic algorithm getting stuck in



(a)



(b)

Figure 5.10: (a) Chromosome (C_r^I). (b) Multiple chromosome crossover.

local minimum. After the crossover and mutation are applied, the offsprings (ω^I, ω^{I+1}) are evaluated with the parents (C_r^I, C_r^{I+1}). For each offspring, we calculate and compare the fitness value to the parents. We discard two of these chromosomes (offsprings and parents) which have the most fitness value. In the step 6, we remove some chromosome in the population which have the most fitness value and new chromosomes are introduced to fill in. The algorithm continuously runs until the fitness value of the top chromosome does not change for Q iterations.

We compare the genetic algorithm with the greedy algorithm to validate the proposed RIP approach. The greedy algorithm searches locally for the minimum fitness value for the next CoI along the camera path. The greedy algorithm is robust to select the CoI but it always gets stuck in local minimums. The pseudo-code of the RIP-greedy algorithm is shown in Algorithm 4.

```
Algorithm 3: RIP-GA
  Step 1. Generate a camera path $CP$(n) where $CP = (i_1, i_2, i_3..., i_n), n \in \mathbf{Z}$.
  for each CP(n) do
      Find a set of feasible CoI$_{(S)}$
  Step 2. Generate a population of $M$ genetic strings randomly from $CP$;
  $P^I = (\omega_1^I, \omega_2^I, \omega_3^I..., \omega_N^I)$ ; I = 1, 2...,M.
  while the genetic algorithm is not converged do
      1. Compute the fitness function using Equation 5.1
      2. Rank the genetic strings from top to bottom, $C^I(I = 1, 2, , M)$
      3. Put the top chromosome to the next generation (Elitism).
      4. Mate the nearest pairs.
      5. Mutate the offspring.
      6. Kill the bottom genetic strings ($B < M$) and keep the top K parents.
      7. New parents = K ∪ R, and R is a random population to replace the bottom
      genetic strings.
      8. Evaluate the best chromosome
      if the genetic algorithm is converged then
          Go to Step 3.
  Step 3. Generate the inspection path based on the CoI sequences (Inspection path).
```

```
Algorithm 4: RIP-Greedy
  Step 1. Generate a camera path $CP$(n) where $CP = (i_1, i_2, i_3..., i_n), n \in \mathbf{Z}$.
  for each CP(n) do
      Find a set of feasible CoI$_{(S)}$
  Step 2. Find the minimum motion templates for each cell.
  Step 3. Generate the robot trajectory based on the CoI (Inspection path).
```

CHAPTER 6

EXPERIMENT RESULTS

We conducted both simulations and experiments to validate the proposed ROCIM system and the associated algorithms. In the first experiment, we apply the ROCIM system to detect real cracks on the bridge deck. In the second experiment, we demonstrate the working of the ROCIM system in the indoor and outdoor environments. Finally, we evaluate the complete coverage path planning (CCPP) algorithm in both simulation and real robot implementation.

6.1 Real crack detection

In order to evaluate the performance of the ROCIM system, we first tested our crack detection algorithm using real crack images which are collected from a bridge deck. The original image is shown in Figure 6.1(a). The Laplacian of Gaussian (LoG) algorithm is applied to the image, and the results are shown in Figure 6.1(b). The detected crack is superimposed on the original image in Figure 6.1(c). Here, the result clearly shows the LoG algorithm successfully detects the crack in the image. The parameters of the LoG algorithm are: σ = 2.5 and threshold T = 3.

One of the limitations of the LoG algorithm is that it is difficult to find the optimal parameters such as σ and T. For example, using $\sigma \leq 1.5$ and T = 2, the result shows too much noise and the crack is barely detectable (See Figure 6.2). On the other hand, if we set σ or T too large, the LoG removes most of the crack information. In Figure 6.2, we show the result of the LoG algorithm using various σ and T to detect the crack. In addition, our initial tests indicate that the smallest detectable crack is around 1 mm in width.

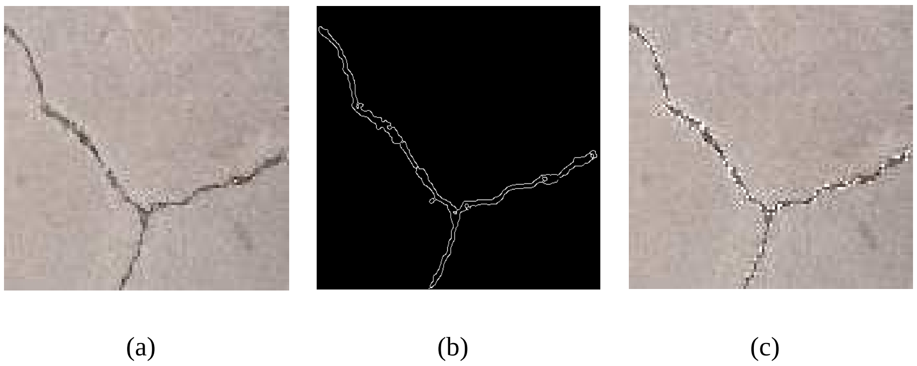


Figure 6.1: Crack detection result on a real bridge deck.

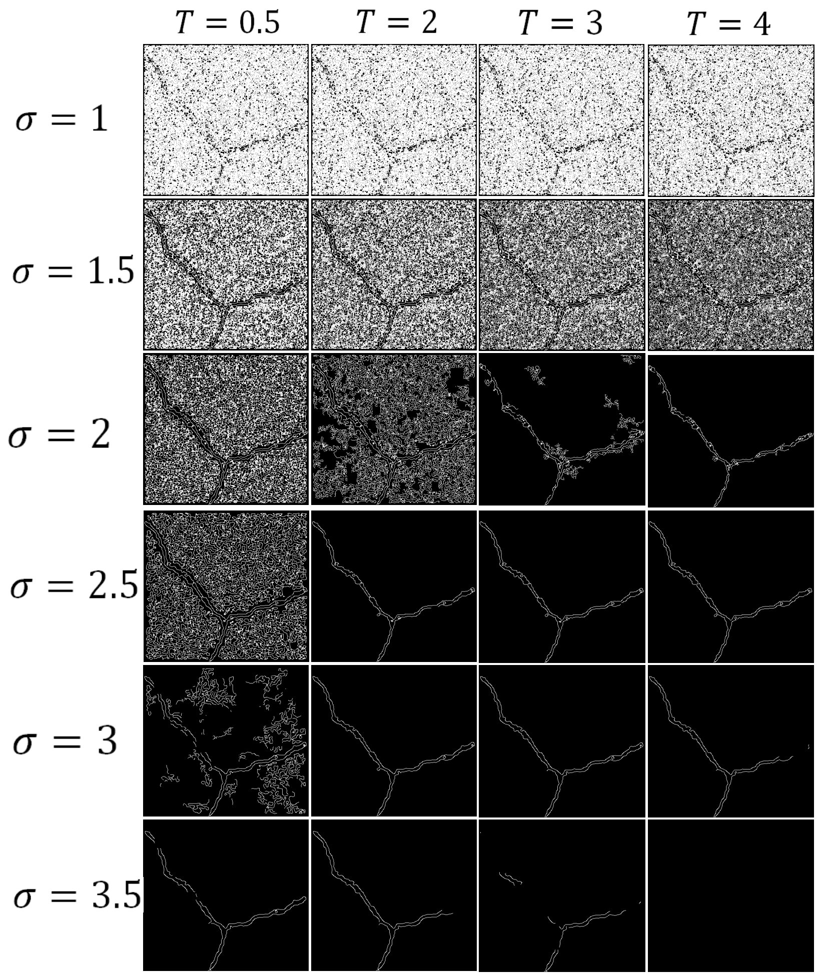


Figure 6.2: Comparison of the LoG parameters for crack detection.

6.2 Validation of the ROCIM system

In this section, we demonstrate the working of the ROCIM system. Three types of environment are evaluated in this experiment. The first is a simple indoor environment. We define a grid by using the line of each tile as cracks. The second is a complex indoor environment. We create artificial cracks by drawing curves on the ground. We use the cones to set the inspection area. The third is an outdoor environment. We investigate the ROCIM system to detect the crack on the concrete surface.

6.2.1 Simple indoor environment

The hardware setup for the simple indoor environment is shown in Figure 6.3. We use the advanced range navigation laser (ARNL) and the Mapper3 software to create a 2D map of the environment [2]. The 2D map of this environment is shown in Figure 6.4(a). Next, the 2D map is used to localize the mobile robot during the inspection using Monte Carlo localization (MCL) algorithm. Before we run the inspection, we calibrate the camera using the grid on the floor. The image that we collected for the camera calibration is shown in

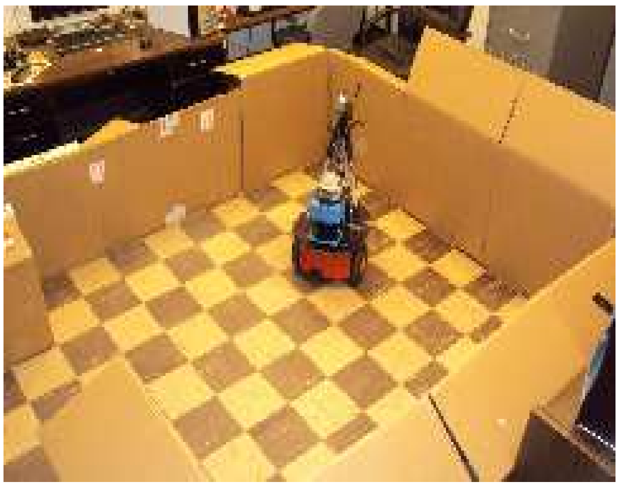
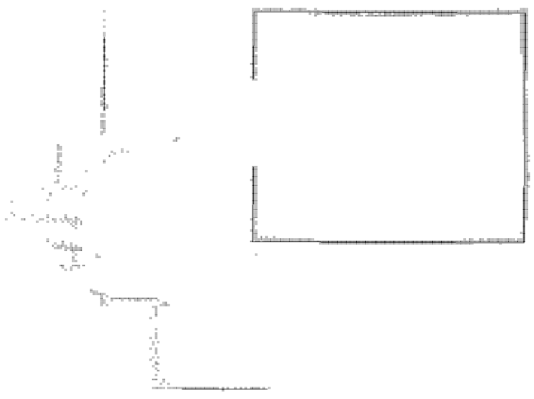
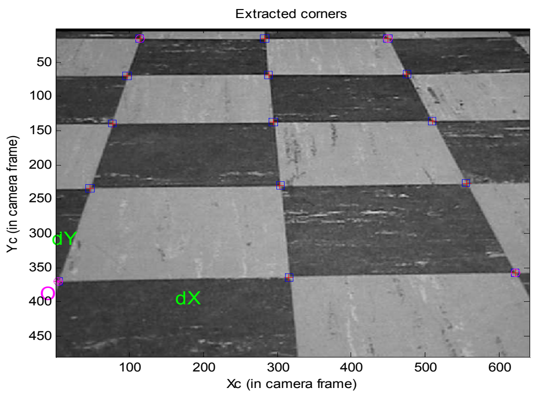


Figure 6.3: Experiment setup for simple indoor environment.


(a) A 2D map of the simple indoor environment.



(b) Grids on the floor for camera calibration.

Figure 6.4: Simple indoor environment.

Figure 6.4(b). The intrinsic and extrinsic parameters for the camera calibration with respect to the robot coordinate system are obtained as follows:

$$L = \begin{bmatrix} 220.3092 & 0 & 79.5000 \\ 0 & 220.3092 & 59.5000 \\ 0 & 0 & 1.0000 \end{bmatrix} M = \begin{bmatrix} 0.0370 & 0.9993 & 0.0102 & -414 \\ 0.7316 & -0.0202 & -0.6814 & -1157 \\ -0.6807 & 0.0327 & -0.7318 & 1924 \end{bmatrix}$$

There are six images collected from different places. Then, the LoG algorithm is applied to detect the crack in each image as shown in Figure 6.5. Next, we map all the crack locations to the global coordinate system. Figure 6.6 gives the overall crack map based on the six images. The blue and black lines represent the crack and the 2D map, respectively. The result shows that the transformation of the crack locations from the image coordinate system to the global coordinate system works well and the crack map is successfully created. However, it can also be noticed that there are some misalignments in the crack map.

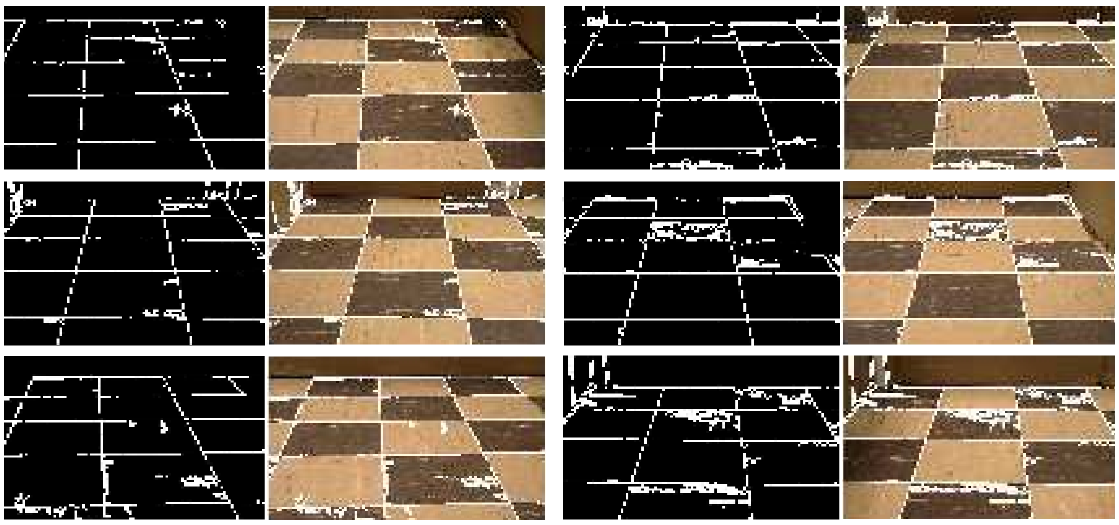


Figure 6.5: Crack detection result for simple indoor environment.

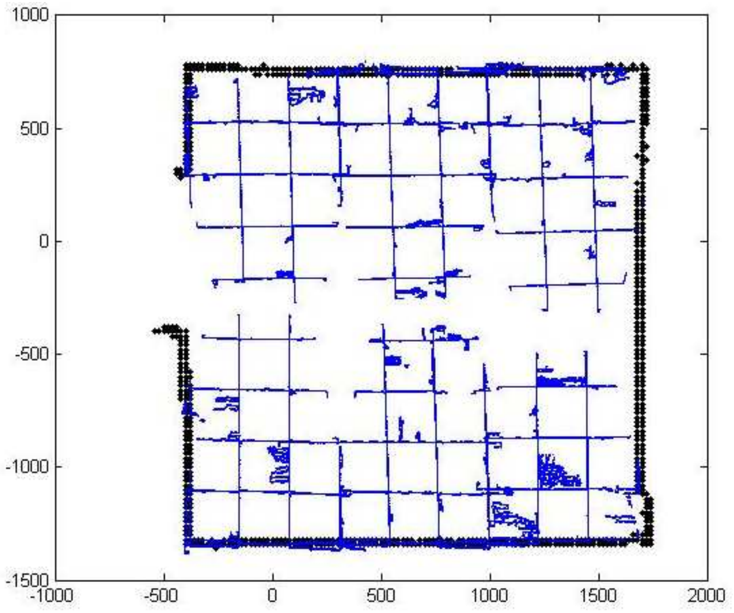


Figure 6.6: Crack map of the first experiment setup

6.2.2 Complex indoor environment

In the second environment, we apply the ROCIM system to detect curved cracks in the indoor environment. The hardware setup for this experiment is shown in Figure 6.7. Here, the robot collects 18 images to cover the whole inspection area in order to create the crack map. The procedure to create the crack map is the same as the previous experiment. The created 2D map of this environment is shown in Figure 6.8(a) and we utilize a chessboard for camera calibration as shown in Figure 6.8(b).

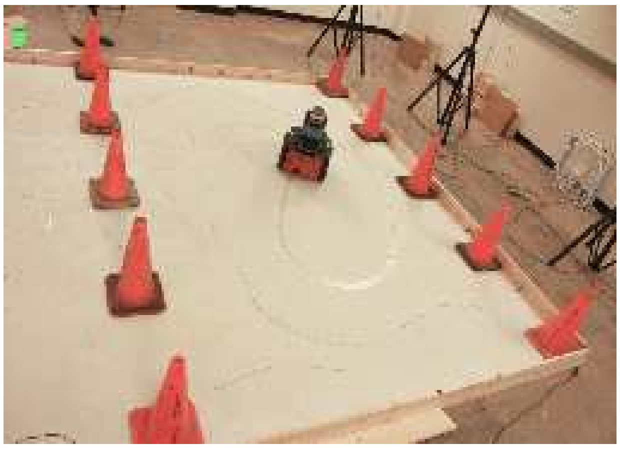
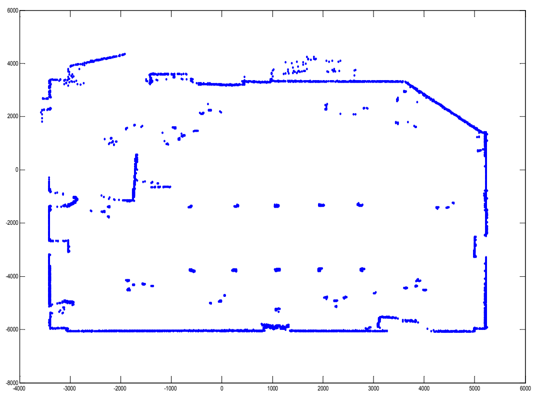
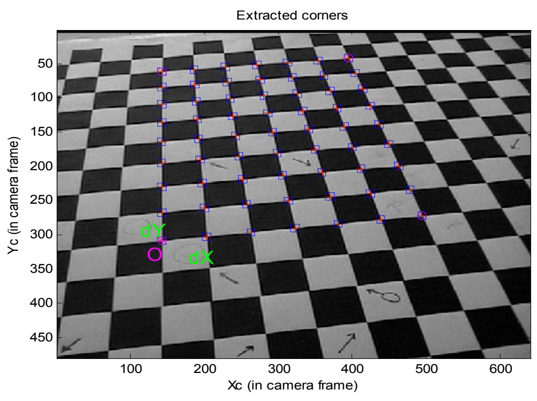


Figure 6.7: Experiment setup for complex indoor environment.

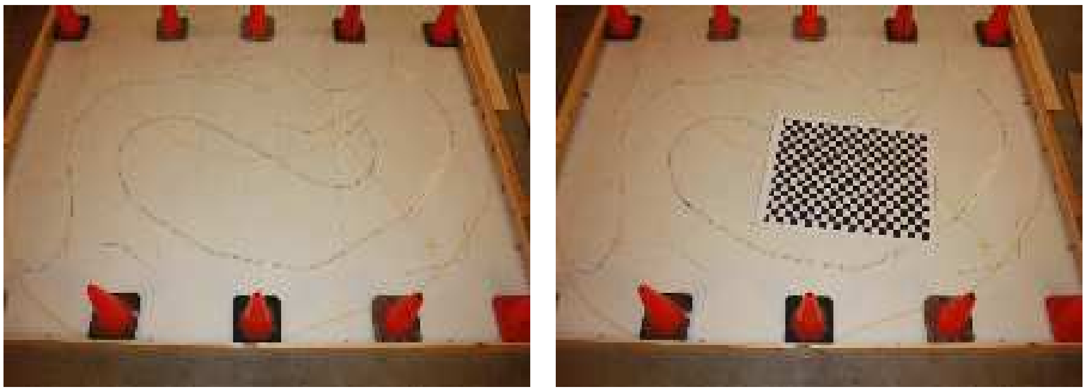


(a) A 2D map of the complex indoor environment.



(b) A chessboard for camera calibration.

Figure 6.8: Complex indoor environment.

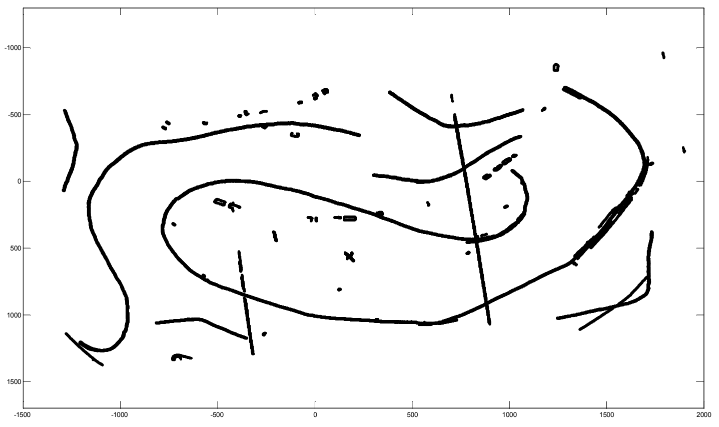


(a) A ground truth image for complex indoor environment. (b) Image for the camera calibration.

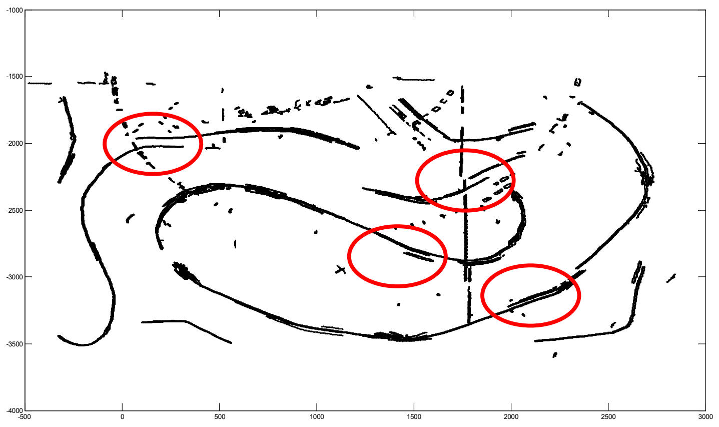
Figure 6.9: Ground truth of crack locations in the complex indoor environment.

To evaluate the crack map for this environment, we captured a single image which has all the crack information as shown in Figure 6.9(a). In order to create this ground truth map, we detect the cracks on this image and transform them to the global coordinate system. The transformation is calculated through the camera calibration and Figure 6.9(b) shows the calibration image.

We compare the result of the crack map with the ground truth map as shown in Figure 6.10. Figure 6.10(a) shows the ground truth map of cracks in the global coordinate system. Figure 6.10(b) shows the crack map created by the ROCIM system. The result shows some misalignments of the crack locations in the crack map. This is mostly due to camera calibration and robot localization errors. We also see some anomalies appear in Figure 6.10(b) that are not shown in Figure 6.10(a). This is due to different thresholds of crack detection algorithm. Figure 6.11 shows the overall crack map. The black line and blue lines represent the crack and the 2D map, respectively.



(a) The ground truth of the crack in the complex indoor environment.



(b) Cracks detected by the ROCIM system.

Figure 6.10: Comparison of the crack map with the ground truth.

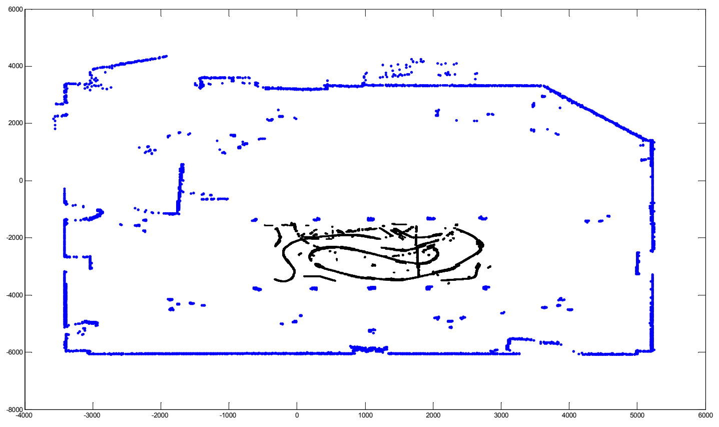


Figure 6.11: The global crack map.

6.2.3 Outdoor environment

In this experiment, we deploy the mobile robot to detect the cracks on a concrete surface. The experimental setup for outdoor test is shown in Figure 6.12. Here, the ROCIM system has to handle an open environment and moving obstacles such as walking people. The main problems with this experiment are poor robot localization and shadows. The robot localization problem is caused by the limited sensing range of the laser sensor which is not sufficient to scan the whole area and the dynamic environment which gives some noises to the MCL algorithm.

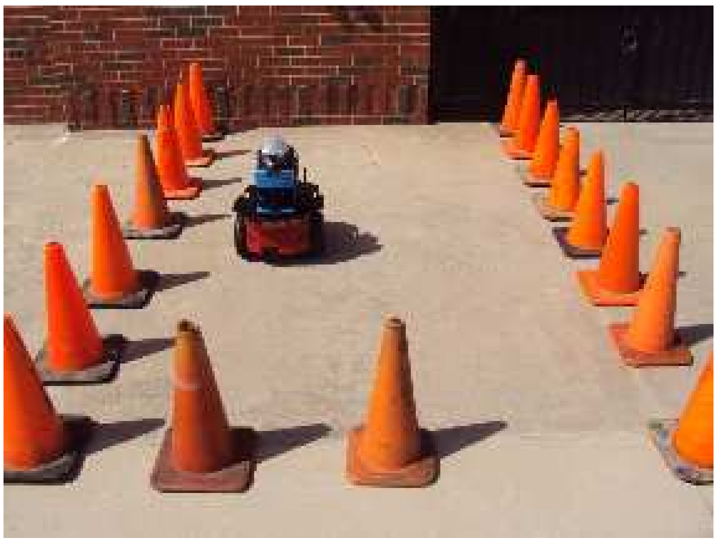
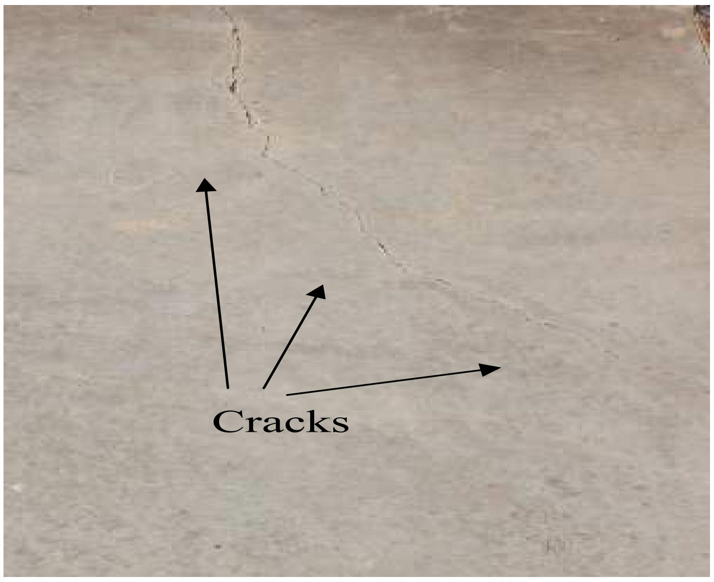
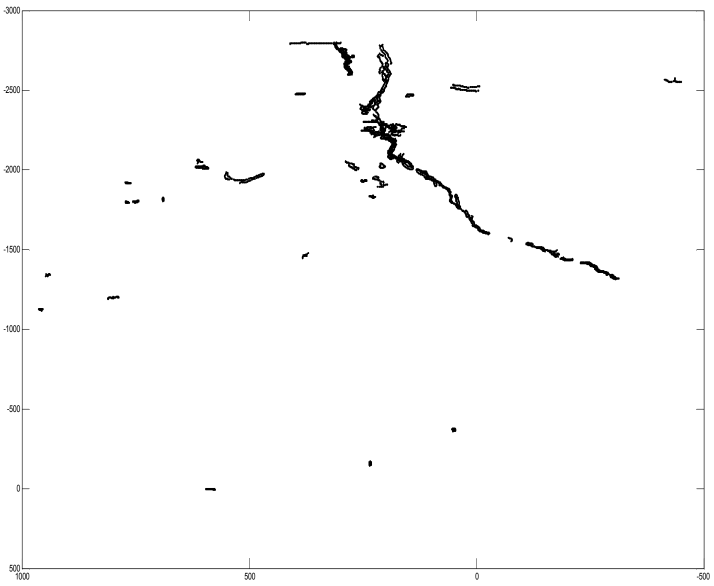


Figure 6.12: Experiment setup to evaluate the ROCIM system in the outdoor environment.

The methodology to create the crack map is the same as in the indoor experiment. We collect 10 images to cover the whole environment. Figure 6.13(a) shows the crack image in this environment and the crack map is shown in Figure 6.12. In this result, we do not evaluate the crack inspection in the shadow area and the misalignments of the crack map are still occurred.



(a) Cracks in the outdoor environment.



(b) Crack map for outdoor environment.

Figure 6.13: Outdoor experiment result.

6.3 Complete coverage path planning

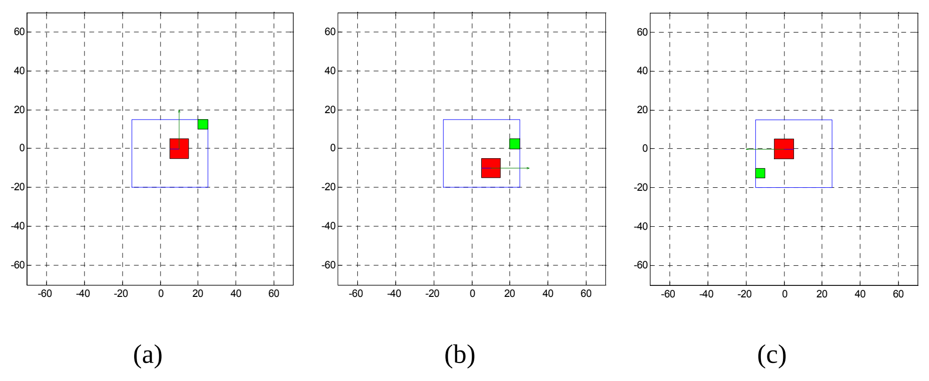
In this section we test our proposed robotic inspection path planning based on genetic algorithm (RIP-GA) in both simulations and real experiments. In simulation, we implement our algorithm in both obstacle and obstacle-free environments. In the real experiment, we implement our algorithm using the real mobile robot, Pioneer3-DX.

6.3.1 Obstacle-free environment

We define an environment without obstacles as in Figure 6.14(a). First, the camera path is planned using boustrophedron motion from top right to bottom left. There are 55 cells in this environment and each cell has to be covered in order to have complete coverage. There are several parameters that need to be initialized before we run the RIP-GA algorithm, such as the number of population M = 500, the weight parameters $\alpha = 0.94$, $\beta = 1.1$, and the stopping condition $Q = 30$. The values of α and β are found by measuring the time require for traveling a unit distance and making the robot to do a 90^0 turns.

For each generation, we discard 50% of the population which has the least rank and introduce new chromosomes to fill in. We run the RIP-GA algorithm to find the sub-optimal solution of the inspection path. After certain number of iterations, the RIP-GA algorithm converges at 59.15. The performance of the RIP-GA algorithm for each generation is shown in Figure 6.15. Moreover, the RIP-GA algorithm finds more than one solutions that have the same fitness value and they are shown in Tabel 6.1.

We compare the result of the RIP-GA algorithm with the RIP-greedy algorithm as shown in Table 6.2. Here, the result of the RIP-GA algorithm is slightly better than that of the RIP-greedy algorithm in terms of the number of robot turns, traveling distance and fitness value. The snapshots of the RIP-GA algorithm in the obstacle-free environment are shown in Figure 6.14.



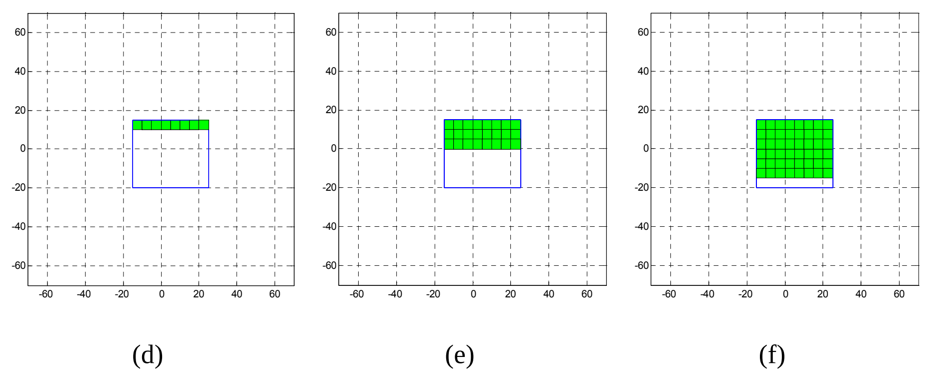


Figure 6.14: Snapshots of the simulation in the obstacle-free environment: (a), (b) and (c) are the snapshot of the mobile robot position, (d), (e) and (f) are the snapshot of the covered camera view.

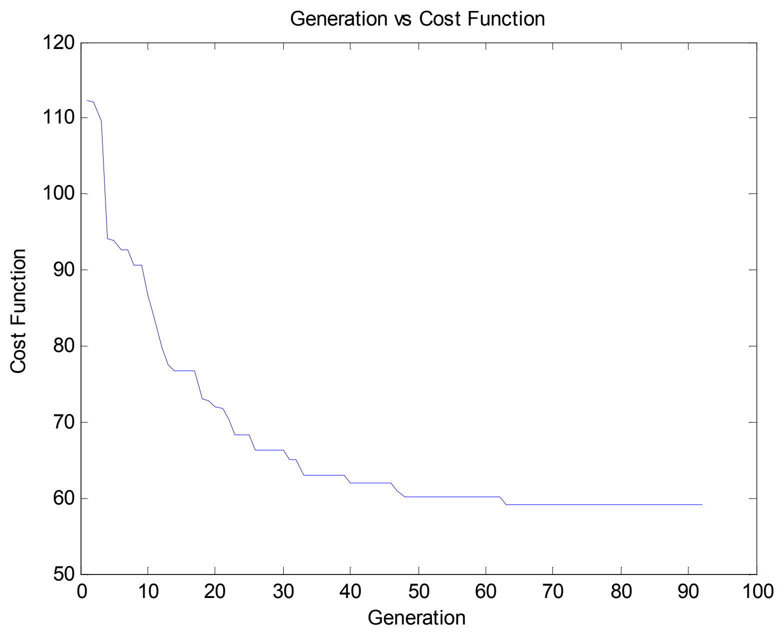


Figure 6.15: Performance of RIP-GA algorithm for obstacle-free environment.

Table 6.1: Several solutions with the minimum fitness value

CoIs
LLLLBDDCDDBLLLLALLLBDDDDFFHJJJJIJJJHFFFGGFFFHJJIIJJJHFFF
LLLLLBDDCDDBLLLAALLBDDDCFFHJJJJIJJJJHFFFFFFHJJJJIIJJJHFFF
LLLLBDDCDDBLLLLLLLLLBDDCFFHJJJJJJJJHFFFGFFFFHJJJJJJJHFFF

Table 6.2: Comparison of complete coverage path planning

	RIP-GA algorithm	RIP-Greedy algorithm
The number of robot turns	26	26
Traveling distance	325	395
fitness value	59.15	65.73

6.3.2 Obstacle environment

In this experiment, we evaluate the proposed algorithm in the obstacle environment as shown in Figure 6.16. Here, the obstacles are described by red squares. First, we decompose the environment into eight regions. Then, the camera path is planned using boustrophedron motion from bottom to top for each region. For each region, we indicate the starting and ending points by gray and black bullets, respectively. In this experiment, the

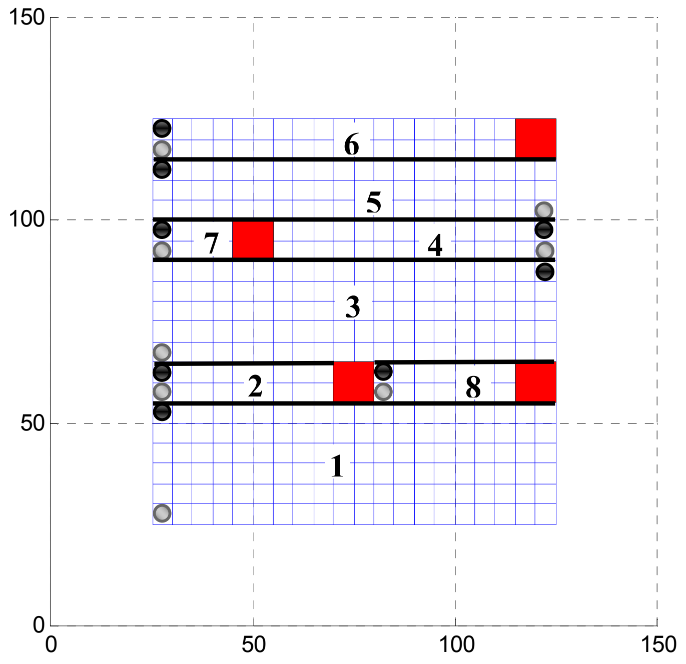


Figure 6.16: Map decomposition for obstacle environment.

initialization parameters are the same as those in the obstacle-free environment except the number of population (M). Since the obstacle environment is decomposed into several regions, we run the RIP-GA algorithm separately for each region. We use different values of M due to the different sizes of the region. We use M = 1000 for region 1 and 3, and M = 500 for the rest of regions.

The RIP-GA algorithm initializes the search with a random sample for each cell. The random sample is basically CoIs. In the obstacle environment, several CoIs can't be used to cover the cells which locate in the corner or around the boundary. Therefore, we apply a CoI filtering to remove these infeasible CoIs. In Table 6.3, we show the result of the CoI filtering for the first region. The subscript indicates the order of camera path and the letters indicate the feasible CoIs. For the first region, the RIP-GA generates the random population based on this table and the RIP-greedy calculates the least fitness value for each CoI according to this table too. We run the RIP-GA and RIP-greedy algorithm to find

Table 6.3: Feasible CoIs in region 1.

IJ_{1}	IJK_{21}	IJK_{41}	$AIJKL_{61}$	$AIJKL_{81}$	$AIJKL_{101}$
HIJ_{2}	$HIJK_{22}$	$HIJK_{42}$	$ABHIJKL_{62}$	$ABHIJKL_{82}$	$ABHIJKL_{102}$
HIJ_{3}	$HIJK_{23}$	$HIJK_{43}$	$ABHIJKL_{63}$	$ABHIJKL_{83}$	$ABHIJKL_{103}$
$FGHIJ_{4}$	$EFGHIJK_{24}$	$EFGHIJK_{44}$	$ABCDEFGHIJKL_{64}$	$ABCDEFGHIJKL_{84}$	$ABCDEFGHIJKL_{104}$
$FGHIJ_{5}$	$EFGHIJK_{25}$	$EFGHIJK_{45}$	$ABCDEFGHIJKL_{65}$	$ABCDEFGHIJKL_{85}$	$ABCDEFGHIJKL_{105}$
$FGHIJ_{6}$	$EFGHIJK_{26}$	$EFGHIJK_{46}$	$ABCDEFGHIJKL_{66}$	$ABCDEFGHIJKL_{86}$	$ABCDEFGHIJKL_{106}$
$FGHIJ_{7}$	$EFGHIJK_{27}$	$EFGHIJK_{47}$	$ABCDEFGHKL_{67}$	$ABCDEFGHKL_{87}$	$ABCDEFGHL_{107}$
$FGHIJ_{8}$	$EFGHIJK_{28}$	$EFGHIJK_{48}$	$ABCDEFGHKL_{68}$	$ABCDEFGHKL_{88}$	$ABCDEFGHL_{108}$
$FGHIJ_{9}$	$EFGHIJK_{29}$	$EFGHIJK_{49}$	$ABCDEFGKL_{69}$	$ABCDEFGKL_{89}$	$ABCDEFGL_{109}$
$FGHIJ_{10}$	$EFGHIJK_{30}$	$EFGHIJK_{50}$	$ABCDEFGIJKL_{70}$	$ABCDEFGIJKL_{90}$	$ABCDEFGIJKL_{110}$
$FGHIJ_{11}$	$EFGHIJK_{31}$	$EFGHIJK_{51}$	$ABCDEFGIJKL_{71}$	$ABCDEFGIJKL_{91}$	$ABCDEFGIJKL_{111}$
$FGHIJ_{12}$	$EFGHIJK_{32}$	$EFGHIJK_{52}$	$ABCDEIJKL_{72}$	$ABCDEIJKL_{92}$	$ABCDIJKL_{112}$
$FGHIJ_{13}$	$EFGHIJK_{33}$	$EFGHIJK_{53}$	$ABCDEHIJKL_{73}$	$ABCDEHIJKL_{93}$	$ABCDHIJKL_{113}$
$FGHIJ_{14}$	$EFGHIJK_{34}$	$EFGHIJK_{54}$	$ABCDEHIJKL_{74}$	$ABCDEHIJKL_{94}$	$ABCDHIJKL_{114}$
$FGHIJ_{15}$	$EFGHIJK_{35}$	$EFGHIJK_{55}$	$ABCDEFGHIJKL_{75}$	$ABCDEFGHIJKL_{95}$	$ABCDEFGHIJKL_{115}$
$FGHIJ_{16}$	$EFGHIJK_{36}$	$EFGHIJK_{56}$	$ABCDEFGHKL_{76}$	$ABCDEFGHKL_{96}$	$ABCDEFGHL_{116}$
$FGHIJ_{17}$	$EFGHIJK_{37}$	$EFGHIJK_{57}$	$ABCDEFGHKL_{77}$	$ABCDEFGHKL_{97}$	$ABCDEFGHL_{117}$
FGH_{18}	$EFGH_{38}$	$EFGH_{58}$	$BCDEFG_{78}$	$BCDEFG_{98}$	$BCDEFG_{118}$
FGH_{19}	$EFGH_{39}$	$EFGH_{59}$	$BCDEFG_{79}$	$BCDEFG_{99}$	$BCDEFG_{119}$
FG_{20}	EFG_{40}	EFG_{60}	$CDEFG_{80}$	$CDEFG_{100}$	$CDEFG_{120}$

out the best combination of CoIs to generate the inspection path. The comparison of both algorithms based on the fitness function can be seen in Table 3. We find that the RIP-GA algorithm gives better solution than the RIP-greedy algorithm in term of fitness value. The snapshots of the RIP-GA algorithm in the obstacle environment are shown in Figure 6.17.

Table 6.4: Comparison of RIP-GA and RIP-greedy algorithm

Region	RIP-GA algorithm	RIP-Greedy algorithm
1	84.3600	88.1300
2	10.190	11.7600
3	70.0900	76.0600
4	14.8900	16.4600
5	47.4250	53.0000
6	19.1200	23.8300
7	5.4900	7.0600
8	8.3100	9.8800

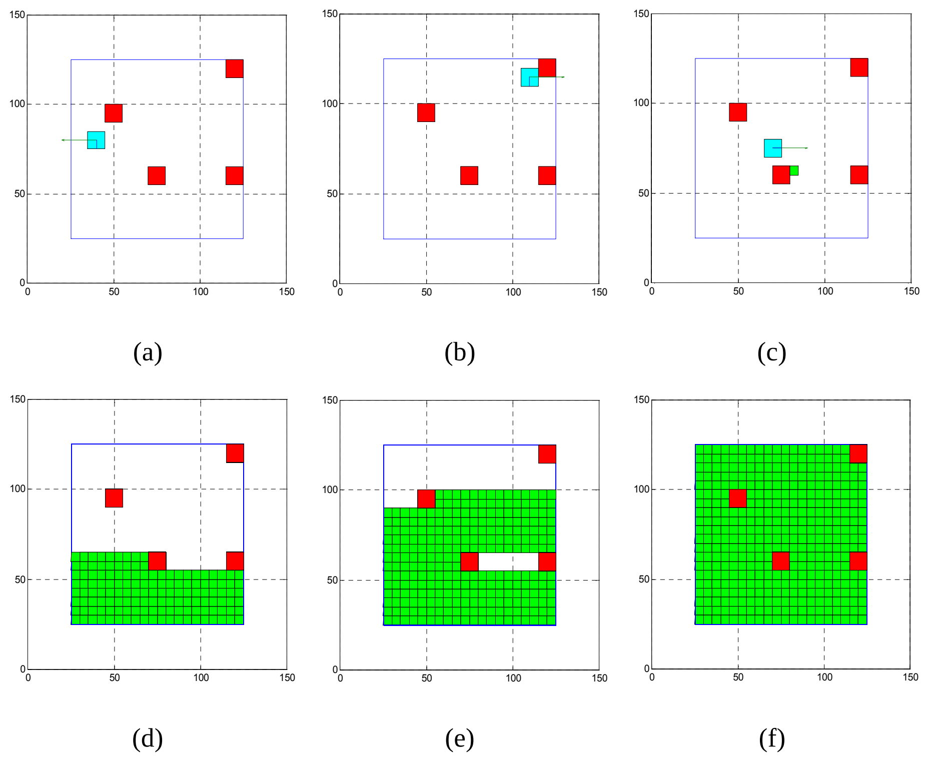


Figure 6.17: Snapshots of the simulation in the obstacle environment: (a), (b) and (c) are the snapshot of covered camera view, (d), (e) and (f) are the snapshot of robot position.

6.3.3 Real robot implementation

In the simulation, the RIP-GA algorithm gives a satisfactory result to solve the CCPP problem. In this section we discuss how to implement the proposed algorithm on the real robot, Pioneer3-DX. The result of the RIP-GA algorithm is an inspection path. This path is a sequences of robot locations in the 2D map with a particular robot and camera orientation. To implement this, the robot needs to travel from one location to another according to the inspection path.

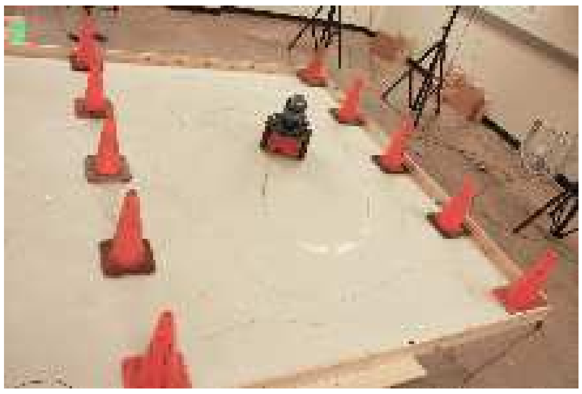


Figure 6.18: Laboratory environment to test the path planning in a real robot.

The laboratory environment for this experiment is shown in Figure 6.18. We use cones to set the inspection area. The procedure of the implementation is described as follows: First, the mobile robot creates a 2D map of this environment using the ARNL and Mapper3 software [2]. Second, we initialize the RIP-GA algorithm by defining the size and the number of cells in the 2D map. For this experiment, the inspection area is decomposed into 28 cells and the size of each cell is 550×550 mm. Third, we run the RIP-GA algorithm to find the sub-optimal of inspection path. Here, the RIP-GA algorithm uses the same set of CoIs as in the simulation. Figure 6.19 shows the result of the RIP-GA algorithm which consists of robot locations (red star) and center of FoV locations (blue star). Last, the robot path is given to the mobile robot to follow using navigation library [2]. For each

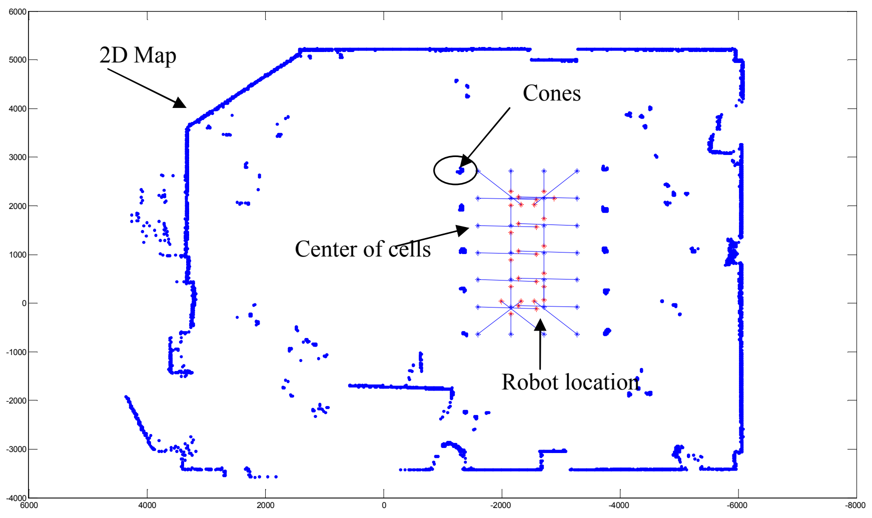


Figure 6.19: Robot locations for each cell in the 2D map.

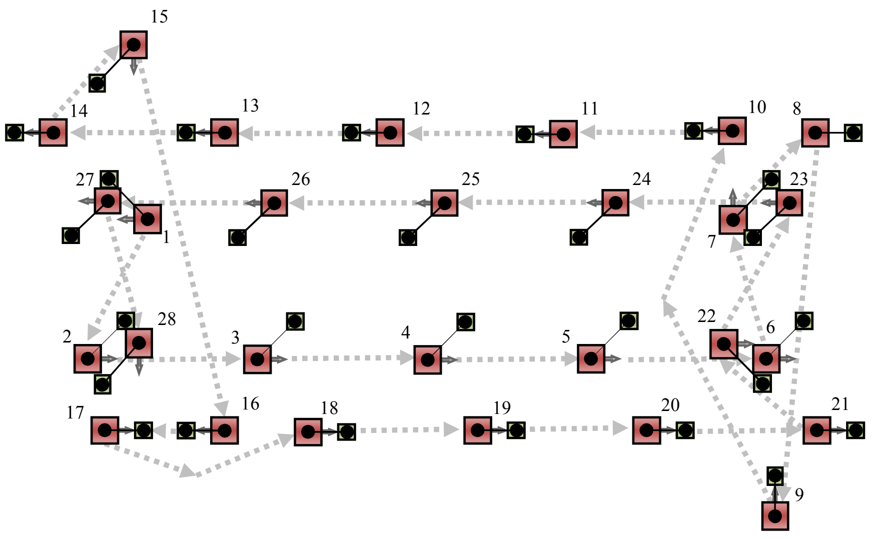


Figure 6.20: Illustration of the inspection path.

location in the robot path, the robot collects the image as illustrated in Figure 6.20. The number indicates the sequence of CoIs which is the sub-optimal solution from the RIP-GA algorithm. Here, the complete coverage of FoV can be guaranteed because the robot localization is very good for indoor environment.

CHAPTER 7

CONCLUSIONS AND FUTURE WORK

In this thesis, we introduce a robotic crack inspection and mapping (ROCIM) system to replace human inspectors for bridge deck crack inspection. The aim of the ROCIM system is to create a crack map which is useful for measuring, classifying, and analysis of crack growth. Three main components of the ROCIM system are presented in this thesis: camera calibration, crack detection and complete coverage path planning. We verify the proposed framework of the ROCIM system with both simulation and experimental result.

The Laplacian of Gaussian algorithm is used for the crack detection. After the crack is successfully detected in the image coordinate system, we apply a transformation matrix which is derived from camera calibration and robot localization, to map the crack locations to the global coordinate system as a crack map. We tested the ROCIM system in both indoor and outdoor environment, and the result shows that the ROCIM system works well. However, there is an alignment error of the crack location due to camera calibration and robot localization error.

For the complete coverage path planning, we introduce the robotic inspection path planning based on the genetic algorithm (RIP-GA) to ensure the mobile robot collects all the images in an efficient way. In simulation, the RIP-GA algorithm proves to be better than the RIP-greedy algorithm in terms of robot turns, traveling distance and inspection time. In addition, we implement the proposed algorithm on the real mobile robot, Pioneer3-DX. The implementation result shows the reliability and robustness of the proposed algorithm.

In the future work, we will improve the ROCIM system. Some of the improvements are as follows:

- Crack detection algorithm: We will improve the image processing techniques under different lighting conditions.
- Alignment process: The purpose of this process is to minimize the misalignment of crack locations. This process will improve the accuracy of crack map.
- On-line complete coverage path planning: The proposed RIP-GA algorithm is developed for static environment and is an off-line path planning algorithm. To deploy the ROCIM system in the passing traffic, we need to develop an on-line path planning for dynamic environments.
- Multi-robot cooperation: We will develop a multi-robot cooperation algorithm to perform the inspection in order to save more time for the crack inspection.

BIBLIOGRAPHY

[1] http://revlu.com/brg.htm/.

[2] http://www.mobilerobots.com/.

[3] J. Heikkil, "Geometric camera calibration using circular control points," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, pp. 1066–1077, 2000.

[4] "http://www.vision.caltech.edu/bouguetj/calib_doc,"

[5] S. N. Yu, J. H. Jang, and C. S. Han, "Auto inspection system using a mobile robot for detecting concrete cracks in a tunnel," *Automation in Construction*, vol. 16, pp. 255–261, 2007.

[6] Wikipedia, "http://en.wikipedia.org/wiki/i-35w mississippi river bridge," 2007.

[7] C. R. Farrar, H. Sohn, and S. W. Doebling, "Structural health monitoring at los alamos national laboratory," *The 13th International Congress and Exhibition on Condition Monitoring and Diagnostic Engineering Management (COMADEM 2000)*, 2000.

[8] H. Sohn, C. R. Farrar, M. L. Fugate, , and J. J. Czarnecki., "Structural health monitoring of welded connections," *The First International Conference on Stell and Composite Structures*, 2001.

[9] E. Sazonov, K. Janoyan, and R. Jha., "Wireless intelligent sensor network for autonomous structural health monitoring," *Proceedings of the SPIE - The International Society for Optical Engineering*, vol. 5384(1), pp. 305–314, 2004.

[10] N. Xu, S. Rangwala, K. K. Chintalapudi, D. Ganesan, A. Broad, R. Govindan, and D. Estrin, "A wireless sensor network for structural monitoring," *In SenSys'04*, 2004.

[11] D. R. Huston, “Adaptive sensors and sensor networks for structural health monitoring,” *Proceedings of SPIE - The International Society for Optical Engineering*, vol. 4512, pp. 203–211, 2001.

[12] S. K. Sinha and P. W. Fieguth, “Automated detection of cracks in buried concrete pipe images,” *Automation in Construction*, vol. 15, pp. 58–72, 2006.

[13] P. C. Tung, Y. R. Hwang, and M. C. Wu, “The development of a mobile manipulator imaging system for bridge crack inspection,” *Automation in Construction*, vol. 11, pp. 717–729, 2002.

[14] J. H. Lee, J. M. Lee, J. W. Park, and Y. S. Moon, “Efficient algorithms for automatic detection of cracks on a concrete bridge,” *The 23rd International Technical Conference on Circuits/Systems, Computers and Communications*, pp. 1213–1216, 2008.

[15] J. K. Oh, G. Jang, S. Oh, J. H. Lee, B. J. Yi, Y. S. Moon, J. S. Lee, and Y. Choi, “Bridge inspection robot system with machine vision,” *Automation in Construction*, vol. 18, pp. 929–941, 2009.

[16] H. G. Sohn, Y. M. Lim, K. H. Yun, and G. H. Kim, “Monitoring crack changes in concrete structures,” *Computer-Aided Civil and Infrastructure Engineering*, vol. 20, pp. 52–61, 2005.

[17] A. Habib and D. Kelley, “Single photo resection using the modified hough transformation,” *Photogrammetric Engineering and Remote Sensing*, vol. 67, no. 8, pp. 909–914, 2001.

[18] A. Ito, Y. Aoki, and S. Hashimoto, “Accurate extration and measurement of fine cracks from concrete block surface image,” *Proceedings IECON2002*, pp. 77–82, 2002.

[19] H. Choset, “Coverage of known spaces: the boustrophedon cellular decomposition,” *Autonomous Robots*, vol. 9, pp. 247–253, 2000.

[20] H. Choset and P. Pignon, “Coverage path planning: The boustrophedon decomposition,” *Proceedings of the International Conference on Field and Service Robotics*, 1997.

[21] M. Kapanoglu, M. Ozkan, A. Yazici, and O. Parlaktuna, “Pattern-based genetic algorithm approach to coverage path planning for mobile robots,” vol. 5544, pp. 33–42, 2009. 10.1007/978-3-642-01970-8_4.

[22] P. A. Jimenez, B. Shirinzadeh, A. Nicholson, and G. Alici, “Optimal area covering using genetic algorithms,” *Proceedings of the 2007 IEEE /ASME International Conference on Advanced Intelligent Mechatronics*, pp. 1–5, 2007.

[23] S. Yang and C. Luo, “A neural network approach to complete coverage path planning,” *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 34, pp. 718 – 724, February 2004.

[24] A. L. Hodgkin and A. F. Huxley, “A quantitative description of membrane current and its application to conduction and excitation in nerve,” *J. Physiol. Lond*, vol. 117, pp. 500–544, 1952.

[25] S. Thrun, D. Fox, W. Burgard, and F. Dellaert, “Robust monte carlo localization for mobile robots,” 2001.

[26] S. Thrun, “Simultaneous localization and mapping,” *Robotics and Cognitive Approaches to Spatial Mapping*, vol. 38, pp. 113–41, 2008.

[27] D. A. Forsyth and J. Ponce, “Computer vision: A modern approarch,” *Prentice Hall*, 2003.

[28] R. S. Lim, H. M. La, Z. Shan, and W. Sheng, “Developing a crack inspection robot for bridge maintenance,” *Proceedings of IEEE International Conference on Robotics and Automation*, 2011.

[29] D. Fox, W. Burgard, F. Dellaert, and S. Thrun, “Monte carlo localization: Efficient position estimation for mobile robots,” *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, 1999.

[30] http://www.naturalpoint.com/.

[31] D. A. Forsyth and J. Ponce, “Computer vision: A modern approarch,” *Prentice Hall*, 2003.

[32] Y. Mao, L. Dou, J. Chen, H. Fang, H. Zhang, and H. Cao, “Combined complete coverage path planning for autonomous mobile robot in indoor environment,” in *Asian Control Conference, 2009. ASCC 2009. 7th*, pp. 1468 –1473, aug. 2009.

[33] T. Oksanen and A. Visala, “Coverage path planning algorithms for agricultural field machines,” *Journal of Field Robotics*, vol. 26, pp. 651–668, 2009.

[34] Wikipedia, “http://en.wikipedia.org/wiki/genetic_algorithm,”

[35] S. A. I. Abuhadrous and F. Nashashibi, “Digitization and 3d modeling of urban environments and roads using vehicle-borne laser scanner system,” *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2004.

[36] S. I. Schwartz, “Catch me, i’m falling,” *New York Times*, 2007.

[37] V. Giurgiutiu, C. A. Rogers, Y. J. Chao, M. A. Sutton, , and X. Deng, “Adaptive health monitoring concepts for spot-welded andweld-bonded structural joints,” *Proceedings of the ASME Aerospace Division*, vol. 54, pp. 99–104, 1997.

[38] G. W. Reich and K. C. Park, “A use of substructural transmission zeros for monitoring,” *AIAA Journal*, vol. 38, pp. 1040–1046, 2000.

[39] Z. L. Cao, Y. Y. Huang, and E. L. Hall, “Region filling operations with random obstacle avoidance for mobile robots,” *J. Robotic Syst.*, pp. 87–102, 1988.

[40] E. M. Arkin and H. Refael, “Approximation algorithms for the geometric covering salesman problem,” *Discrete Applied Mathematics*, vol. 55, pp. 197–218, 1994.

[41] H. Choset, “Coverage for robotics.,” *Annals of Mathematics and Artificial Intelligence*, vol. 31, pp. 113–126, 2001.

[42] C. Luo and S. Yang, “A bioinspired neural network for real-time concurrent map building and complete coverage robot navigation in unknown environments,” *IEEE Transactions Neural Networks*, vol. 19, pp. 1279–1298, 2008.

[43] A. Zelinsky, R. A. Jarvis, J. C. Byrne, and S. Yuta, “Planning paths of complete coverage of an unstructured environment by a mobile robot,” *Proceedings of International Conference on Advanced Robotics*, pp. 533–538, 1993.

[44] S. O. Joon, H. C. Yoon, B. P. Jin, and Y. F. Zheng, “Complete coverage navigation of cleaning robots using triangular-cell-based map,” *IEEE Trans. Ind. Electron.*, 2004.

[45] H. B. D. M. T. Hagan and M. Beale, “Neural network design,” *Boston: PWS Publishing Co.*, 1996.

[46] A. Hoover, V. Kouznetsova, and M. Goldbaum, “Locating blood vessels in retinal images by piecewise threshold probing of a matched filter response,” *Medical Imaging, IEEE Transactions on*, vol. 19, pp. 203 –210, march 2000.

[47] S. Rusinkiewicz and M. Levoy, “Efficient variants of the icp algorithm,” *3D Digital Imaging and Modeling, International Conference on*, vol. 0, p. 145, 2001.

VITA

Ronny Salim Lim

Candidate for the Degree of

Master of Sciences

Thesis: DEVELOPING A CRACK INSPECTION ROBOT FOR BRIDGE DECK MAINTENANCE

Major Field: Electrical and Computer Engineering

Biographical:

Personal Data: Jakarta, DKI-Jakarta, Indonesia on February 27th, 1986.

Education:
Received the B.S. degree from Pelita Harapan University, Tangerang, Jawa Barat, Indonesia, 2008, Electrical Engineering
Completed the requirements for the degree of Master of Science with a major in Electrical and Computer Engineering at Oklahoma State University in July, 2011.

Experience:
Research Assistant at Oklahoma State University, United States (01/2009 - 07/2011)
Teaching Assistant at Pelita Harapan University, Indonesia (01/2006 - 06/2008)

Professional Memberships:
Phi Kappa Phi Honor Society
Eta Kappa Nu Honor Society

Name: Ronny Salim Lim Date of Degree: July, 2011

Institution: Oklahoma State University Location: Stillwater, Oklahoma

Title of Study: DEVELOPING A CRACK INSPECTION ROBOT FOR BRIDGE DECK MAINTENANCE

Pages in Study: 58 Candidate for the Degree of Master of Science

Major Field: Electrical and Computer Engineering

One of the important tasks for bridge maintenance is bridge deck crack inspection. Traditionally, a human inspector detects cracks using his/her eyes and finds the location of cracks manually. Thus the accuracy of the inspection result is low due to the subjective nature of human judgement. We propose a system that uses a mobile robot to conduct the inspection, where the robot collects bridge deck images with a high resolution camera. In this method, the Laplacian of Gaussian algorithm is used to detect cracks and the crack map is obtained through camera calibration and robot localization. To ensure that the robot collects all the images on the bridge deck, we develop a complete coverage path planning algorithm for the mobile robot. We compare it with other path planning strategies. Finally, we validate our proposed system through experiments and simulation.

ADVISOR'S APPROVAL: ____________________