Detection and Localization of Linear Features Based on Image Processing Methods

Sean Cormick Matz
Claremont Graduate University

Detection and Localization of Linear Features based on Image Processing Methods

by

Sean Matz

Approval of the Dissertation Committee

This dissertation has been duly read, reviewed, and critiqued by the Committee listed below, which hereby approves the manuscript of Sean Cormick Matz as fulfilling the scope and quality requirements for meriting the degree of Doctor of Philosophy in Mathematics.

Marina Chugunova, Chair
Claremont Graduate University
Professor
Mathematics

John Angus
Claremont Graduate University
Professor
Mathematics

Ali Nadim
Claremont Graduate University
Professor
Mathematics

Abstract

In this work, the general problem of the detection of features in images is considered. One of the methods, the orientation detection of lines, utilized the Radon transform (sinogram) of an image to detect lines at different angles in an image. The line thickness algorithm was generated by finding a pattern formed by particular lines in an image. The filtering of reconstructed images dealt with the removal of blur and other artifacts that arose in the course of inverting the Radon transform of an image to attempt to obtain the original image.

In memory of my beloved parents Joseph and Audrey Matz

Acknowledgements

I would very much like to thank my dissertation advisor, Professor Marina Chugunova, for her forbearance with me and for patiently guiding and for patiently guiding me in the research and writing of this thesis with all of her very helpful suggestions. I would also like to thank the members of my dissertation committee, Professors John Angus and Ali Nadim for reviewing my thesis.

Contents

Abstract **iii**

Acknowledgements **v**

List of Figures **viii**

List of Tables **xiv**

1 Introduction **1**

2 Previous Work **5**

2.1 Filters Used Before or After Filtered Backprojection (FBP) 5

2.2 Non-Image Processing Methods Used for Crack Detection 6

2.3 Wavelet-Based Methods of Crack Detection 6

2.4 Radon Transform-Based Methods of Crack Detection 8

2.5 Methods of Crack Detection Utilizing Wavelet and Radon Transforms 8

2.6 Other Image Processing Methods for Analyzing Cracks 9

2.7 Percolation-Based Image Processing 10

2.8 The Use of the Radon Transform for Orientation Detection 11

3 Filtering the Reconstructed Image **13**

Grayscale Images 13

Color Images 19

4 Preliminary Results 21

Grayscale Images 21

Color Images 28

5 Crack Detection with the Radon Transform 35

6 Enhancement Using the Radon Transform 40

7 Use of Optical Flow for Warping One Edge-Detected Crack Image into Another 47

8 Image Processing Algorithms for Feature Detection 72

8.1 Crack Endpoint Detection 72

8.2 Detection of Crack Intersection Points 79

8.3 Orientation Detection of Lines 84

8.4 Determination of Line Thickness 94

8.5 Filtering of Reconstructed Images 104

8.5.1 Filtering of Reconstructed Color Images 125

9 Conclusions 136

List of Figures

3.1 The Four Subregions for the Kuwahara Filter 18

3.2 The Five Subregions for the Tomita-Tsuji Filter 19

3.3 Five Subregions of the Tomita-Tsuji 9x9 Filter 20

4.1 Input Image 22

4.2 Sinogram 22

4.3 Reconstructed Image with Ram-Lak Filter 23

4.4 Reconstructed Image with Shepp-Logan Filter 24

4.5 Ram-Lak Reconstructed Image after Processing with a 9x9 Tomita-Tsuji Filter . . . 25

4.6 Shepp-Logan Reconstructed Image after Processing with a 9x9 Tomita-Tsuji Filter . 26

4.7 Color Image Input into a Radon Transform 29

4.8 Reconstructed Color Image with RL Filter 30

4.9 Reconstructed Color Image with SL Filter 31

4.10 Kuwahara-Filtered RL Reconstructed Image 32

4.11 Kuwahara-Filtered SL Reconstructed Image 33

5.1 Input Image 001-105 37

5.2 Edge Map 38

5.3 Sinogram With Single Bright Point 39

6.1 The Mapping of Image Space to Feature Space under the Radon Transform 41

6.2 Line in Image Space 42

7.1 Input Image 002-100 of a Pavement Crack 50

7.2 Edge Map 52

7.3 Upper Edge 53

7.4 Lower Edge 54

7.5 Merged Upper and Lower Edges of Pavement Crack after Applying Optical Flow Warping 55

7.6 Image 7003-18 of a Bridge Deck Crack 56

7.7 Image of Upper Portion of Edge Map for Bridge Deck Crack 57

7.8 Image of Lower Portion of Edge Map for Bridge Deck Crack 58

7.9 Merged Upper and Lower Edges of Bridge Deck Crack After Applying Optical Flow Warping 59

7.10 Image 7070-24 of a Wall Crack 60

7.11 Image of Upper Portion of Edge Map for Wall Crack 61

7.12 Image of Lower Portion of Edge Map for Wall Crack 62

7.13 Merged Upper and Lower Edges of Wall Crack After Applying Optical Flow Warping 63

7.14 Image 004-114 of a Pavement Crack 64

7.15 Image of Upper Portion of Edge Map for Pavement Crack 65

7.16 Image of Lower Portion of Edge Map for Bridge Deck Crack 66

7.17 Merged Upper and Lower Edges of Pavement Crack After Applying Optical Flow Warping 67

7.18 Image 7002-28 of a Bridge Deck Crack 68

7.19 Image of Upper Portion of Edge Map for Bridge Deck Crack 69

7.20 Image of Lower Portion of Edge Map for Bridge Deck Crack 70

7.21 Merged Upper and Lower Edges of Bridge Deck Crack After Applying Optical Flow Warping 71

8.1 Input Image 7070-24-det-up.jpg 74

8.2 Input Image of figure 8.1 with Horizontal and Vertical Axes Added 75

8.3 Input Image 7070-24-det-low 76

8.4 Input Image of figure 8.3 with Horizontal and Vertical Axes Added 76

8.5 Input Image 7003-18-det-up.jpg 77

8.6 Input Image in figure 8.5 with Horizontal and Vertical Axes Added 78

8.7 Input Image 7003-18-det-low.jpg 78

8.8 Input Image in figure 8.8 with Horizontal and Vertical Axes Added 79

8.9 Input Image 7070-24-det-clean.jpg 80

8.10 Input Image in figure 8.9 with Horizontal and Vertical Axes Added 81

8.11 Input Image 7003-18-det-clean.jpg 82

8.12 Input Image in figure 8.11 with Horizontal and Vertical Axes Added 83

8.13 Input Image for Orientation Detection Algorithm 85

8.14 Sinogram Plot for Threshold Determination 85

8.15 Input Image with Lines at 14 Different Angles 86

8.16 Histogram of Maximum Sinogram at each Angle for 14 Input Angles 86

8.17 The Number of Lines per Angle for 14 Angles 87

8.18 Input Image Showing Lines at 35 Different Angles 87

8.19 Histogram of Maximum of Sinogram at Each Angle for 35 Input Angles 88

8.20 Sinogram of 35 Lines at 35 Angles Showing the Peaks of the 35 Detected Lines . . . 88

8.21 The Number of Lines per Angle for 35 Angles 89

8.22 Input Image with Lines at Eight Different Angles 90

8.23 Sinogram Corresponding to Image with Lines at Eight Different Angles 90

8.24 Image with 10 Lines at 25 degrees and 5 lines at 60 degrees, with angles measured with respect to the vertical 91

8.25 Sinogram Showing Peaks at Two Different Angles 91

8.26 The Number of Lines at Each Angle for Two Different Angles 91

8.27 Image with 18 Lines at 6 Different Angles 92

8.28 Sinogram Associated with Image of 18 Lines at 6 Different Angles 92

8.29 Maximum Value of the Sinogram for 18 Lines 92

8.30 True Number of Lines at Each Angle for 18 Lines at 6 Different Angles 92

8.31 Computed Number of Lines at Each Angle for 18 Lines at 6 Different Angles 93
8.32 Depiction of a Horizontal Line 94
8.33 Depiction of a Vertical Line 95
8.34 Input Image with Eight Lines of Different Thicknesses 96
8.35 Input Image with Twelve Lines of Different Thicknesses 97
8.36 Input Image with 24 Lines of Different Thicknesses 98
8.37 Input Image with Seven Parallel Lines With Four Different Thicknesses 99
8.38 Sinogram Corresponding to the Input Image with Seven Parallel Lines with Four Different Thicknesses 99
8.39 The Number of Lines at Each Thickness for 8 Lines 100
8.40 The Number of Lines at Each Thickness for 12 Lines 101
8.41 The Number of Lines at Each Thickness for 18 Lines 101
8.42 The Number of Lines at Each Thickness for 24 Lines 102
8.43 Input Image with Two Sets of Parallel Lines, 13 Lines Total 103
8.44 Sinogram Corresponding to the Input Image with Two Sets of Parallel Lines, 13 Total Lines 103
8.45 Bar Plot of the True Line Width Distribution for Parallel Lines at 2 Different Angles 103
8.46 Bar Plot of the Computed Line Width Distribution for Parallel Lines at 2 Different Angles 103
8.48 Example of Changing Windows in All Four Subregions 105
8.47 Example of Changing Window Size in a Subregion 105
8.49 Reconstructed Image Using Filtered Back Projection with a Ram-Lak Filter 106
8.50 Filtered RL Reconstructed Image Using 9x9 Tomita-Tsuji Filter with IQR and Mean 107
8.51 Adaptive Kuwahara Applied to RL Reconstructed Image Using IQR and Mean . . . 108
8.52 Reconstructed Image Using Filtered Back Projection with a Shepp-Logan Filter . . . 109
8.53 Filtered SL Reconstructed Image Using 9x9 Tomita-Tsuji Filter with IQR and Mean 109
8.54 Adaptive Kuwahara Filter Applied to SL Reconstructed Image Using IQR and Mean 110
8.55 Reconstructed Image Using Filtered Back Projection with Ram-Lak Filter 111

8.56 Filtered RL Resconstructed Image Using 9x9 Tomita-Tsuji Filter with IQR 111
8.57 Filtered RL Reconstructed Image Using Adaptive Kuwahara Filter with IQR 112
8.58 Image Reconstructed with Filtered Back Projection and the Shepp-Logan Filter . . . 113
8.59 SL-Reconstructed Image Filtered with 9x9 Tomita-Tsuji Filter 114
8.60 SL-Reconstructed Image After Filtering with Adaptive Kuwahara Filter 114
8.61 Reconstructed Image Using Filtered Back Projection with Ram-Lak Filter 115
8.62 Filtered RL Reconstructed Image Using 9x9 Tomita-Tsuji Filter and IQR 116
8.63 RL- Reconstructed Image Filtered with Adaptive Kuwahara Filter with IQR 116
8.64 Reconstructed Image Using Filtered Back Projection and the Shepp-Logan Filter . . 117
8.65 SL-Reconstructed Image Filtered with 9x9 Tomita-Tsuji Filter with IQR 118
8.66 SL-Reconstructed Image Filtered with Adaptive Kuwahara Filter with IQR 118
8.67 Reconstructed Image Produced by Filtered Back Projection with Ram-Lak Filter . . 119
8.68 RL-Reconstructed Image Using 9x9 Tomita-Tsuji with IQR 120
8.69 RL-Reconstructed Image Filtered with Adaptive Kuwahara Filter with IQR 120
8.70 Reconstructed Image Produced by Filtered Back Projection with Shepp-Logan Filter 121
8.71 SL-Reconstructed Image Filtered with 9x9 Tomita-Tsuji Filter with IQR 122
8.72 SL-Reconstructed Image Filtered with Adaptive Kuwahara Filter with IQR 122
8.73 Reconstructed Balloons by Filtered Back Projection with the Ram-Lak Filter 125
8.74 RL Color Kuwahara with IQR 126
8.75 RL Adaptive Color 3x3 Kuwahara Filter Using IQR 126
8.76 Reconstructed Image Produced by Filtered Back Projection with Shepp-Logan Filter 127
8.77 SL-Reconstructed Image Filtered with 3x3 Kuwahara Filter with IQR 128
8.78 SL-Reconstructed Image Filtered with Adaptive Kuwahara Filter with IQR 129
8.79 Reconstructed Image Produced by Filtered Back Projection with Ram-Lak Filter . . 130
8.80 RL-Reconstructed Image with 3x3 Kuwahara Filter with IQR 131
8.81 RL-Reconstructed Image Filtered with Adaptive Kuwahara Filter with IQR 131
8.82 Reconstructed Image Produced by Filtered Back Projection with Shepp-Logan Filter 132
8.83 SL-Reconstructed Image Filtered with 3x3 Kuwahara Filter with IQR 133

8.84 SL-Reconstructed Image Filtered with Adaptive Kuwahara Filter with IQR 133

List of Tables

4.1 Total Variation Values for Grayscale Images 28

4.2 Total Variation Values for Color Images 34

8.1 Numerically Recovered Angles versus True Angles 90

8.2 True versus Detected Thickness for 7 parallel lines with 4 Thicknesses 99

8.3 Total Variation Values for Grayscale Images 123

8.4 Total Variation for Grayscale Images (Continued) 124

8.5 Total Variation Values for Color Images 134

Chapter 1

Introduction

Inverse Problems

According to Jacques Hadamard, a well-posed problem has three properties:

(1) (Existence) A solution to the problem exists.

(2) (Uniqueness) The problem has only one solution.

(3) (Stability) The solution depends in a continuous fashion on the data associated with the problem.

Unfortunately, almost all inverse problems in physics or biology are ill-posed. However, the problems of existence and uniqueness can often be addressed by considering a generalized solution and then placing constraints on it. Stability is often lacking in inverse problems.

Inverse problems involve determining an unknown quantity (a cause) associated with a particular object based upon measurements of the effects associated with this object. Therefore, the concepts in inverse problems give rise to an underlying theory for remote sensing and non-destructive testing. For example, if a sound wave is scattered by a target, and if one collects the scattered far field signal, then the inverse problem is to determine the shape and physical characteristics of the target. These problems are important in the identification of airborne objects as well as those submerged in water.

An inverse problem for the earth is to find subterranean targets of interest based upon the analysis of scattered field data for different positions or frequencies. Examples of this include an oil reserve, a cave, or a mine. Another inverse problem is to find a fault or imperfection in a material by determining the natural frequencies of a sample of the material. Additionally, an inverse problem can involve the computation of an image from X-ray data, or the determination of the Earth's density based upon the collection of gravitational field data. If one can find the anomalies in an object by collecting and analyzing the scattered returns from signals sent toward the object, then one avoids invasive and potentially destructive testing.

Among the many inverse problems are inverse problems of potential theory, inverse spectral theory, inverse scattering problems in quantum physics, inverse problems in geophysics, inverse problems for the heat and wave equations, inverse obstacle scattering, finding small subsurface inhomogeneities from the measurements of the scattered field on the surface, inverse problem of radiomeasurements, impedance tomography (inverse conductivity) problem, and tomography and other integral geometry problems. Inverse problems in the area of tomography involve the use of the Radon transform. The Radon transform was discovered by the Austrian mathematician Johann Karl August Radon in 1917.

The Radon transform in two dimensions is the integral transform which takes a function f defined on the plane to a function Rf defined on the two-dimensional space of lines in the plane whose value at a particular line is equal to the line integral of the function over that line. The Radon transform represents a transformation from a function of rectangular coordinates x and y to a function of coordinates t and θ.

$$R(t,\theta) = \int_{R^2} f(x,y)\delta(t - x\cos\theta - y\sin\theta)dxdy$$

where

$$t = x\cos\theta + y\sin\theta$$

is the line along which f is integrated to produce the line integral which is R(t,θ).

Given the Radon transform of a function f, we can recover the original function f through the use of the Fourier transform.

Taking the Fourier transform of the above equation, we obtain

$$\begin{aligned}\hat{R}(\theta,\omega) &= \int_R \exp(-2\pi i\omega t)R(t,\theta)dt \\ &= \int\int\int_{R^3} f(x,y)\exp(-2\pi i\omega t)\delta(t-x\cos\theta-y\sin\theta)dtdxdy \\ &= \int\int_{R^2} f(x,y)\exp(-2\pi i\omega(x\cos\theta-y\sin\theta))dxdy\end{aligned}$$

after applying the sifting property of the Fourier Transform. Also, changing the order of integration to put dt first follows from Fubini's Theorem since f is bounded and measurable.

Now, the 2D Fourier Transform of a function f is given by

$$F(u,v)=\int\int_{R^2} f(x,y)\exp(-2\pi i(ux+vy))dxdy$$

The 2D inverse Fourier Transform is given by

$$f(x,y)=\int\int_{R^2} F(u,v)\exp(2\pi i(ux+vy))dudv$$

Converting from rectangular to polar coordinates with

$$u=\omega\cos\theta \qquad v=\omega\sin\theta$$

and replacing F(u,v) with $\hat{R}(\omega,\theta)$, we obtain

$$f(x,y)=\int\int_{R^2}\hat{R}(\omega,\theta)|\omega|\exp(2\pi i\omega(x\cos\theta+y\sin\theta)d\omega d\theta$$

The Fourier Slice theorem relates the Radon transform to the Fourier transform. The Fourier Slice Theorem [1] states that the one-dimensional Fourier transform of a projection $P_\theta(\rho)$ is equal to the two-dimensional Fourier transform of f(x,y) evaluated at an angle θ.

Proof: Taking the one-dimensional Fourier transform of the projection, we obtain:

$$\begin{aligned}
\hat{P}_\theta(\omega) = \mathcal{F}(g_\theta(\rho)) &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)\delta(\rho - x\cos\theta - y\sin\theta)\exp(-i2\pi\omega\rho)d\rho dx dy \\
\text{where} &\quad \text{changing the order of integration to put d}\rho\text{ first is justified by Fubini's Theorem} \\
\text{since} &\quad \text{f is bounded and measurable} \\
&= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)\exp(-i2\pi\omega(x\cos\theta + y\sin\theta))dxdy \\
\text{by} &\quad \text{applying the sifting property of the delta function} \\
&= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)\exp(-i2\pi(\omega x\cos\theta + \omega y\sin\theta))dxdy
\end{aligned}$$

Now, the two-dimensional Fourier transform of f(x,y) is given by

$$F(u,v) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)\exp(-i2\pi(ux + vy)dxdy$$

In polar coordinates, (u,v) becomes ($\omega\cos\theta$,$\omega\sin\theta$). Thus,

$$F(u,v)\Big|_{u=\omega\cos\theta, v=\omega\sin\theta} = \hat{P}_\theta(\omega)$$

In the proof above, if we assume that f represents an image in the plane, then f is bounded since values of f are constrained to lie in the interval [0,255]. Now f(x,y) is measurable if {(x,y) — f(x,y) < r } ∀ r ∈ R is measurable. For r< 0, the above set has measure zero and is thus measurable. For r>255, the above set is the entire image which is measurable and has measure m×n where the image size is mxn. For 0<r<255, the above set is some portion of the image and is therefore some percentage of m×n and is thus measurable.

Applications of the Radon reconstruction include stress analysis, geophysics, air pollutant studies, nondestructive testing, pattern recognition, communications, and computerized geophysical tomography (CGT), computed axial tomography (CAT), barcode scanners, electron microscopy, and seismology.

Chapter 2

Previous Work

2.1 Filters Used Before or After Filtered Backprojection (FBP)

Some of the authors employed filters during the process of performing backprojection, while others filtered the reconstructed images after FBP. Lyra and Ploussi, in their paper [2], discuss filters utilized during the process of performing filtered backprojection (FBP). In his paper [3], Demirkaya filters noise-degraded projection images using the nonlinear anisotropic diffusion filter. The images were reconstructed from the filtered projections using filtered backprojection (FBP). In their paper, Yang, Zhang, Huang, and Yang [4] employed a multiple sampling method in the projection domain for low-signal and noisy projections. Then, a fuzzy entropy-based method with a block matching 3D (BM3D) filtering algorithm was used to improve image quality by reducing artifacts and noise in the image domain. In their presentation and conference paper for the Radiological Society of North America 2012 Scientific Assembly and Meeting, members of the Mayo Clinic [5] discussed image-space denoising (ISD). They stated that linear or non-linear filters are directly applied to reconstructed images to remove noise. They didn't, however, describe the nature of the linear or non-linear filters used. In his algorithm which was awarded a patent by the U.S. Patent Office [6], Boas mentioned the use of an edge-preserving blur filter applied after filtered backprojection (FBP) and linear interpolation but before forward projection and another FBP. These steps are applied in an iterative fashion. Boas, however, did not discuss the nature of the edge-preserving smoothing filter used. The purpose was to remove metal artifacts from the image. It appears that this is the closest to the work that we explore in chapter 3.

2.2 Non-Image Processing Methods Used for Crack Detection

Some researchers made use of external data or other methods besides image processing methods for the detection of cracks.

[7] introduces a new optical method known as shearography. Shearography is an interferometric method which uses an image-shearing camera. It allows full-field measurement of derivatives of surface displacements. [8] describes a new method for crack detection in beams based on instantaneous frequency and empirical mode decomposition. In [9], two new methods for detecting a fatigue crack in a planet carrier of an epicyclic transmission are presented.

The authors, [10] of this paper, note that the process of locating and configuring a shaft crack is an inverse problem. They describe a genetic algorithm based method for shaft crack detection. This method describes the shaft crack detection problem as an optimization problem via the finite element method and employs genetic algorithms to search for the solution. The paper by [11] presents the recent progress of the phased array and electromagnetic inspection techniques. The authors, [12] of this paper, conduct an eigenvalue analysis using Lanczo's algorithm in an adaptive h-version finite element environment to control the discretization error for accurate evaluation of modal parameters.

2.3 Wavelet-Based Methods of Crack Detection

Some of the researchers utilized wavelet transforms on external data and some applied them to images. The former constituted non image-based methods. The paper [13] presents a method for estimating the damage location in beam and plate structures. The two-dimensional wavelet transform for plate damage detection is described. The location of the damage is determined by finding a peak in the spatial variation of the transformed response. Wavelet analysis can effectively identify the defect position without information concerning structure properties or a mathematical model. However, the technique appears to depend on displacement data for beams or plates. This is not an image-based method. The paper [14] discusses a structural damage detection technique based on wavelet analysis of spatially distributed structural response measurements. The method assumes that cracks in a structure will cause structural response perturbations at damage sites. This technique uses simulated deflection responses or smooth analytical crack-tip displacement fields. The deflection of the displacement response is analyzed with the wavelet transform. This is not an image-based method. The authors of [15] attempt to detect structural damage with spatial wavelets. The displacement response data along parallel and perpendicular lines at different positions from the crack are analyzed with the Haar wavelet. The peak of the spatial variation of the wavelets indicates the direction of the crack. This method uses simulated structural response data and is not an image-based technique. The paper [16] describes a method for crack identification of bridge beam structures under a moving load based on wavelet analysis. The response obtained at a single measuring point is analyzed using the continuous wavelet transform, and the location of cracks is estimated. This is not an image-based technique. The paper [17] considers the sensitivity of the wavelet technique in the detection of cracks in beam structures. Two types of wavelets are compared: Haar and Gabor wavelets. The dimension of the crack projected along the longitudinal direction can be deduced from analysis. The authors of this paper, [18], use wavelet-based elements to model a cracked shaft in order to obtain precise frequencies. The normalized crack location and depth are detected by means of a genetic algorithm. The paper [19] describes a new method for automatically detecting cracks in pavement surface images. This technique is based on the continuous wavelet transform. Post-processing yields a binary image which indicates the presence of cracks if there any in a pavement surface image.

2.4 Radon Transform-Based Methods of Crack Detection

In this paper, [20], computed tomography is used to investigate cracks in composite materials such as short-fibre reinforced carbon fibre reinforced plastics (CFRP). The objective of this paper is to detect all relevant cracks inside an object in a stable way, reducing the effect of noise.

2.5 Methods of Crack Detection Utilizing Wavelet and Radon Transforms

These papers either combine the wavelet and Radon transforms or else use both separately for the detection of cracks in images. The paper [21] describes an empirical method for the identification of linear structural damage in asphalt pavements which are categorized by longitudinal, transverse, diagonal, block (radon) and alligator (fatigue) fractures or cracks. A two-dimensional discrete wavelet transform (DWT) is used for multidirectional and multiscale crack detection. The circular Radon transform (CRT) is used for angular-geometric orientation analysis for the identification and classification of crack types. The paper [22] describes an automatic diagnosis system for the detection and classification pavement crack problem based on the Wavelet-Radon transform (WR) and the Dynamic Neural Network (DNN) threshold selection. This method combines feature extraction using WR and classification using the neural network. The authors of this paper, [23], use a Radon neural network, based on a wavelet transform expert system to improve the capability of the scale invariant feature extraction algorithm. The wavelet modulus is calculated, and the Radon transform is then applied to the wavelet modulus. The features and parameters of the peaks are used for training and testing the neural network. The paper [24] compared multi-resolution texture analysis techniques using wavelet, ridgelet, and curvelet-based texture descriptors. Recall that the Radon transform is defined as

$$R_f(\theta, t) = \int_{R^2} f(x, y)\delta(t - x\cos\theta - y\sin\theta)dxdy$$

The continuous ridgelet transform is defined as the application of the one-dimensional wavelet transform to the projections of the Radon transform and can be expressed as

$$CRT_f(a,b,\theta) = \int_{R^2} \psi_{a,b}(t) R_f(\theta,t) dt$$

The curvelet transform is a higher dimensional generalization of the wavelet transform designed to represent images at different scales and different angles. The technique in this paper consisted of image collection, segmentation of the regions of interest, extraction of the most discriminative texture features, and creation of a classifier that automatically identifies pavement distress and storage. Curvelet-based methods perform better than all other multi-resolution techniques for pothole damage. Ridgelet-based techniques perform better than all of the multi-resolution methods for crack problems.

2.6 Other Image Processing Methods for Analyzing Cracks

The authors, [25], describe a vision-based visual inspection technique involving the processing and analysis of a big set of collected images. Utilizing images from different angles and prior knowledge of typical appearance and properties of this class of faults, this technique can detect cracks near bolts. The article by [26] describes a beamlet transform-based approach to automatically detect and classify pavement cracks in images. To extract linear features such as surface cracks from pavement images, the image is partitioned into small windows. Then, a Beamlet transform-based algorithm is applied to the image. Crack segments are linked together and classified into four categories: vertical, horizontal, transversal, and block. The authors of this paper, [27], describe a detection technique for road cracks in several directions and each layer for pavement image in a contourlet domain. The contourlet transform is a double filter bank structure composed of two pieces, the Laplacian pyramid (LP) used to find point discontinuities, and a directional filter bank (DFB) then used to link point discontinuities into linear structures which form contour segments. In this paper [28], a new technique is described to detect and segment a crack on a pavement surface image from its background. A weighted neighborhood pixels method is discussed, which is based on the intensities of all pixels in three surrounding loops. The authors, [29], present a novel crack detection method based on local binary pattern (LBP), support vector machine (SVM), and Bayesian decision theory. The technique described combines the information obtained from different video frames to enhance the robustness and reliability of detection. The paper [30] describes the use of computer-vision techniques in detection and analysis of cracks on a bridge deck. After feature extraction using the training set images, statistical inference algorithms are employed to identify the cracks. The paper [31] describes a new measure which considers simultaneously brightness and connectivity, in the segmentation step, for crack detection on road pavement images. Features which are computed along every free-form path allow detection of cracks with any form and orientation. The authors, [32], of this paper propose a new method of crack image processing for concrete bridge bottom crack inspections. They construct a machine vision system based on this method, which can detect cracks in real time. In this paper [33], a novel local binary group (LBP)-based operator for pavement crack detection is proposed. In this method, local neighbors are classified into a smooth area and a rough area. Segmentation is only performed in the rough area to capture local feature information.

The authors, [34], reviewed fifty papers related to crack detection. They classified the image processing methods in those papers into four categories: integrated algorithm, morphological approach, percolation-based method, and practical technique. They considered features like length, width, depth, surface of crack, and direction of propagation.

2.7 Percolation-Based Image Processing

The papers in this section utilize percolation-based methods of image processing for crack detection. Percolation theory considers the activity of connected sets in a random graph. The authors of this paper, [35], present an efficient and high-speed crack detection method that employs percolation-based image processing. In this paper, the authors, [36], introduce an efficient and high-speed method for crack detection employing percolation-based image processing. To lower the computation time, they use ideas from the sequential similarity detection algorithm (SSDA). From SSDA, the percolation process is terminated by calculating circularity half-way through processing. The authors, [37], describe a highly accurate and efficient method for crack detection using percolation-based image processing. Their technique uses a percolation model for crack detection in order to analyze the crack features. The paper [38] describes a new technique for image processing based on a percolation model. First, a cluster is formed through the percolation process. The, feature extraction is performed based on the cluster. This technique was verified by tests on crack detection.

2.8 The Use of the Radon Transform for Orientation Detection

The paper [39] by Jafari-Khouzani and Soltanian-Zadeh presents a method for addressing the problem of rotation invariant texture classification. For directional textures, the wavelet features must be computed for a particular direction. In this paper, the Radon transform is first utilized to determine the primary texture direction. The texture is then rotated in such a way that its primary direction is at 0 degrees.

The paper [40] by Aggarwal and Karl notes that the determination of the location and orientation of straight lines in images is of primary interest in fields like computer vision and image processing. The Hough transform (a special case of the Radon transform) has been employed to address this problem for binary images. The authors of this paper consider the line detection problem in images as an inverse problem. They make use of the inverse Radon operator which relates parameters involving the line location and orientation to the noise-degraded image. This places the problem within a regularization context and improves the performance of Hough-based line detection through the use of prior information with respect to regularization.

The paper [41] by Rajput, Som, and Kar uses the Radon Transform to determine the orientation of license plates. There, each image is of a license plate oriented at a particular angle. They don't, however, deal with multiple license plates at different orientations in a single image. They briefly discuss other methods which attempt to address multiple orientations and their limitations.

The authors of [42] utilize the Radon transform to determine the orientation of fingerprints as part of a fingerprint recognition system. They note that the orientation of the ridge and the valley in a fingerprint is very important in the identification of fingerprints. The Radon transform is employed for obtaining this orientation information.

Chapter 3

Filtering the Reconstructed Image

Grayscale Images

Backprojection involves computing the inverse Radon transform and integrating (or summing in the computer approximation to integration) the projections in the sinogram from 0 to 179 degrees. What is implemented on the computer for backprojection [43] is something of the form:

$$f(x,y) = \frac{1}{N_{ap}} \sum_{i=0}^{N_{ap}-1} \Delta\theta_i p_{\theta_i}(x\cos\theta_i + y\sin\theta_i),$$

where N_{ap} is the number of angular projections computed in the course of calculating the sinogram, and p_{θ_i} is a projection or Radon transform.

Backprojection from several angles (directions) has the effect of applying a low-pass filter (with frequency response $\frac{1}{|\omega|}$) to an image. This will boost the lower frequencies and attenuate the higher frequencies. This can be compensated by multiplying the filter used during backprojection by $|\omega|$.

The reason for the need to multiply the filter by $|\omega|$ is due to the Central Slice Theorem which relates the Fourier and Radon transforms. By the Fourier Slice Theorem, the one-dimensional Fourier Transform of a projection (the Radon transform of a function) is equal to the two-dimensional Fourier transform of the original function. Recall that the Radon transform of a function f(x,y) is given by

$$Rf = p_\theta(\rho) = \int_{-\infty}^{\infty} f(x,y)\delta(x\cos\theta + y\sin\theta - \rho)dxdy$$

Let

$$P(\omega) = \mathcal{F}[p(\rho)]$$

where $\mathcal{F}$ denotes the Fourier transform, p denotes the projection, and P is its Fourier transform. Let

$$f(x,y) = \mathcal{F}_2^{-1}F(v_x, v_y)$$

where $\mathcal{F}_2$ denotes the two-dimensional inverse Fourier transform of F. Now,

$$f(x,y) = \mathcal{F}_2^{-1}F(v_x,v_y) = \int_{-\infty}^{\infty} dv_x \int_{-\infty}^{\infty} F(v_x,v_y)\exp 2\pi i(v_x x + v_y y)dv_y$$

Now, in polar coordinates,

$$v_x = \omega\cos\theta$$

$$v_y = \omega\sin\theta$$

and thus

$$dv_x dv_y = \omega d\omega d\theta$$

$$\begin{aligned} f(x,y) &= \int_0^{2\pi} d\theta \int_0^{\infty} \omega F(\omega\cos\theta, \omega\sin\theta)\exp\left(2\pi i\omega(x\cos\theta + y\sin\theta)\right)d\omega \\ &= \int_0^{\pi} d\theta \int_{-\infty}^{\infty} |\omega| P(\omega)\exp\left(2\pi i\omega\rho\right)d\omega \\ &= \int_0^{\pi} p_\theta(\rho)d\theta \end{aligned}$$

where

$$F(\omega\cos\theta, \omega\sin\theta) = P(\omega)$$

by the Fourier Slice Theorem. Therefore,

$$\int_{-\infty}^{\infty} |\omega|P(\omega)\exp\left(2\pi i\omega\rho\right) = \mathcal{F}^{-1}(|\omega|P(\omega)) = p_\theta(\rho)$$

In filtered backprojection, the above equation is modified to show the convolution of a projection with a high-pass filter as in the following:

$$f(x,y) = \frac{1}{N_p}\sum_{i=0}^{N_p-1} \Delta\theta_i p_{\theta_i} hp(x\cos\theta_i + y\sin\theta_i),$$

where hp is the high-pass filter that is convolved with a projection p_{θ_i}.

In filtered backprojection each 1-D projection is convolved with a 1-D high-pass filter (either Ram-Lak or Shepp-Logan) to eliminate the blurring inherent in backprojection. A back-projected image is also known as a laminogram.

The Ram-Lak filter is due to Ramachandran and Lakshiminarayanan. It has a frequency response given by

$$H(\omega) = \begin{cases} 1 & , \quad \omega < \Omega \\ 0 & , \quad \text{otherwise} \end{cases}$$

The overall RL filter is a ramp filter and has a frequency response given by

$$H_{RL}(\omega) = H(\omega)|\omega| = \begin{cases} |\omega| & , \quad \omega < \Omega \\ 0 & , \quad \text{otherwise} \end{cases}$$

Taking the inverse Fourier transform of this transfer function yields the following impulse response function in the spatial domain which is applied to the sinogram (Radon transform) of the image

$$h(x) = \frac{\Omega^2}{\pi}\left(sinc\left(\Omega x\right) - sinc^2\left(\frac{\Omega x}{2}\right)\right)$$

There is another filter called the Shepp-Logan filter. However, unlike the Ram-Lak filter, the Shepp-Logan filter employs a function with a smoother frequency response. The frequency response of the Shepp-Logan filter is given by

$$H(\omega) = \begin{cases} \frac{\pi}{2\Omega}\left|sinc\left(\frac{\pi\omega}{2\Omega}\right)\right| & , \quad \omega < \Omega \\ 0 & , \quad \text{otherwise} \end{cases}$$

The overall filter has a frequency response given by

$$H_{SL}(\omega) = H(\omega)|\omega| = \begin{cases} \left|\sin\left(\frac{\pi\omega}{2\Omega}\right)\right| & , \quad \omega < \Omega \\ 0 & , \quad \text{otherwise} \end{cases}$$

Taking the inverse Fourier transform of this filter yields the following impulse response function which is applied to the sinogram (Radon transform) of the image.

$$h(x) = \frac{1}{\pi}\frac{\frac{\pi}{2\Omega} - x\sin\left(\Omega x\right)}{\left(\frac{\pi}{2\Omega}\right)^2 - x^2}$$

In the process of reconstructing an image by filtered backprojection from a sinogram image, artifacts were produced. In particular, when the image-building.tif source image was used, after a Radon transform and subsequent reconstruction from the sinogram, some ghost artifacts could be seen in the reconstructed image. Three different edge-preserving smoothing filters were studied for this purpose [44].

The first such filter considered is the Malik-Perona filter. Consider the diffusion equation

$$\frac{\partial I}{\partial t} = div(c \cdot \nabla I)$$

where c is the conductivity. If c is a real constant, then the diffusion process (which reduces to the heat equation) is isotropic, conveying heat evenly in every direction. A characteristic of isotropic diffusion is that it produces the same result as a Gaussian filter whose width increases with time. Since Gaussian filters reduce the level of noise in an image while also unfortunately blurring the edges, isotropic diffusion will do the same thing. In order to obtain edge-preserving smoothing, the conductivity, c, is taken to be a function of the magnitude of the gradient of I. That is

$$c(x, y, t) = f(||\nabla I(x, y, t)||)$$

To smooth an image while not degrading the edges, f (which maps the real line into the unit interval) must have high values in low gradient areas of the image (allowing smoothing there) while taking on low values in regions of the image characterized by high values of the gradient. One such example of a conductivity function is the function

$$f(r) = \exp(-r/\kappa^2)$$

When the Perona-Malik filter was coded in MATLAB for the image-building image, horizontal streaking was produced in the image.

Another filter that was considered was the Bilateral filter. Typically, a low-pass filter for an image works by convolving an image with a smoothing kernel. The kernel coefficients weight the associated image pixels and are a function only of the distance to the center of the filter. In non-uniform filters, the coefficients near the center have a larger value than those near the edges of the filter mask.

In this way, the kernel incorporates the closeness of the associated image pixels. A filter whose weights depend only on the spatial distance is a domain filter. The usual convolution mask filters are all thus domain filters.

To construct a low-pass filter so that it will do the least amount of harm to the edges in an image, either particular (edge) pixels can be removed from filtering or a lower weight can be assigned to such pixels if they differ by more than a specified amount from the center pixel. If this is represented by a filter, the kernel weights are a function of the difference in pixel values or the range. This type of filter is called a range filter. A range filter can be expressed in the following way

$$I_{out}(u,v) = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} I(i,j) \cdot H_r(I(i,j) - I(u,v))$$

where H_r is the range filter kernel. While a domain filter will either act as a high-pass or low-pass filter on an image, a range filter will have no spatial effect on the image.

A Bilateral filter combines both domain and range filters into an edge-preserving low-pass filter. If a given image pixel is near in value and in distance to the center pixel, then the Bilateral filter acts as a low-pass filter. For example, in a flat region, where most surrounding pixels have values similar to the center pixel, the Bilateral filter acts as a smoothing filter, controlled only by the domain kernel H_d. If however, the image pixel has a much different value than the center pixel, no filtering is done since this is likely an edge pixel. For example, when situated near a step edge or an intensity ridge, only those pixels are included in the smoothing process that are similar to the center pixel, thus avoiding blurring the edges. Thus, the edges are not degraded with this filter. When a bilateral filter with a 2D Gaussian kernel for the domain filter and a 1D Gaussian kernel for the range filter was used on the image after FBP, the resulting image did not show much improvement from the FBP image.

The last edge-preserving, smoothing filter that was employed was the Kuwahara filter. A Kuwahara filter uses a set of filter masks. It then computes the mean and variance for the image pixels corresponding to these masks and chooses the mean value of the mask with the lowest variance to replace the pixel associated with the center of the mask. The filter area R is broken up into k

$$\begin{bmatrix} \bullet & \bullet & \cdot \\ \bullet & \bullet & \cdot \\ \cdot & \cdot & \cdot \end{bmatrix} \begin{bmatrix} \cdot & \bullet & \bullet \\ \cdot & \bullet & \bullet \\ \cdot & \cdot & \cdot \end{bmatrix} \begin{bmatrix} \cdot & \cdot & \cdot \\ \cdot & \bullet & \bullet \\ \cdot & \bullet & \bullet \end{bmatrix} \begin{bmatrix} \cdot & \cdot & \cdot \\ \bullet & \bullet & \cdot \\ \bullet & \bullet & \cdot \end{bmatrix}$$

Figure 3.1: The Four Subregions for the Kuwahara Filter

partially overlapping subregions R$_1$,...,R$_k$. At each image pixel (m,n), the mean and variance of each subregion are calculated as

$$\mu_k(m,n) = \frac{1}{|R_k|} \sum_{(i,j) \in R_k} I(m+i, n+j) = \frac{1}{n_k} Sum_{1,k}(m,n)$$

$$\begin{aligned} \sigma_k^2(m,n) &= \frac{1}{|R_k|} \cdot \sum_{(i,j) \in R_k} (I(m+i,n+j) - \mu_k(m,n))^2 \\ &= \frac{1}{R_k} \cdot \left[Sum_{2,k}(m,n) - \frac{Sum_{1,k}^2(m,n)}{|R_k|} \right] \end{aligned}$$

for k = 1,...,K, with

$$Sum_{1,k}(m,n) = \sum_{(i,j) \in R_k} I(m+i, n+j),$$
$$Sum_{2,k}(m,n) = \sum_{(i,j) \in R_k} I^2(m+i, n+j)$$

The mean of the subregion centered at (m,n) with the smallest variance is selected as the new pixel value at (m,n). This means that

$$I_{out}(m,n) = \mu_k^*(m,n)$$

with

$$k^* = \arg \min_{k=1,\ldots,K} \sigma_k^2(m,n)$$

The original Kuwahara filter uses four 2x2 subregions in a 3x3 filter. The first subregion is the 2x2 block in the upper left portion of the 3x3 mask. The second subregion is the 2x2 block in the upper right portion of the 3x3 mask. The third subregion is the 2x2 block in the lower right portion of the 3x3 mask. The fourth subregion is the 2x2 block in the lower left portion of the 3x3 mask. The Tomita-Tsuji filter uses five 3x3 subregions in a 5x5 filter. Like the Kuwahara filter, four of the subregions cover the upper left, upper right, lower right, and lower left portions of the filter. The fifth subregion is a 3x3 subregion centered at the center of the 5x5 filter. When

$$\begin{bmatrix} \bullet & \bullet & \bullet & \cdot & \cdot \\ \bullet & \bullet & \bullet & \cdot & \cdot \\ \bullet & \bullet & \bullet & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \end{bmatrix} \begin{bmatrix} \cdot & \cdot & \bullet & \bullet & \bullet \\ \cdot & \cdot & \bullet & \bullet & \bullet \\ \cdot & \cdot & \bullet & \bullet & \bullet \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \end{bmatrix} \begin{bmatrix} \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \bullet & \bullet & \bullet \\ \cdot & \cdot & \bullet & \bullet & \bullet \\ \cdot & \cdot & \bullet & \bullet & \bullet \end{bmatrix} \begin{bmatrix} \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \bullet & \bullet & \bullet & \cdot & \cdot \\ \bullet & \bullet & \bullet & \cdot & \cdot \\ \bullet & \bullet & \bullet & \cdot & \cdot \end{bmatrix} \begin{bmatrix} \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \bullet & \bullet & \bullet & \cdot \\ \cdot & \bullet & \bullet & \bullet & \cdot \\ \cdot & \bullet & \bullet & \bullet & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \end{bmatrix}$$

Figure 3.2: The Five Subregions for the Tomita-Tsuji Filter

filtering the reconstructed version of image-building, the Tomita-Tsuji filter was used as well as 7x7 and 9x9 versions of that filter. Ghosting artifacts were visible in the reconstructed version of image-building. With a 9x9 version of the Tomita-Tsuji filter, these artifacts were removed or greatly reduced. There were still some visible artifacts from the computed projections, but the filtered image was clearer than the reconstructed image.

Color Images

Filtering reconstructed color images involves applying a filter to the three color channels separately. For the color image chosen, balloons-desaturate, the artifacts were not nearly as pronounced as those in the grayscale image that was used. As a result, when the color version of the 9x9 Tomita-Tsuji filter was used, it provided too much filtering, resulting in a black image. Next, a 5x5 Tomita-Tsuji filter with 3x3 subregions was employed. In this case, the resulting image was saturated, causing at least one of the original colors to change. Finally, a 3x3 Kuwahara filter with four 2x2 subregions was used. The resulting image looked very similar to the reconstructed image. There might have been a very slight improvement, which is understandable since the applied filter was a very light one.

$$\begin{bmatrix}
\bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \cdot & \cdot \\
\bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \cdot & \cdot \\
\bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \cdot & \cdot \\
\bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \cdot & \cdot \\
\bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \cdot & \cdot \\
\bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \cdot & \cdot \\
\bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \cdot & \cdot \\
\bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot
\end{bmatrix}
\begin{bmatrix}
\cdot & \cdot & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\
\cdot & \cdot & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\
\cdot & \cdot & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\
\cdot & \cdot & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\
\cdot & \cdot & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\
\cdot & \cdot & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\
\cdot & \cdot & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\
\cdot & \cdot & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot
\end{bmatrix}$$

$$\begin{bmatrix}
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\
\cdot & \cdot & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\
\cdot & \cdot & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\
\cdot & \cdot & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\
\cdot & \cdot & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\
\cdot & \cdot & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\
\cdot & \cdot & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet
\end{bmatrix}
\begin{bmatrix}
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \cdot & \cdot \\
\bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \cdot & \cdot \\
\bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \cdot & \cdot \\
\bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \cdot & \cdot \\
\bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \cdot & \cdot \\
\bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \cdot & \cdot \\
\bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \cdot & \cdot
\end{bmatrix}$$

$$\begin{bmatrix}
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \cdot \\
\cdot & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \cdot \\
\cdot & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \cdot \\
\cdot & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \cdot \\
\cdot & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \cdot \\
\cdot & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \cdot \\
\cdot & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \cdot \\
\cdot & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot
\end{bmatrix}$$

Figure 3.3: Five Subregions of the Tomita-Tsuji 9x9 Filter

Chapter 4

Preliminary Results

Grayscale Images

The image in figure 7.1 was input into a Radon transform. The sinogram was produced using MATLAB code which utilized 180 1D projections taken starting at 0 degrees to 179 degrees.

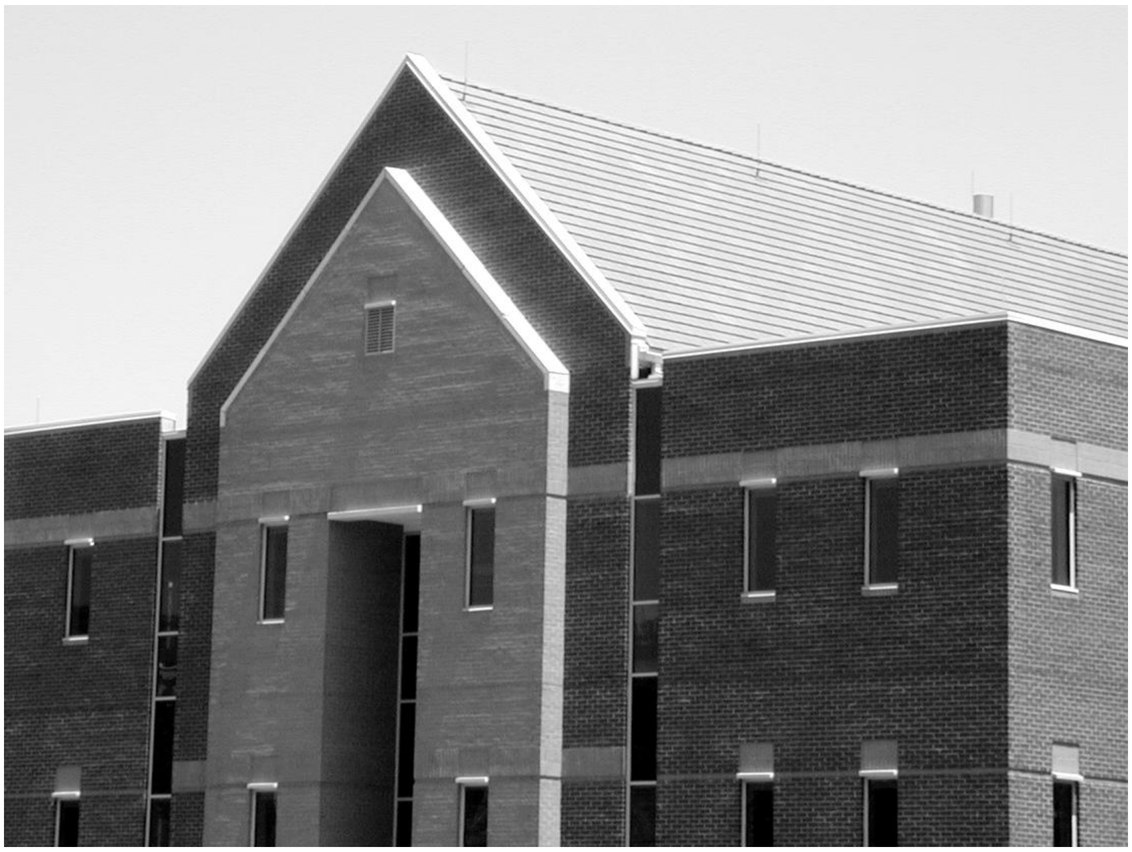

Figure 4.1: Input Image

After this image was processed by the Radon transform, the sinogram in figure 4.2 below was produced.

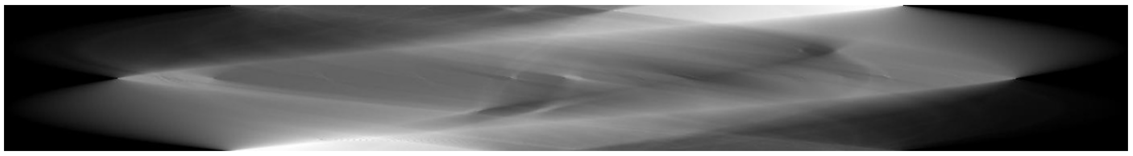

Figure 4.2: Sinogram

Then after applying filtered backprojection to the sinogram, and, in the process, applying the Ram-Lak filter to the sinogram, the reconstructed image in figure 4.3 was produced.

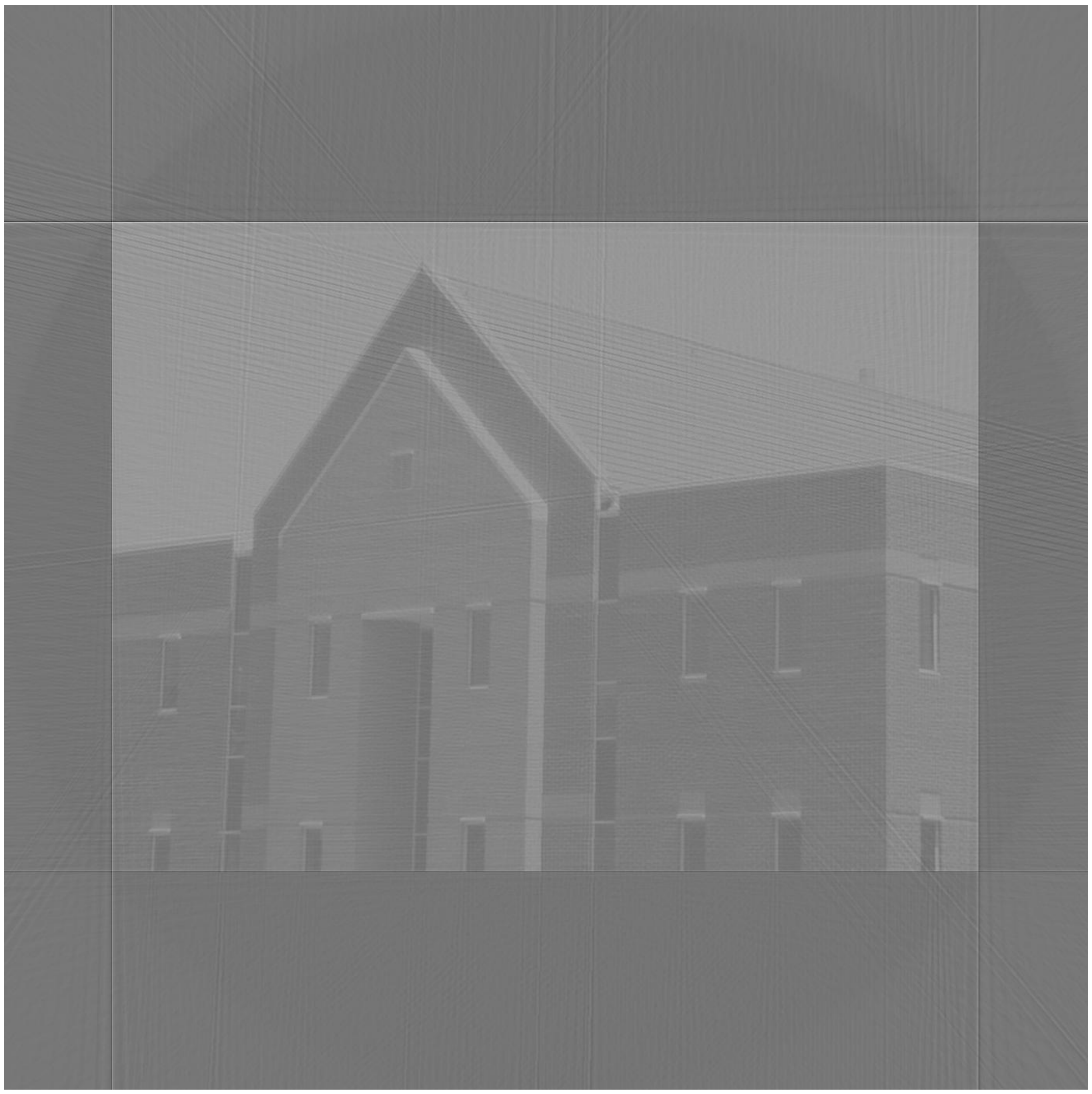

Figure 4.3: Reconstructed Image with Ram-Lak Filter

After applying filtered backprojection to the sinogram and, in the process, applying the Shepp-Logan filter to the sinogram, the reconstructed image in figure was produced.

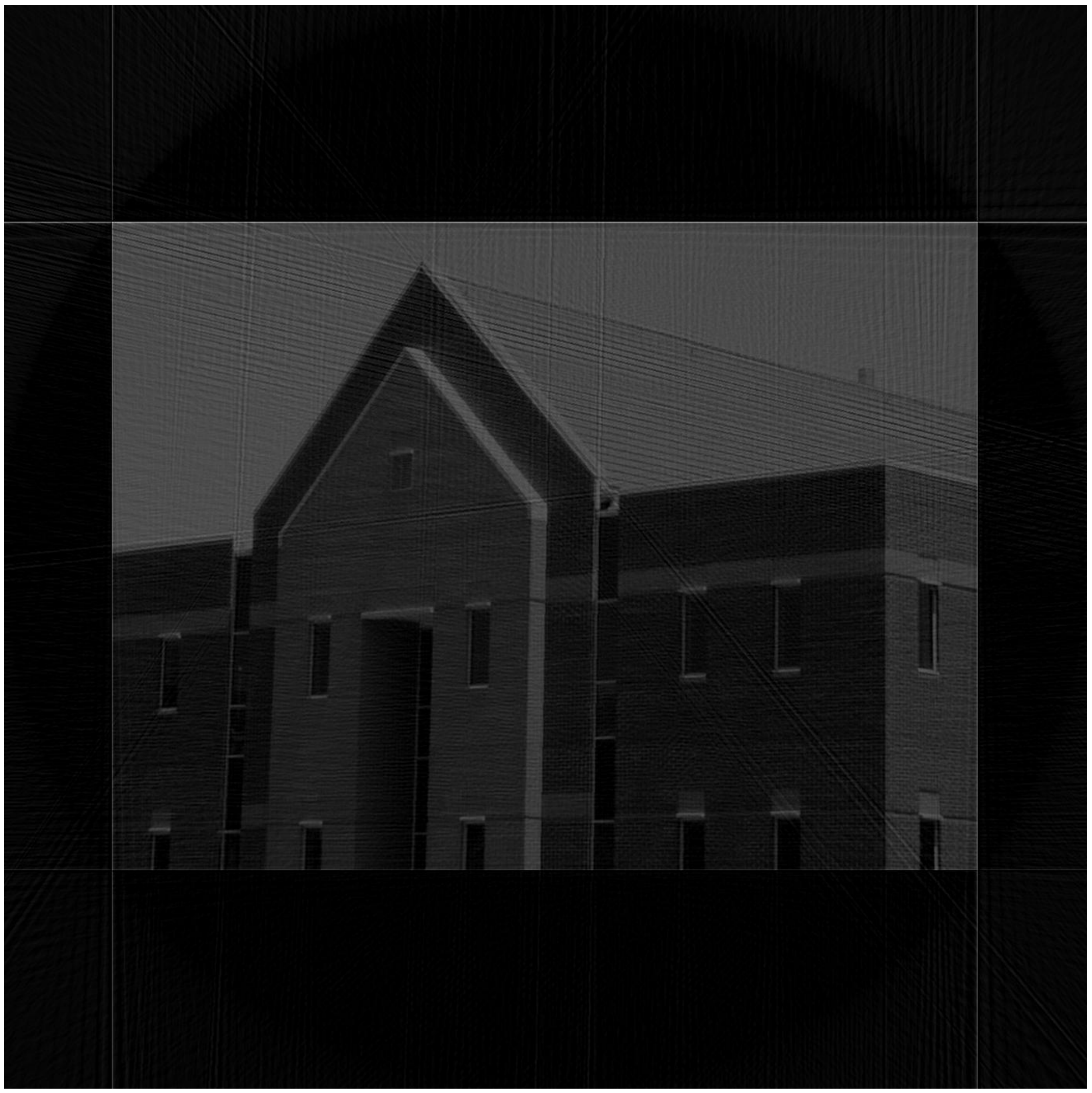

Figure 4.4: Reconstructed Image with Shepp-Logan Filter

After applying a 9x9 Tomita-Tsuji filter to the reconstructed image in figure 4.3, the image in figure 4.5 was produced.

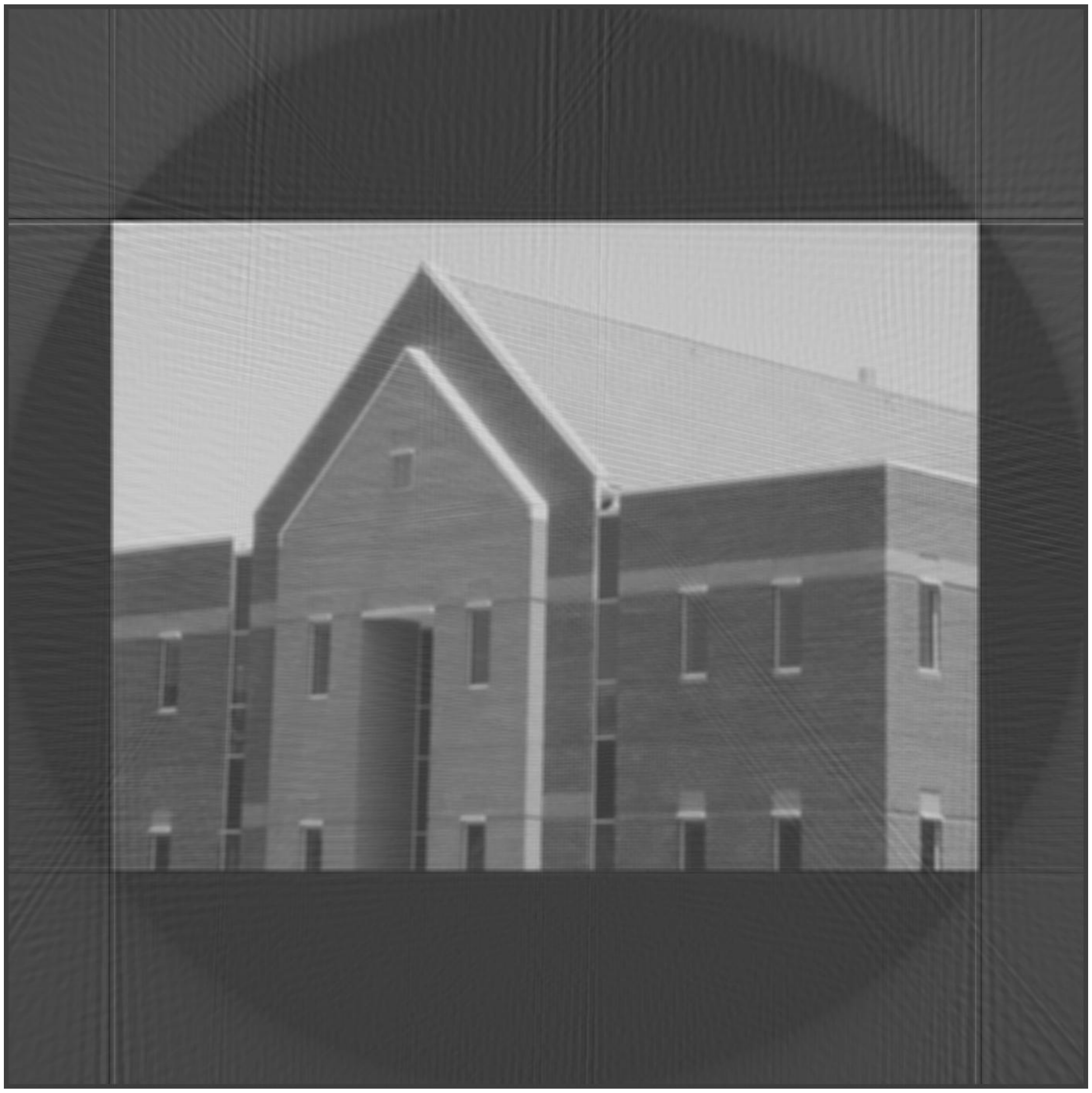

Figure 4.5: Ram-Lak Reconstructed Image after Processing with a 9x9 Tomita-Tsuji Filter

After applying a 9x9 Tomita-Tsuji filter to the reconstructed image in figure 4.4, the image in 4.6 figure was produced.

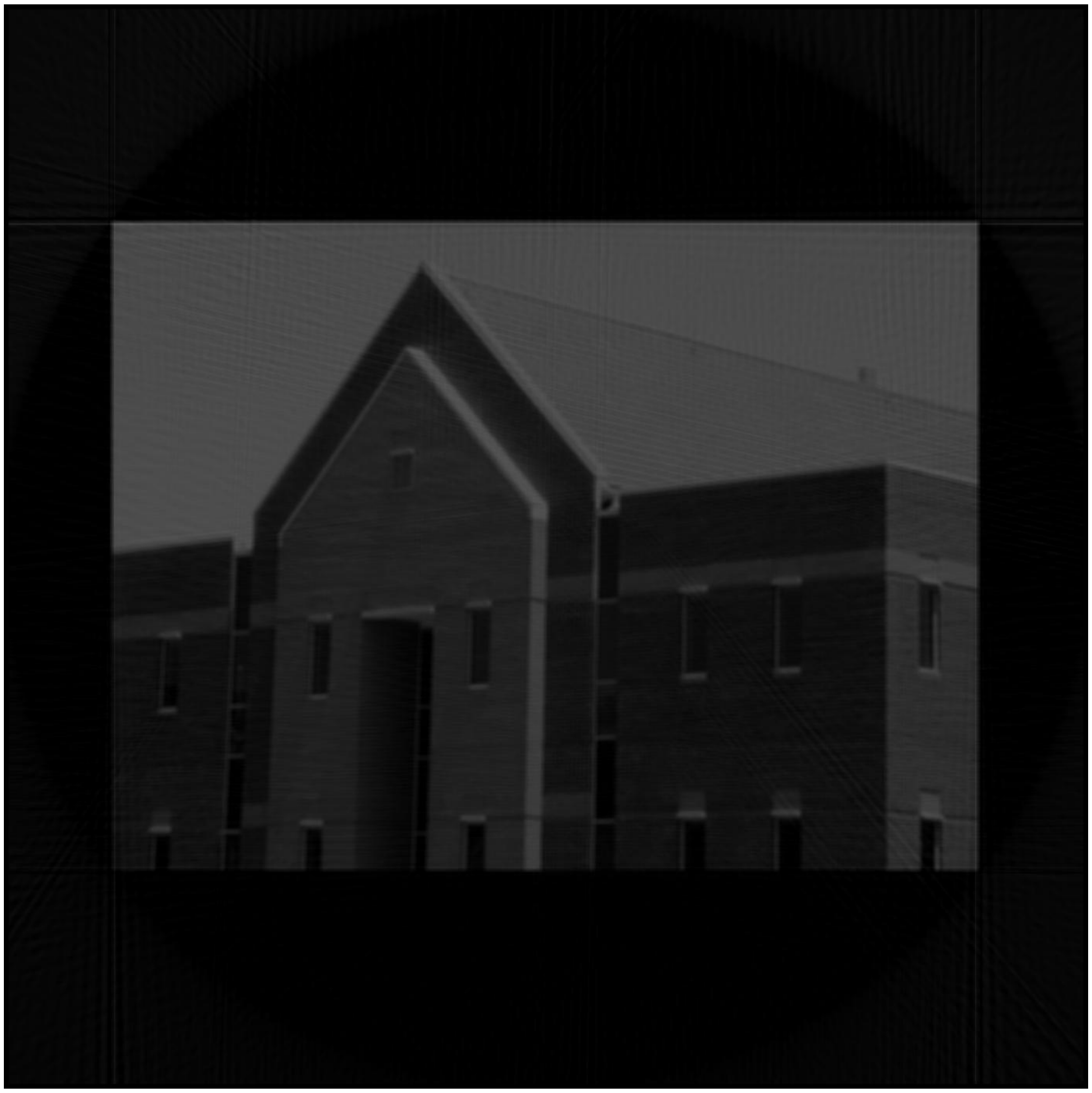

Figure 4.6: Shepp-Logan Reconstructed Image after Processing with a 9x9 Tomita-Tsuji Filter

A measure of the improvement achieved by filtering the reconstructed images was computed with the total variation (TV) norm of the images. The TV norm is computed by calculating the magnitude of the intensity gradient of the image using the horizontal and vertical Sobel edge operators. In particular, the magnitude of the gradient is approximated as

$$|\nabla I| = \sqrt{I_x^2 + I_y^2}$$

where I$_x$ is approximated by

$$I_x \approx I * S_x \text{ where * denotes convolution and I denotes the input image}$$

with

$$S_x = \frac{1}{8}\begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix}$$

where I$_y$ is approximated by

$$I_y \approx I * S_y \text{ where * denotes convolution and I is the input image}$$

with

$$S_y = \frac{1}{8}\begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{pmatrix}$$

Then, the total variation norm is given by

$$TV = \sum_{j=1}^{N}\sum_{i=1}^{M} \frac{|\nabla I(i,j)|}{M \times N}$$

where the input image I is an array of size MxN. Table 8.5 below shows the results of computing the TV norm for three different grayscale images.

Table 4.1: Total Variation Values for Grayscale Images

Image	TV Norm
reconstructed (with RL filter) image-building	4.1297
reconstructed (with RL filter) and filtered image-building	1.9437
reconstructed (with SL filter) image-building	3.2138
reconstructed (with SL filter) and filtered image-building	0.8783
reconstructed (with RL filter) image-moon	3.6955
reconstructed (with RL filter) and filtered image-moon	0.3128
reconstructed (with SL filter) image-moon	0.8164
reconstructed (with SL filter) and filtered image-moon	0.2586
reconstructed (with RL filter) image-irish	4.8112
reconstructed (with RL filter) and filtered image-irish	1.1226
reconstructed (with SL filter) image-irish	3.5019
reconstructed (with SL filter) and filtered image-irish	1.0014

From Table 8.5 above, we see that the total variation of the filtered, reconstructed images is significantly reduced from that of the reconstructed images without post-filtering.

Color Images

The image in figure 4.7 was processed by a Radon transform. That is, each of the three color channels were processed separately by a Radon transform.

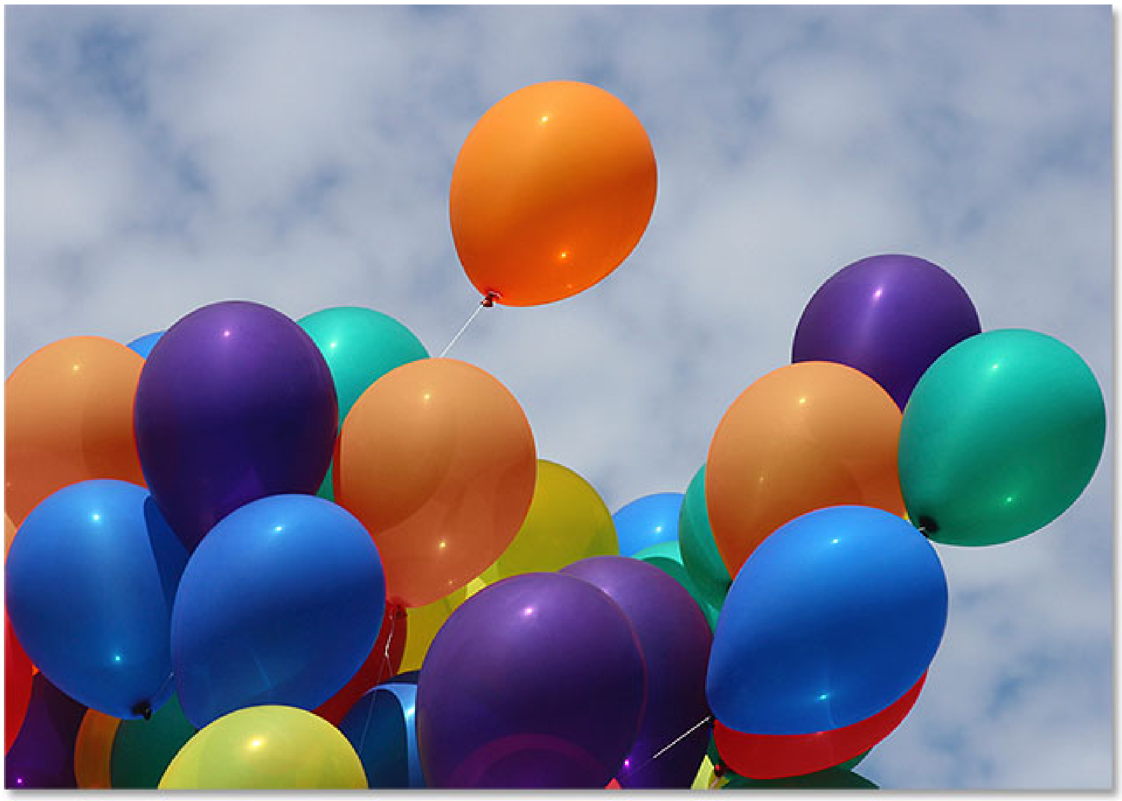

Figure 4.7: Color Image Input into a Radon Transform

Figure 4.8 shows a reconstructed version of the color image above after applying filtered backprojection (with the Ram-Lak filter) to the sinograms produced by the Radon transform applied to each of the three color channels of the original image.

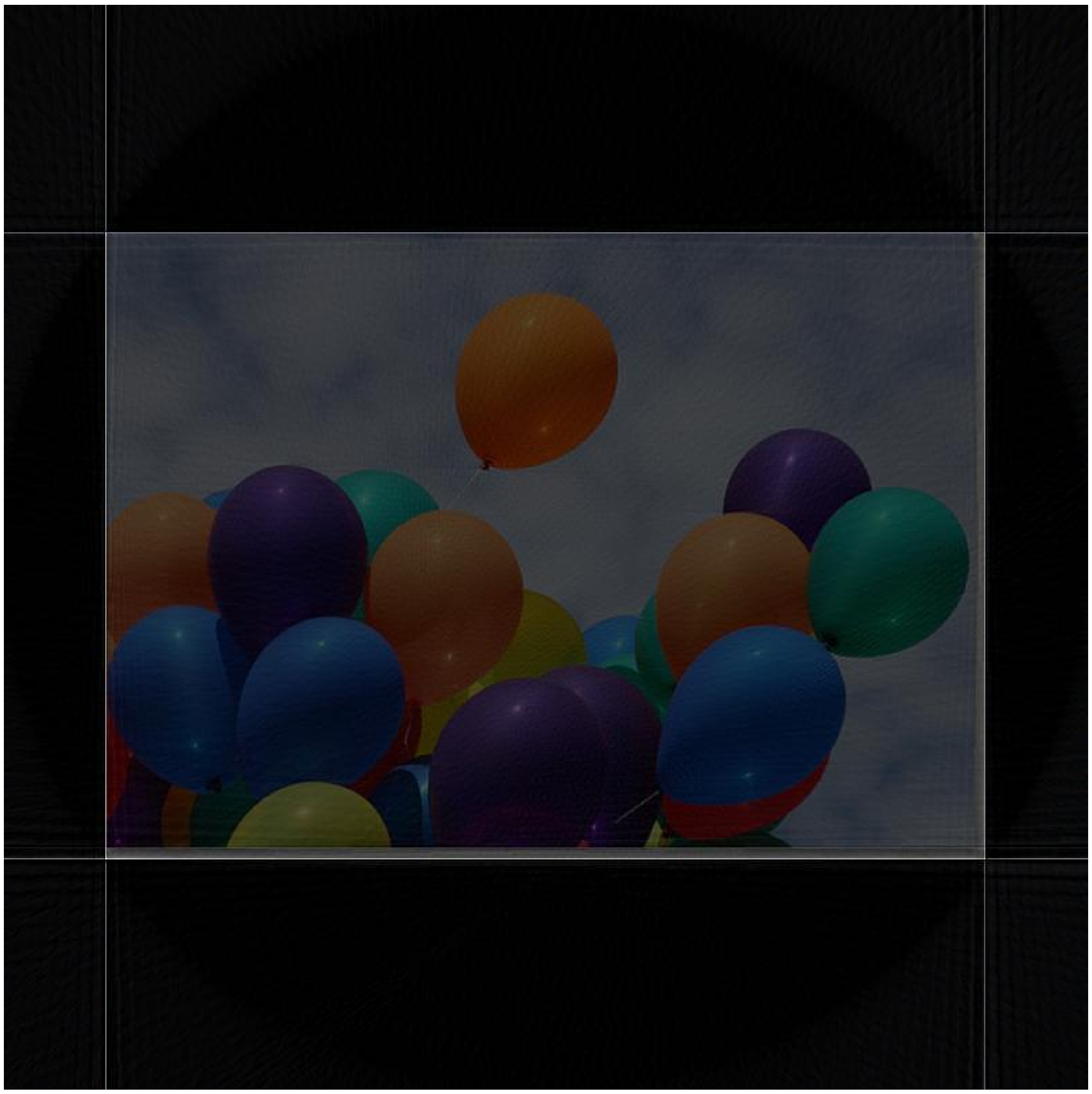

Figure 4.8: Reconstructed Color Image with RL Filter

Figure 4.9 shows a reconstructed version of the color image above after applying filtered backprojection (with the Shepp-Logan filter) to the sinograms produced by the Radon transform applied to each of the three color channels of the original image.

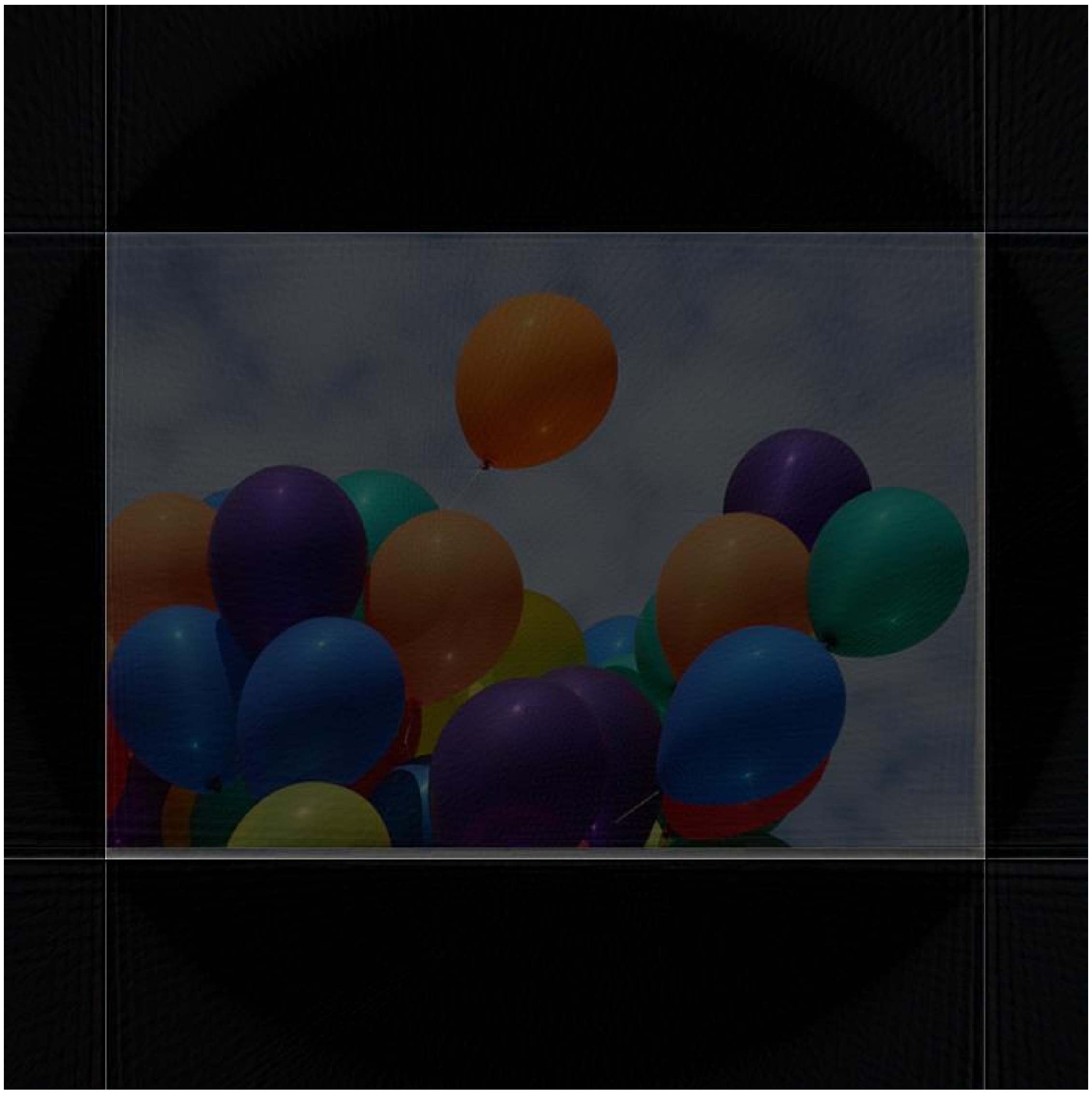

Figure 4.9: Reconstructed Color Image with SL Filter

The image in figure 4.10 shows the result of applying a 3x3 Kuwahara filter with four 2x2 subregions to the reconstructed color image in figure 4.8 above.

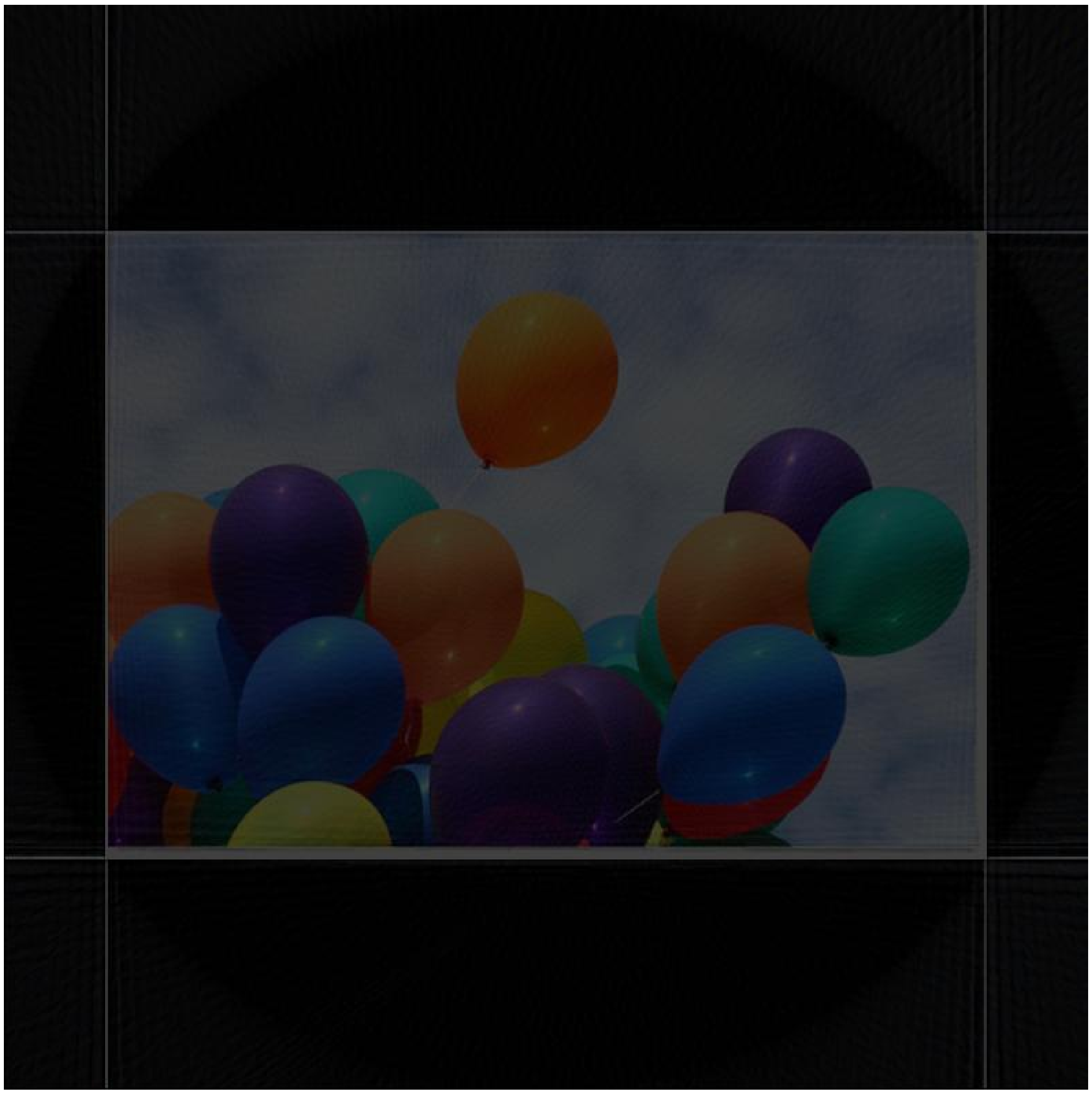

Figure 4.10: Kuwahara-Filtered RL Reconstructed Image

The image in figure 4.11 shows the result of applying a 3x3 Kuwahara filter with four 2x2 subregions to the reconstructed color image in figure 4.9 above.

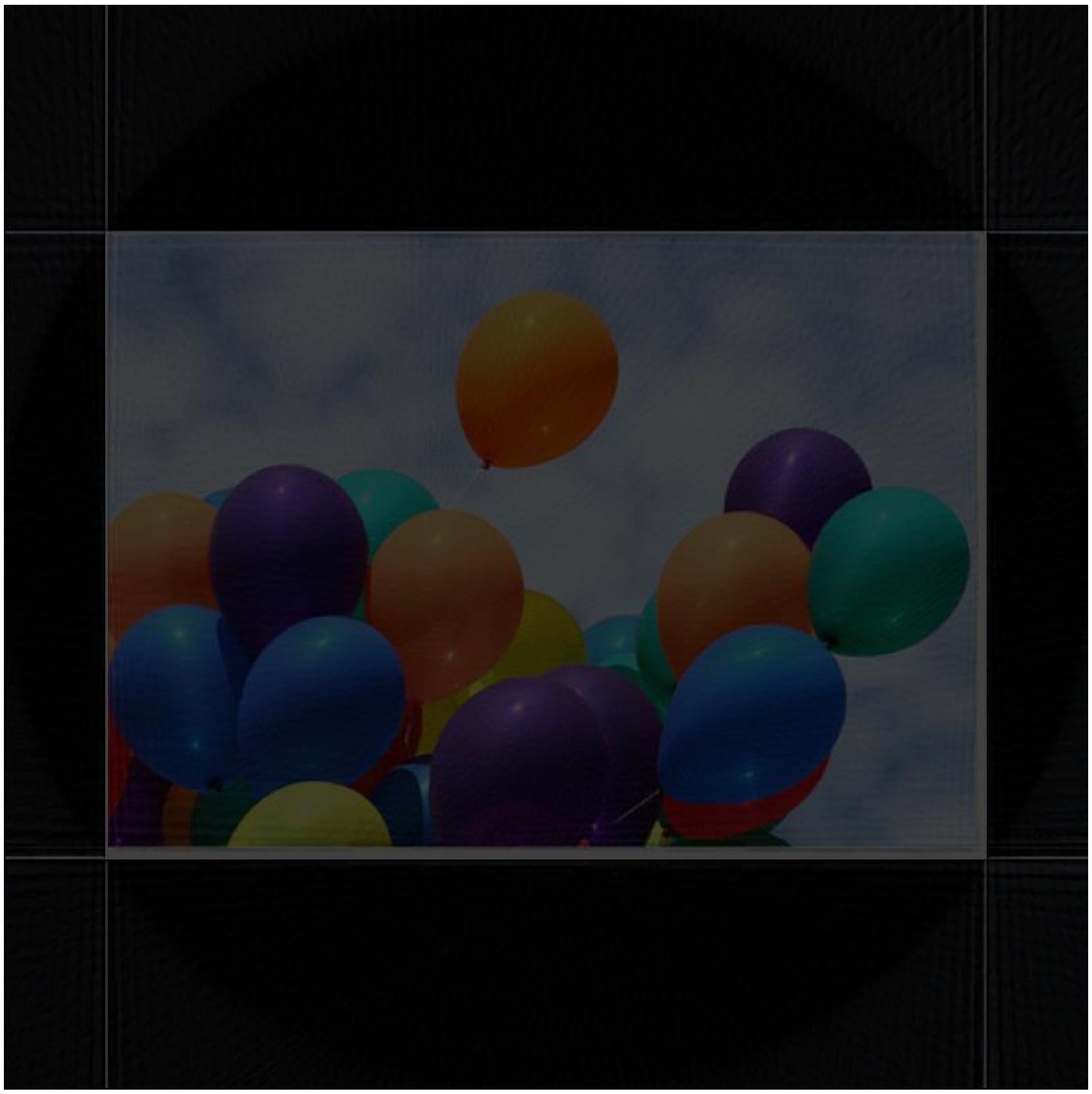

Figure 4.11: Kuwahara-Filtered SL Reconstructed Image

To measure improvement after application of the Kuwahara filter to color images, a color version of the TV norm was employed. In particular, the TV norm was computed for the red, green, and blue channels of the color images. The results are shown in Table 4.2 below.

Table 4.2: Total Variation Values for Color Images

Image	Red TV Norm	Blue TV Norm	Green TV Norm
color psychology recon (with RL filter)	1.9092	1.6846	1.5265
color psychology rec (with RL filter) filt	1.3505	1.3396	1.2234
color psychology rec (with SL filter)	1.7813	1.5696	1.4194
color psychology rec (with SL filter) filt	1.2927	1.2759	1.1601
balloons recon (with RL filter)	2.364	2.1695	2.4539
balloons rec (with RL filter) filt	1.7207	1.6134	1.7349
balloons rec (with SL filter)	2.1999	2.0149	2.2815
balloons rec (with SL filter) filt	1.6319	1.5356	1.6537

From Table 4.2 above, we see that the total variation of the filtered, reconstructed images is significantly reduced from that of the reconstructed images without post-filtering.

Chapter 5

Crack Detection with the Radon Transform

The Radon Transform can be used to detect cracks in images of cracked and non-cracked bridge decks, walls, and pavements. Since the Radon transform can readily detect line segments that cross the entire width or length of an image, the input image was carefully selected so that a crack that appeared had that property. Figure 5.1 shows the input image that was used. This input image (001-105.jpg) which is an image of cracked pavement was selected from a set of crack images available at

`https://digitalcommons.usu.edu/all\_datasets/48/` The dataset is called SDNET2018: A concrete crack image dataset for machine learning applications. The images in this dataset consist of cracked and non-cracked images of pavement, walls, and bridges. First, the image was smoothed using a 5x5 averaging, low-pass filter. Then, an edge map was generated from the image by computing the magnitude of the gradient across the image using the 3x3 Sobel approximations to the horizontal and vertical partial derivatives of the image. The binary edge map was produced by thresholding the magnitude of the gradient at 10% of the maximum gradient intensity value in the image. In particular, the magnitude of the gradient is approximated as

$$|\nabla I| = \sqrt{I_x^2 + I_y^2}$$

where I$_x$ is approximated by

$$I_x \approx S * S_x \text{ where * denotes convolution}$$

with S$_x$ and S$_y$ denoting the horizontal and vertical Sobel operators defined in the previous chapter.
where I$_y$ is approximated by

$$I_y \approx S * S_y \text{ where * denotes convolution}$$

where S = I*SF with I = the input image and

$$SF = \frac{1}{25}\begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

Figure 5.2 shows the edge map after applying the smoothing filter and the approximation to the magnitude of the gradient to the image. The resulting edge map is then reduced in size by removing the image border. This is done so that the Radon transform will not try to detect the borders as line segments in the image. The reduced image is then processed by the Radon transform. Since the crack is not a linear feature but rather a curve, one wouldn't expect the Radon transform of the crack to be a single point in the Radon domain but a cluster of points. The resulting sinogram 5.3 in figure shows a cluster of bright points. Since the bright points appear to be around an angle of 90 degrees and between approximately 20 to 50 pixels in distance from the origin, this cluster appears to correspond to the crack in the original image.

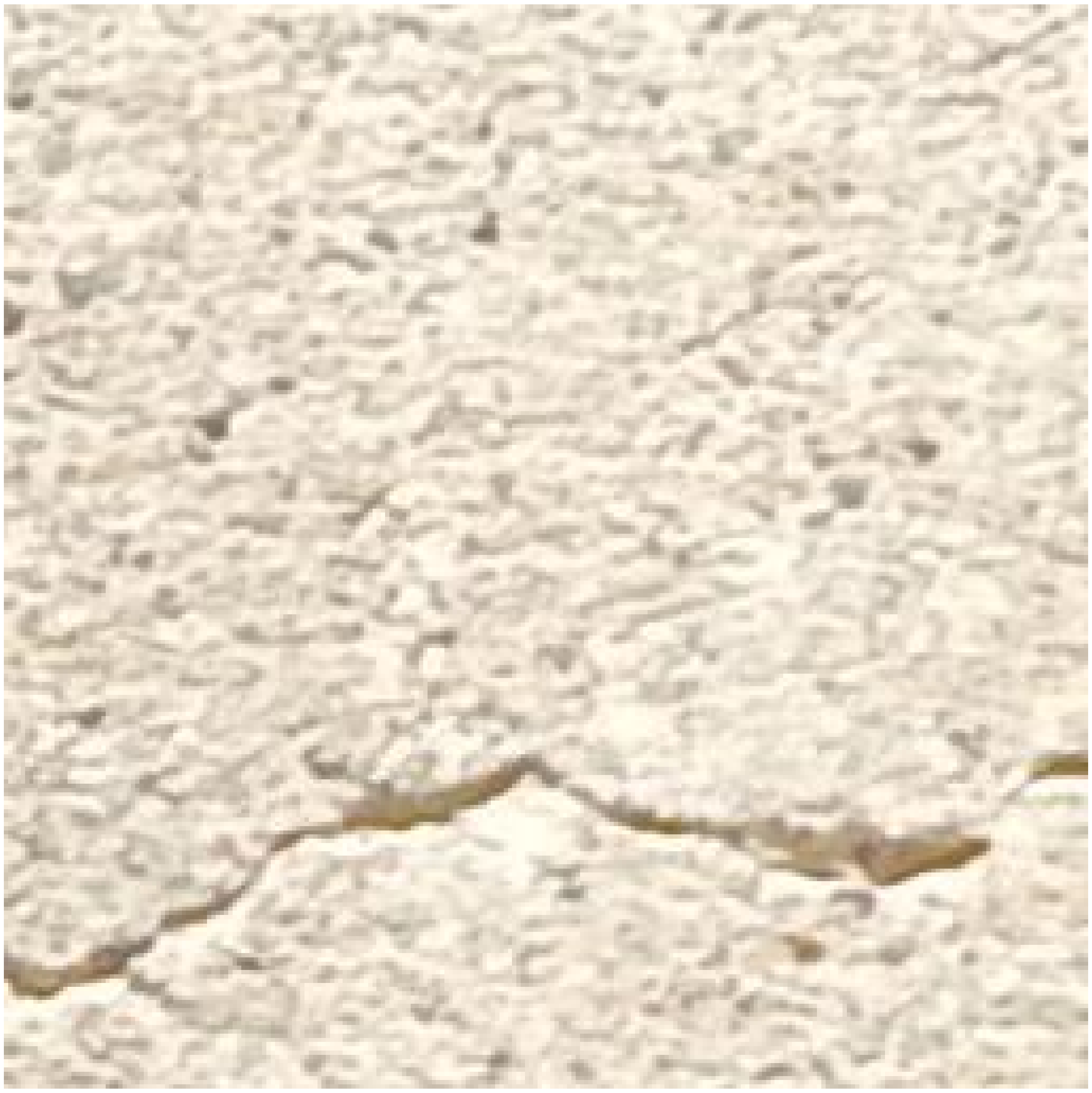

Figure 5.1: Input Image 001-105

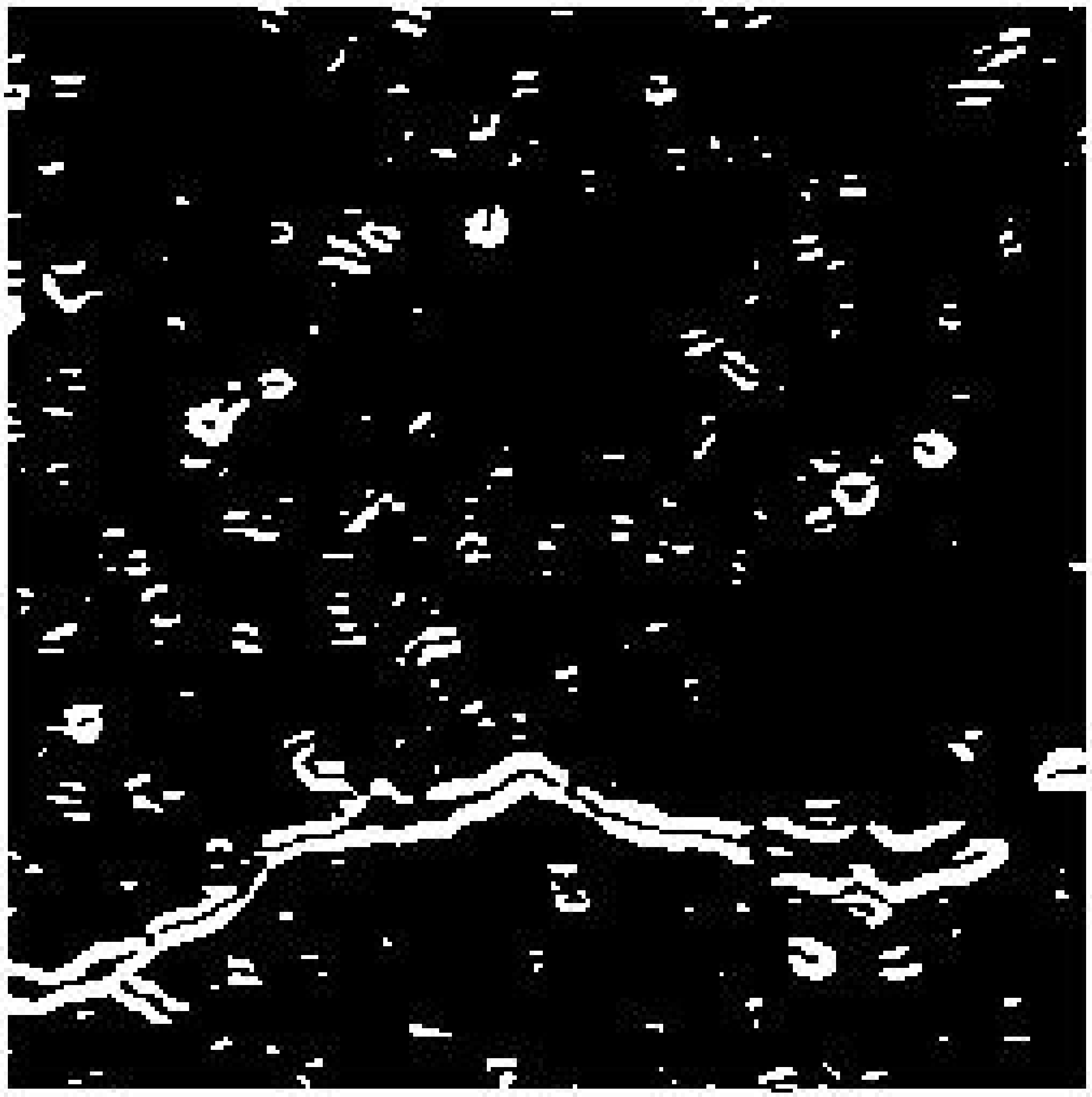

Figure 5.2: Edge Map

Figure 5.3: Sinogram With Single Bright Point

Chapter 6

Enhancement Using the Radon Transform

The Radon transform of a function f(x,y) on two-dimensional Euclidean space is defined by

$$P_\theta(\rho) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)\delta(\rho - x\cos\theta - y\sin\theta)dxdy$$

where $\delta(r)$ is the Dirac delta function. The presence of the term $\delta(\rho - x\cos\theta - y\sin\theta)$ in the definition of the Radon transform forces the integration of f(x,y) along the line

$$\rho - x\cos\theta - y\sin\theta = 0$$

Thus, the Radon transform becomes a line integral.

If f(x,y) is a two-dimensional image intensity function, computation of its Radon transform yields the projections across the image at varying orientations θ and distances (from the origin) ρ.

The Radon transform is a mapping from Cartesian (x,y) coordinates to coordinates (ρ,θ).

The Radon transform maps lines in image space to points in feature space. The mapping of image space to feature space is illustrated in Figure 6.1 from [45]. Bright (dark) lines in an image are mapped by the Radon transform to bright (dark) points in feature space.

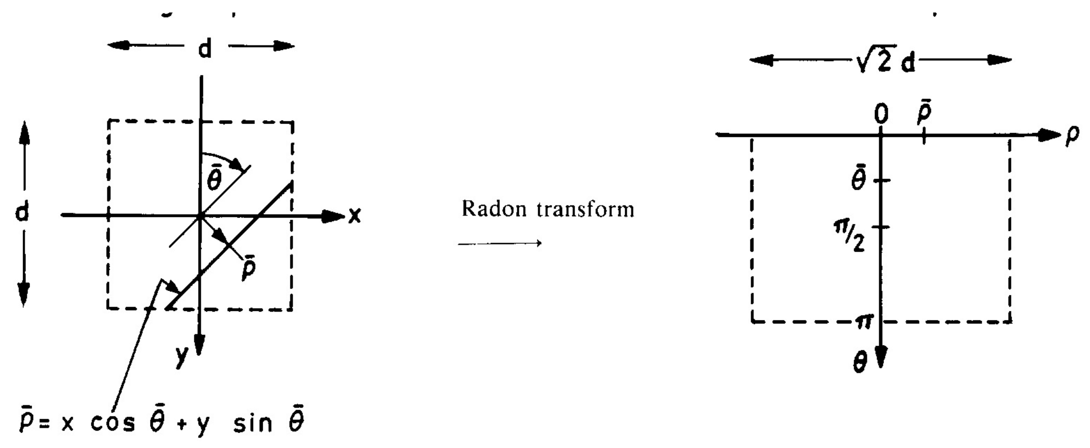

Figure 6.1: The Mapping of Image Space to Feature Space under the Radon Transform

For each angle θ and each distance ρ, the intensity of the object through which a ray perpendicular to the ρ axis passes is summed up at $P_\theta(\rho)$. A set of many such projections under different angles is a sinogram.

Consider the equation of a summation line as y=ax+b, shown in Figure 6.2 , where b is the y-intercept.

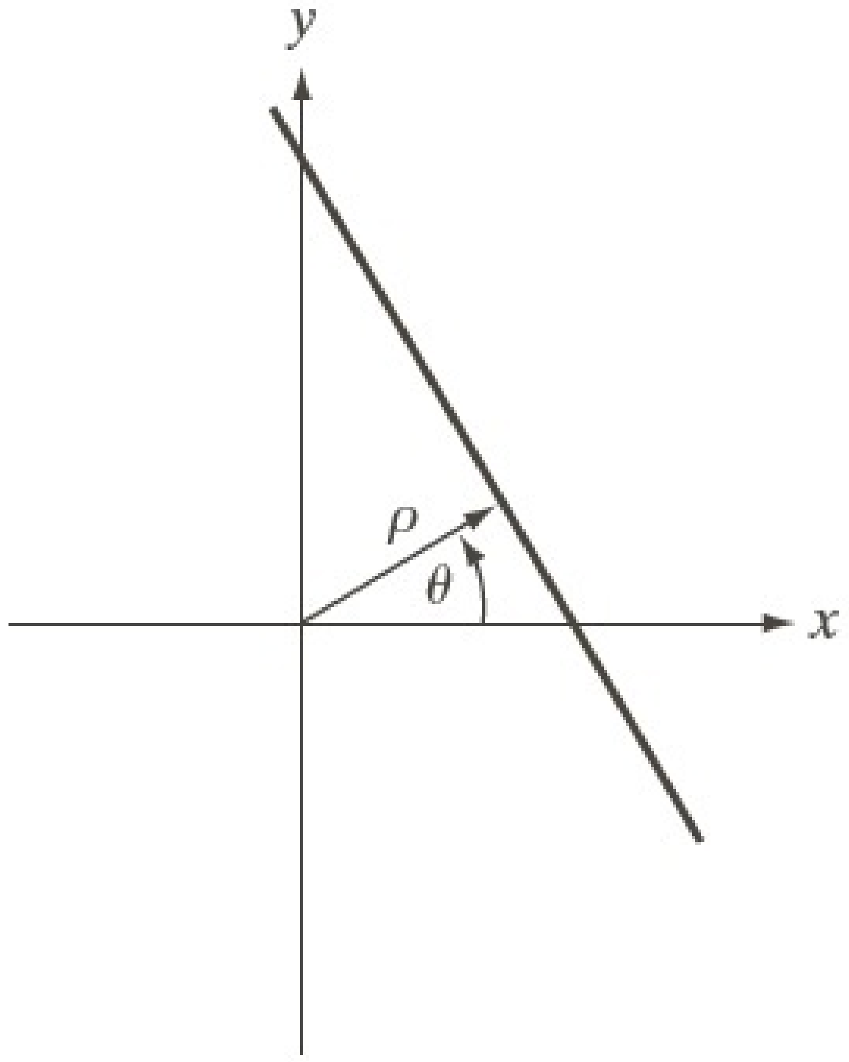

Figure 6.2: Line in Image Space

Now,

$$a = -\frac{\cos\theta}{\sin\theta}$$

Also,

$$b\sin\theta = \rho \Rightarrow b = \frac{\rho}{\sin\theta}$$

Therefore,

$$y = \left(\frac{-\cos\theta}{\sin\theta}\right)x + \frac{\rho}{\sin\theta}$$

Thus,

$$\rho = x\cos\theta + y\sin\theta$$

Let R denote the Radon transform operator. If f and g are functions, and if f(x,y)=g(x,y) for all (x,y) $\in$ R^2, then Rf = Rg. Consequently, the Radon transform is well-defined.

In 1981, S.R. Deans in his paper, "Hough Transform from the Radon Transform," [1] listed four properties of the Radon transform that he said Duda and Hart (in their paper entitled,"Use of the Hough transform to detect lines and curves in pictures") listed for the Hough transform. Deans concluded that the Hough transform is a special case of the Radon transform.

Since both transforms utilize integration of a function on a collection of lines, they work well on noisy images. The reason for this is that if we view a noise signal as having zero mean, then its variation about the zero mean is cancelled out by the integration process. Also, as a consequence, the signal-to-noise ratio of a noisy image containing a linear feature can be greater than that of the image itself. The signal-to-noise ratio increases because the noise level is lowered by integration.

The Radon transform is more computationally efficient than the Hough transform, and this is a consequence of the Fourier Slice Theorem.

This way of computing the Radon transform involves calculating the Fourier transform. The following are the three steps in computing the Radon transform via the Fourier transform.
(1) Compute the two-dimensional Fourier transform F(u,v) of the image intensity function f(x,y).
(2) Interpolate the two-dimensional Fourier transform to obtain a set of functions $S_\theta(\omega)$, each defined in the frequency domain along a radial line set at an angle θ. Several angles need to be considered. Here, $S_\theta(\omega)$ is the Fourier transform of $P_\theta(\rho)$. Also,

$$S_{\theta(\omega)} = F(\omega\cos\theta, \omega\sin\theta)$$

as a consequence of the Fourier Slice Theorem.
(3) Compute the inverse Fourier transform of each function $S_\theta(\omega)$. The result is a set of projections $P_\theta(\rho)$ which together form the Radon transform of the image.

The efficiency of the technique is due to the fact that the mapping from image space to feature space is made via the frequency domain. Consequently, much of the computation can be carried out by repeatedly using the Fast Fourier Transform (FFT)

The Radon transform will generate prominent and easily detectable peaks in feature space when it is used on images which contain linear features extending across the image.

A drawback of the Radon transform method for line detection is that it cannot be used to detect linear features with size much less than that of the image itself.

This is due to the fact that short lines may not generate peaks or troughs in the Radon transform even though they might be very bright or dark (in intensity), since less intense, longer lines may produce Radon transform values of equal magnitude.

Even when a small local maximum (minimum) occurs, it is hard to distinguish between two possibilities
(1) The maximum (minimum) is due to a high intensity (dark) short line.
(2) The maximum (minimum) is due to a much longer line with less intensity.

A second drawback of the Radon transform method for line detection is the fact that it is unable to extract information on the coordinates of the endpoints of detected lines, or on the distance between endpoints.

The use of the Radon transform for line detection, along with the invertibility property of the Radon transform operator, allows for a linear feature enhancement method in which an enhancement operator is applied to the Radon transform of the image in feature space. Let s denote the Radon transform of an image, i. Then s = Ri, where R is the Radon transform operator. Consider an enhancement operator, E, which raises the peaks and lowers the valleys in feature space to yield a modified Radon transform s' where $s' = Es = ERi$.

As a consequence of the invertibility characteristic of the Radon transform, there exists an image i' with Radon transform s'. Therefore, $i' = R^{-1}s' = R^{-1}ERi$.

Any bright (dark) linear features in the original image will appear in the enhanced image i' with greater (less) mean intensity, for an appropriately chosen enhancement operator. Appropriate filtering operators are those which raise the peaks and lower the valleys in the Radon transform by large amounts while not affecting the values lying between the maximum and minimum much if at all. Operators which square or cube the difference between each Radon transform value are examples of such suitable operators.

The linear feature enhancement technique can be implemented using the Fourier transform to compute the Radon transform and the filtered-Backprojection method to compute the inverse Radon transform. Filtered-Backprojection is based on the following equations which allow an intensity function f(x,y) to be expressed in terms of its Radon transform $P_\theta(\rho)$. The reconstructed image f(x,y) can be expressed as follows:

$$f(x,y) = \int_0^\pi Q_\theta(t)d\theta$$

where

$$Q_\theta(t) = \int_{-\infty}^{\infty} S_\theta(\omega)|\omega| \exp(i\omega t)d\omega$$

with

$$S_\theta(\omega) = \int_{-\infty}^{\infty} P_\theta(\rho) \exp(-i\omega\rho)d\rho$$

and

$$t = x\cos\theta + y\sin\theta$$

The filtered-Backprojection technique can be described in the following steps: (1) Compute the Fourier transforms, $S_\theta(\omega)$ of each projection $P_\theta(\rho)$.

(2) Filter the results by multiplying by $|\omega|$ in the frequency domain.

(3) Compute the inverse Fourier transforms, $Q_\theta(t)$ of each product $S_\theta(\omega)|\omega|$.

(4) Backproject the functions Q_θ(t) over the the image plane to form the reconstructed image f(x,y). This involves calculating the contribution made by each Q_θ(t) to the reconstructed image. Since each function Q_θ(t) provides the same contribution to the image at all points (x,y) on the line t = xcos θ+ysin θ, the image is effectively reconstructed by combining a large number of lines.

Filtered-Backprojection in general provides excellent reconstructions. Reconstruction using backprojection allows better resolution than the interpolation method. It also induces greater noise because the filter tends to amplify high-frequency content.

The filtered-backprojection algorithm can be summarized with the following five steps:

(1) Compute projections $g(\rho, \theta)$ obtained at each fixed angle θ.

(2) Compute the Fourier transform $G(\omega, \theta)$ of each projection $g(\rho, \theta)$.

(3) Multiply $G(\omega, \theta)$ by the filter function $|\omega|$.

(4) Compute the inverse of the results from 3.

(5) Integrate (sum) over all θ the results from 4.

Chapter 7

Use of Optical Flow for Warping One Edge-Detected Crack Image into Another

Consider an image intensity function given by I(x(t),y(t),t) for moving objects in a sequence of image frames. Then, under the assumption that the objects retain their luminance along the path on which they travel (the constant brightness assumption), I(x(t),y(t),t) is constant with respect to time. Then upon taking the time derivative of I, we obtain:

$$\frac{\partial I}{\partial x}\frac{dx}{dt} + \frac{\partial I}{\partial y}\frac{dy}{dt} + \frac{\partial I}{\partial t} = 0 \tag{7.1}$$

$$\text{Let } v_1 = \frac{dx}{dt} \text{ and } v_2 = \frac{dy}{dt}$$

Then equation (7.1) can be expressed as

$$\frac{\partial I}{\partial x}v_1 + \frac{\partial I}{\partial y}v_2 + \frac{\partial I}{\partial t} = 0$$

This is an example of an inverse problem in which the image intensity function, I, is known, but the optical flow variables v_1 and v_2 are unknown and must be found. There are a number of methods for finding the optical flow associated with objects in an image. The method that was used for this work was the TV-L^1 norm, based upon a paper by [46]. In that paper, two image frames, I$_0$ and I$_1$:$\Omega \rightarrow$ R are considered. The goal is to determine a function u:$\Omega \rightarrow R^2$, which minimizes the sum of an image-based error constraint and a regularization term. The map u can then be found by minimizing

$$\int_\Omega \{\lambda \phi I_0(x) - I_1(x + u(x))) + \psi(u, \nabla u, ...)\}dx$$

where $\phi(I_0(x) - I_1(x + u(x)))$ is the image data fidelity term or optical flow constraint term, and $\psi(u, \nabla u, ...)$ is the regularization term. λ provides a weighting between data fidelity and the regularization force. Taking $\phi(x) = |x|$ and $\psi(\nabla u) = |\nabla u|$ produces a functional with an L^1 data penalty term and a total variation regularization term given by

$$E = \int_\Omega \{\lambda |I_0(x) - I_1(x + u(x))| + |\nabla u|\}dx \tag{7.2}$$

This equation presents computational problems since neither the data fidelity term nor the regularization term is continuously differentiable. The authors then utilize Chambolle's method for solving the Rudin-Osher-Fatemi energy for estimating a denoised image using total variation. In particular that method is adapted to estimating optical flow. The authors then consider a first-order Taylor approximation to image I$_1$ near x+u$_0$ which is given by

$$I_1(x + u) = I_1(x + u_0) + (u - u_0) + \frac{\partial I_1}{\partial x}(x + u_0)$$

where u$_0$ is a given discrepancy map. For a fixed u$_0$ and utilizing the first-order approximation for I$_1$, the TV-L^1 functional in equation (7.2) can now be expressed as

$$E = \int_\Omega \{\lambda |u \frac{\partial I_1}{\partial x} + I_1(x + u_0) - u_0 \frac{\partial I_1}{\partial x} - I_0| + |\nabla u|\}dx \tag{7.3}$$

Utilizing the first-order Taylor approximation to image I$_1$ requires that the functional in equation (7.3) be incorporated into an iterative warping technique to offset nonlinearities in the image. Also, a multi-level method is used to deal with large discrepancies between images. Let $\rho(u, u_0, x)$ (or just $\rho(u)$ be equal to the residual term. That is

$$\rho(u, u_0, x) = I_1(x + u_0) + (u - u_0)\frac{\partial I_1}{\partial x} - I_0$$

Then, introducing a variable v, the authors seek to minimize a convex approximation to the functional in equation (7.3).

$$E_\theta = \int_\Omega \{|\nabla u| + \frac{1}{2\theta}(u-v)^2 + \lambda|\rho(v)|\}dx$$

where θ is a small real number, chosen so that v is a close approximation of u. This convex optimization problem can be solved by alternately updating u and v in every iteration. For fixed v, solve the following minimization problem.

$$\min_u \int_\Omega \left\{ |\nabla u| + \frac{1}{2\theta}(u-v)^2 \right\} dx$$

For fixed u, solve the following minimization problem.

$$\min_v \int_\Omega \left\{ \frac{1}{2\theta}(u-v)^2 + \lambda|\rho(v)| \right\} dx$$

The authors then state results from other papers that provide the solutions of these two minimization problems.

Proposition 1: The solution of the first minimization problem is given by

$$u = v - \theta\nabla \cdot \vec{p}$$

where p = (p_1,p_2) satisfies

$$\nabla(\theta\nabla \cdot \vec{p} - v) = |\nabla(\theta\nabla \cdot \vec{p} - v)|\vec{p}$$

which can be solved via an iterative fixed-point method given by

$$\vec{p}_{k+1} = \frac{p_k + \tau\nabla(\nabla \cdot \vec{p}_k - v/\theta)}{1 + \tau|\nabla(\nabla \cdot \vec{p}_k - v/\theta)|}$$

where $p_0 = 0$ and the time step $\tau \leq \frac{1}{8}$

Proposition 2: The solution of the second minimization problem above is given by a thresholding step:

$$v = u + \begin{cases} \lambda\theta\frac{\partial I_1}{\partial x} & \text{if} \quad \rho(u) < -\lambda\theta\left(\frac{\partial I_1}{\partial x}\right)^2 \\ -\lambda\theta\frac{\partial I_1}{\partial x} & \text{if} \quad \rho(u) > \lambda\theta\left(\frac{\partial I_1}{\partial x}\right)^2 \\ \frac{-\rho(u)}{\frac{\partial I_1}{\partial x}} & \text{if} \quad |\rho(u)| \leq \lambda\theta\left(\frac{\partial I_1}{\partial x}\right)^2 \end{cases}$$

The algorithm described in the two propositions above was what was implemented in the MATLAB code found on GitHub.

Image 002-100 in Figure 7.1 is a crack image of pavement that was selected from a set of crack images available at
`https://digitalcommons.usu.edu/all\_datasets/48/` The dataset is called SDNET2018: A concrete crack image dataset for machine learning applications. The dataset consists of crack and non-crack images of pavement, walls, and bridges. After the edge maps were computed, the images which consisted of upper and lower cracks, were segmented into two separate crack images by partitioning the column width of each edge map into many subintervals and approximating the crack by a linear or constant function in each subinterval. The optical flow MATLAB code was then used to warp one of these cracks into the other. First, the image of a pavement crack in

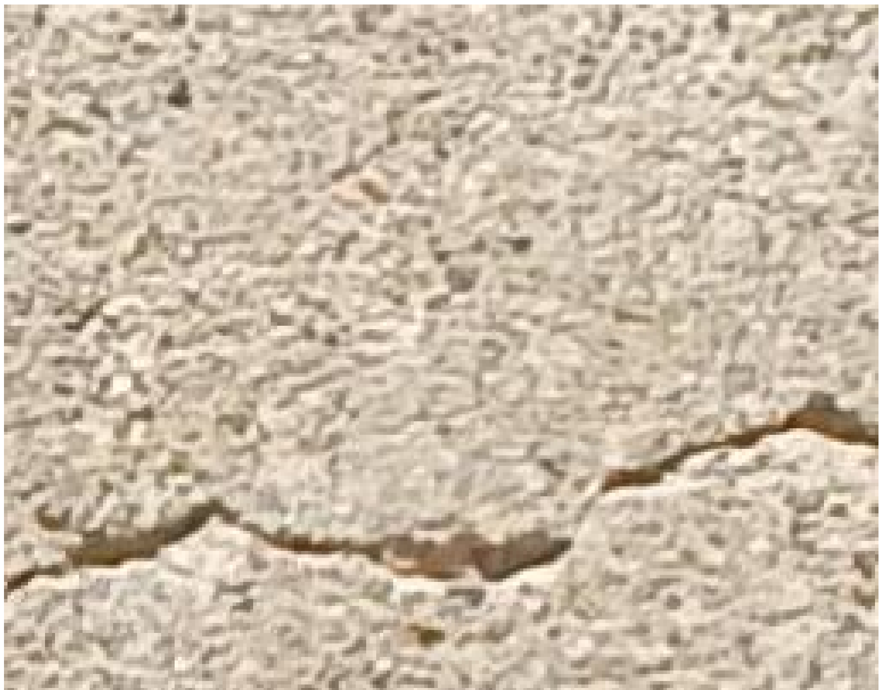

Figure 7.1: Input Image 002-100 of a Pavement Crack

figure 7.1 was smoothed using a 5x5 averaging, low-pass filter. Then, an edge map was generated from the image by computing the magnitude of the gradient across the image using the 3x3 Sobel approximations to the horizontal and vertical partial derivatives of the image. The binary edge map was produced by thresholding the magnitude of the intensity gradient at 22% of the maximum intensity value in the image. In particular, the magnitude of the gradient is approximated as

$$|\nabla I| = \sqrt{I_x^2 + I_y^2}$$

where I$_x$ is approximated by

$$I_x \approx S * S_x \text{ where * denotes convolution}$$

where S$_x$ is the Sobel vertical operator and S is the 5x5 low-pass filter, both discussed in chapter 4. where I$_y$ is approximated by

$$I_y \approx S * S_y \text{ where * denotes convolution}$$

where S$_y$ is the Sobel horizontal operator discussed in chapter 4.

Figure 7.2 shows the edge map after applying the smoothing filter and the approximation to the magnitude of the gradient to the image.

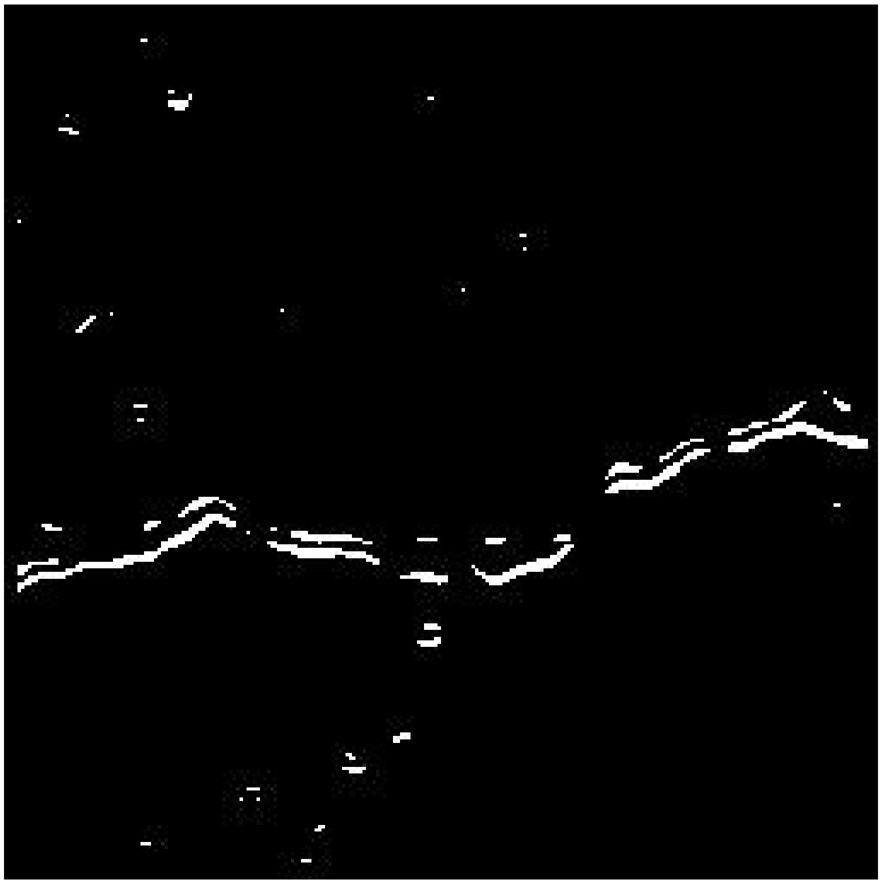

Figure 7.2: Edge Map

The image in figure 7.2 was carefully cut into two separate images shown in figure 7.3 and 7.4, using a piecewise linear function defined on appropriate subintervals of the column domain to separate them or segment the original edge map via thresholding. The piecewise linear function was chosen as functions of the columns contained in each subinterval.

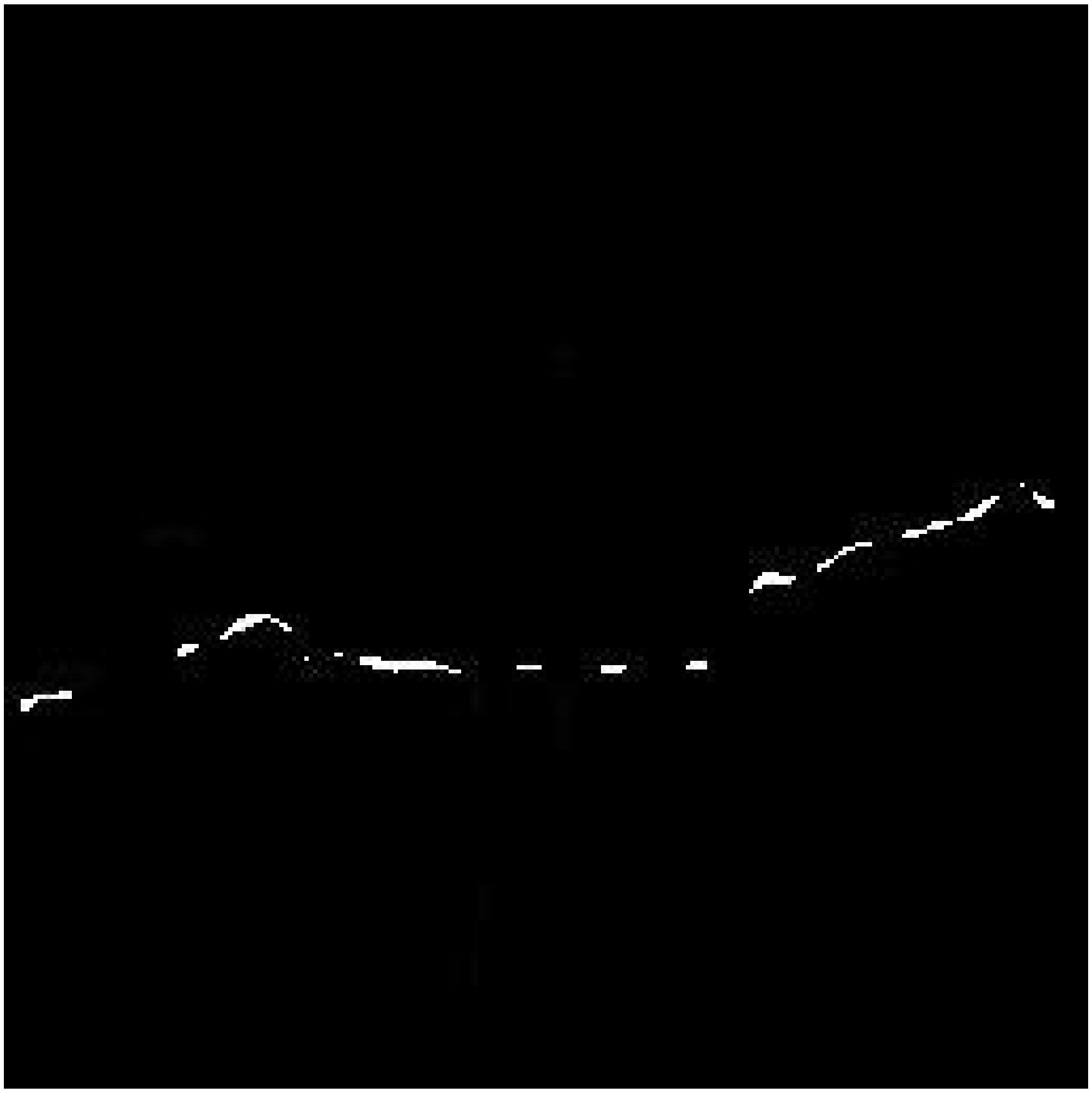

Figure 7.3: Upper Edge

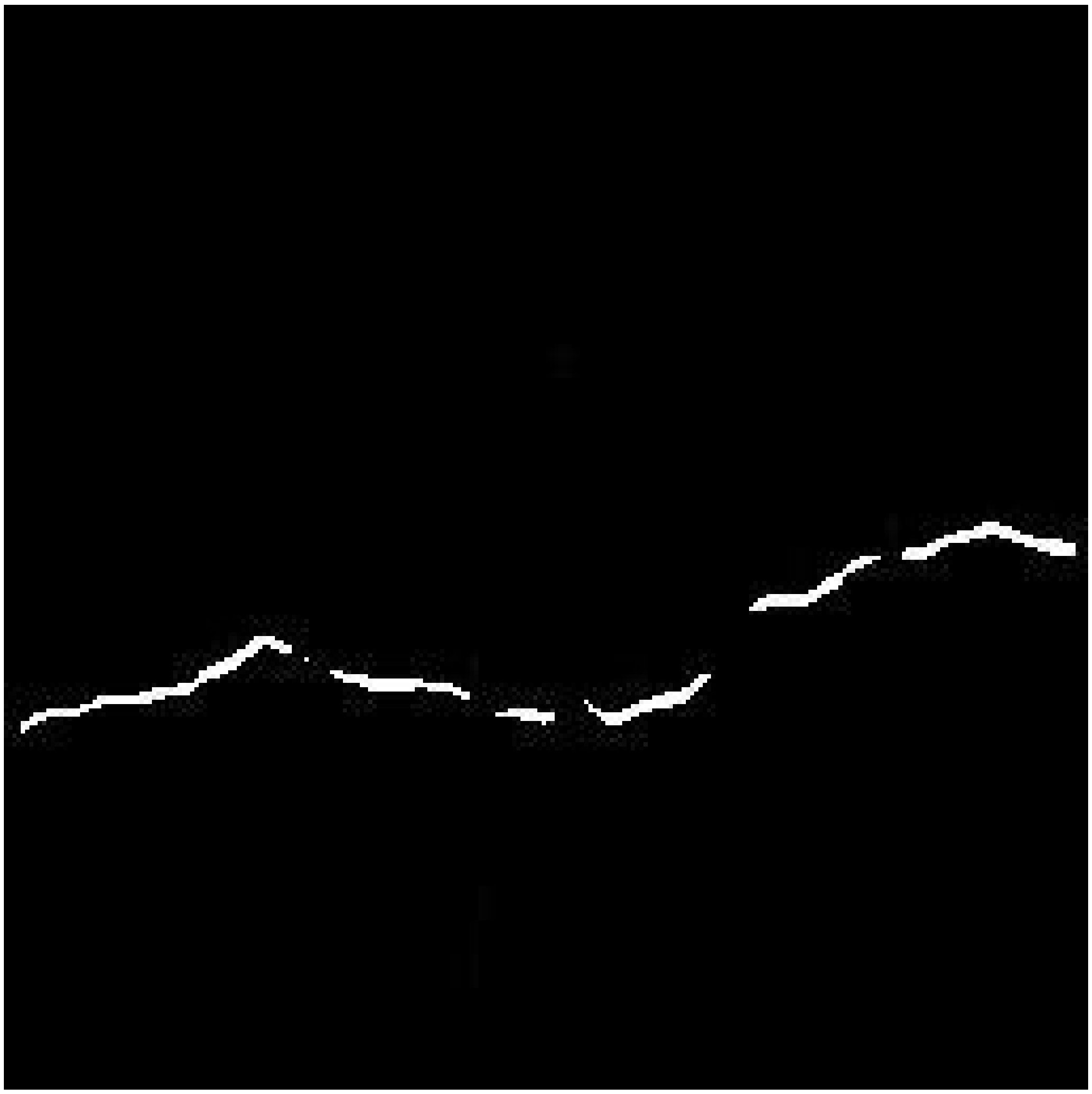

Figure 7.4: Lower Edge

Applying the warping optical flow MATLAB code to the images in figures 7.3 and 7.4 yielded the image in figure 7.5

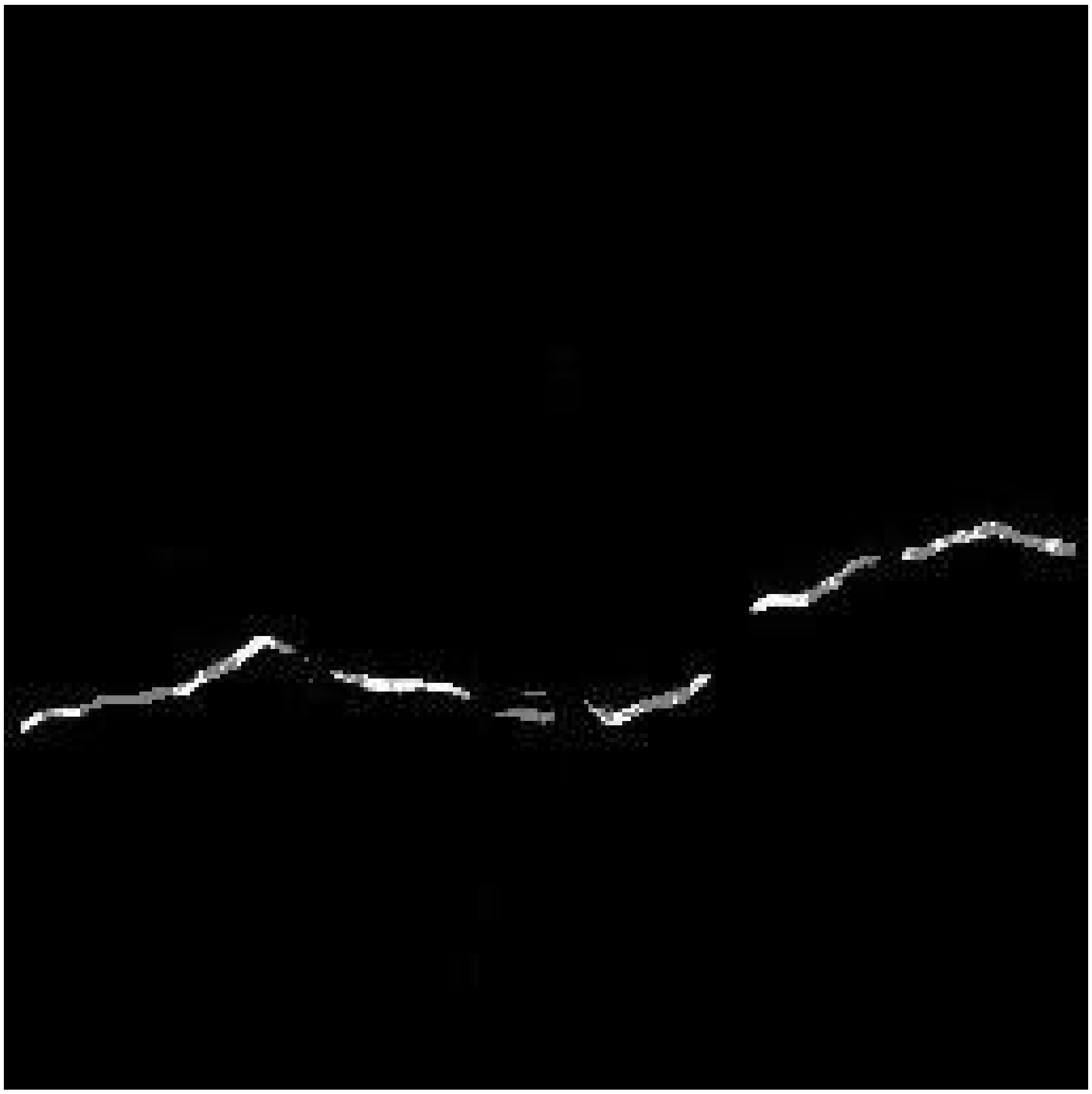

Figure 7.5: Merged Upper and Lower Edges of Pavement Crack after Applying Optical Flow Warping

The second image, in figure 7.6, was of a bridge deck crack. Its edge map was formed by thresholding the the magnitude of the intensity gradient at 10% of the maximum gradient intensity value in the image.

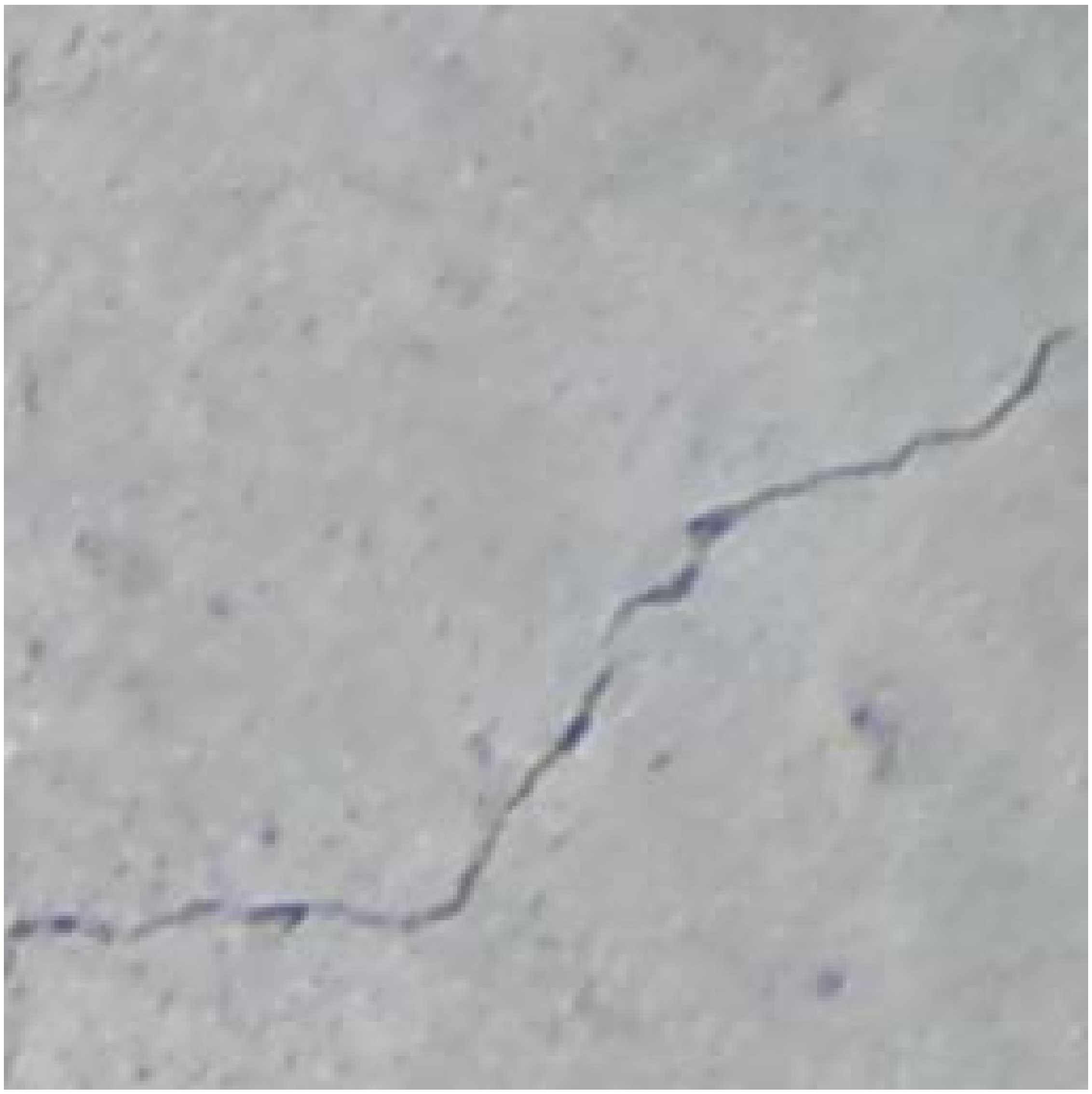

Figure 7.6: Image 7003-18 of a Bridge Deck Crack

Figures 7.7 and 7.8 show the edge map after it as been separated into two images, an upper image and a lower image.

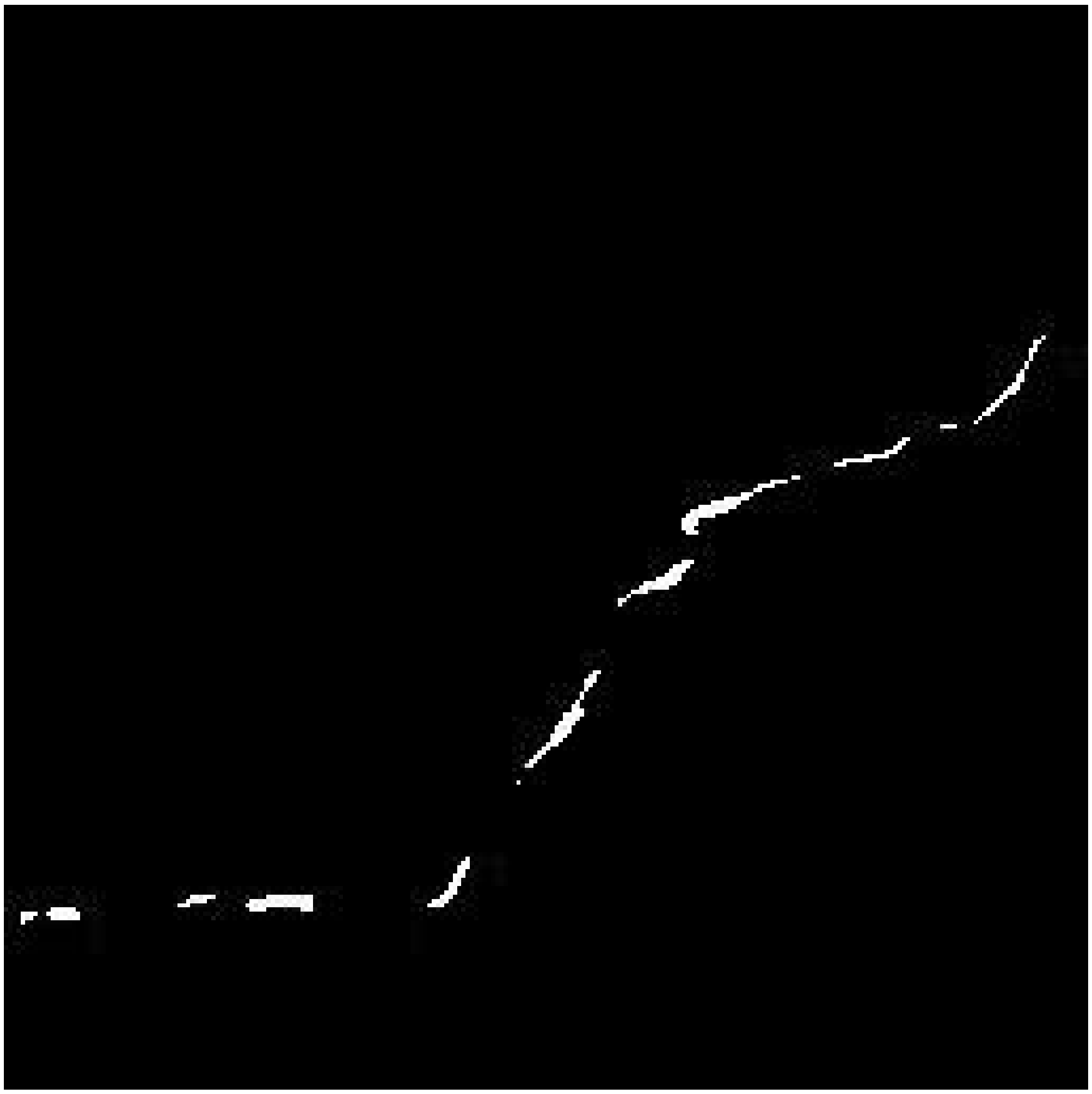

Figure 7.7: Image of Upper Portion of Edge Map for Bridge Deck Crack

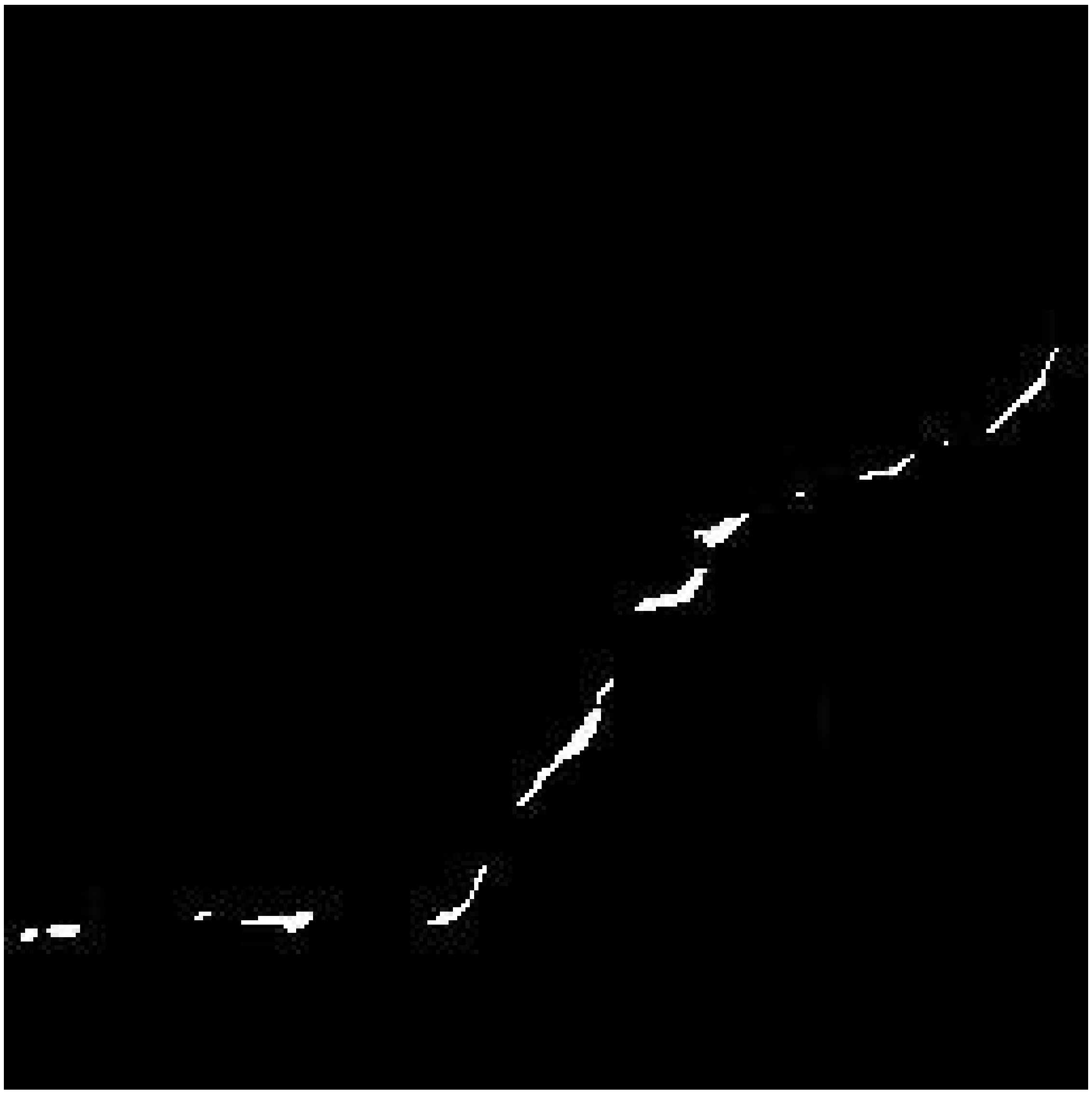

Figure 7.8: Image of Lower Portion of Edge Map for Bridge Deck Crack

Figure 7.9 shows the result of applying the optical flow warping code to merge the two crack portions of the bridge deck crack into one crack

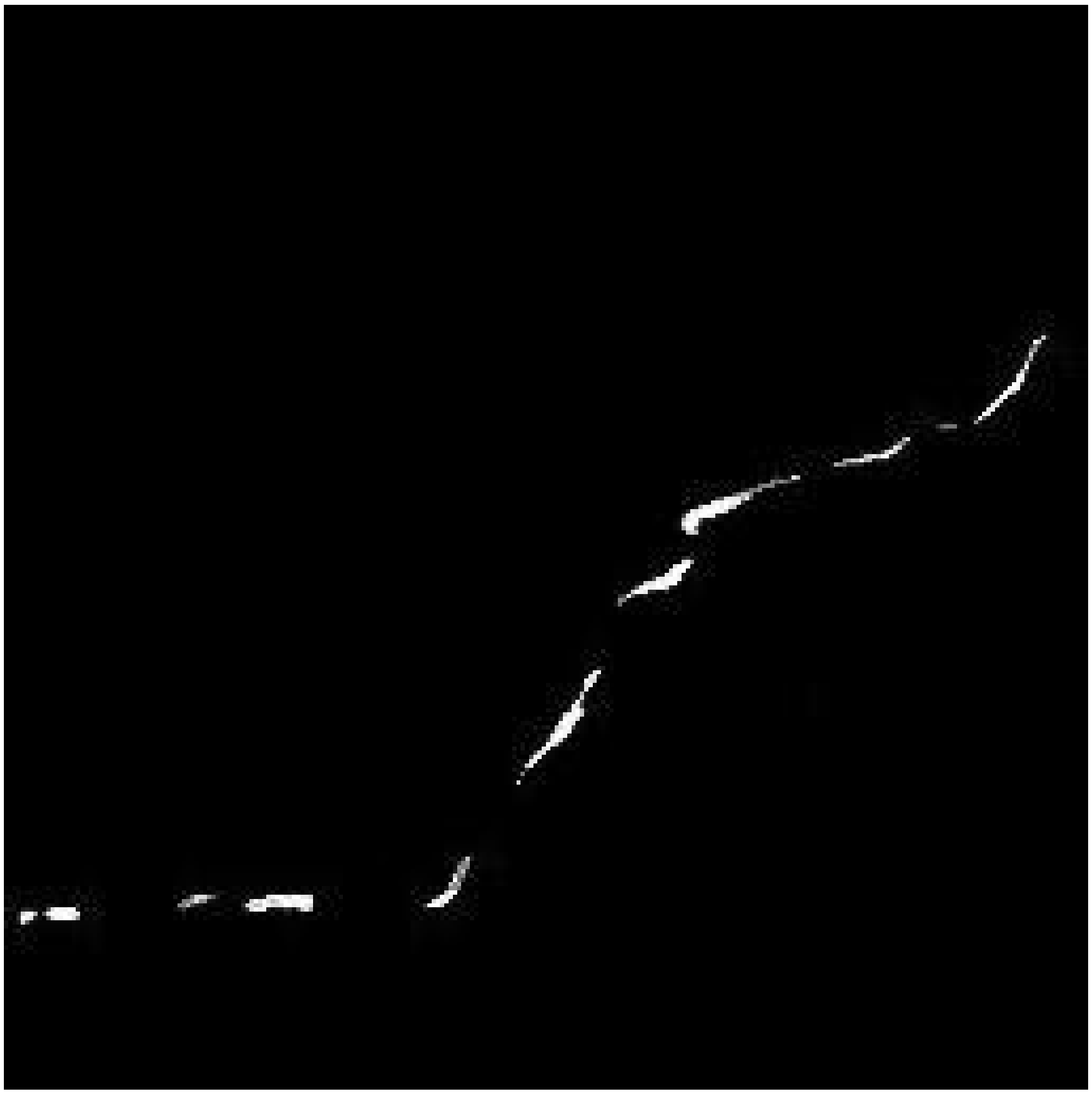

Figure 7.9: Merged Upper and Lower Edges of Bridge Deck Crack After Applying Optical Flow Warping

The third image, in figure 7.10 , was of a wall crack. Its edge map was formed by thresholding the the magnitude of the intensity gradient at 10% of the maximum gradient intensity value in the image.

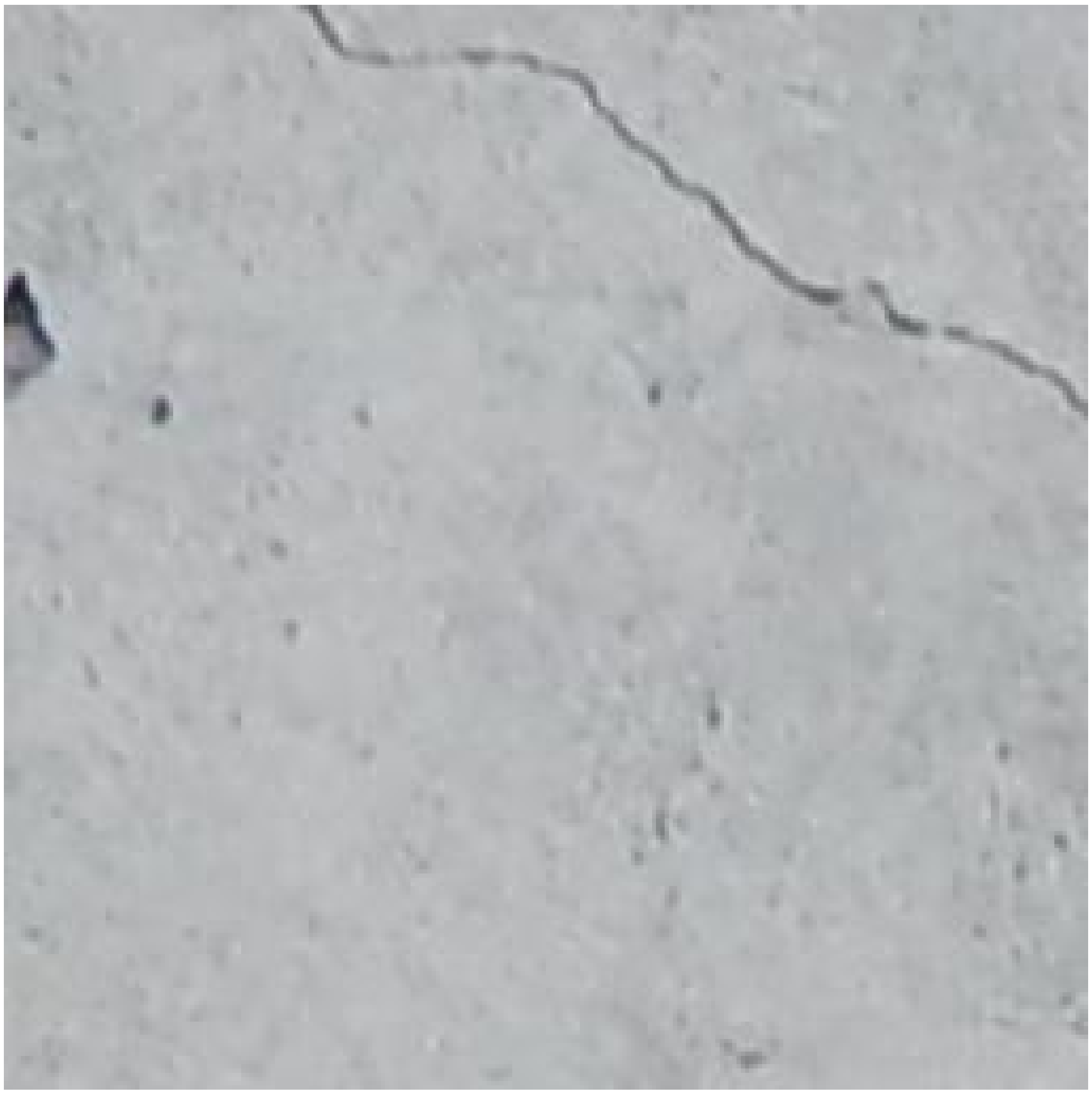

Figure 7.10: Image 7070-24 of a Wall Crack

Figures 7.11 and 7.12 show the edge map after it as been separated into two images, an upper image and a lower image.

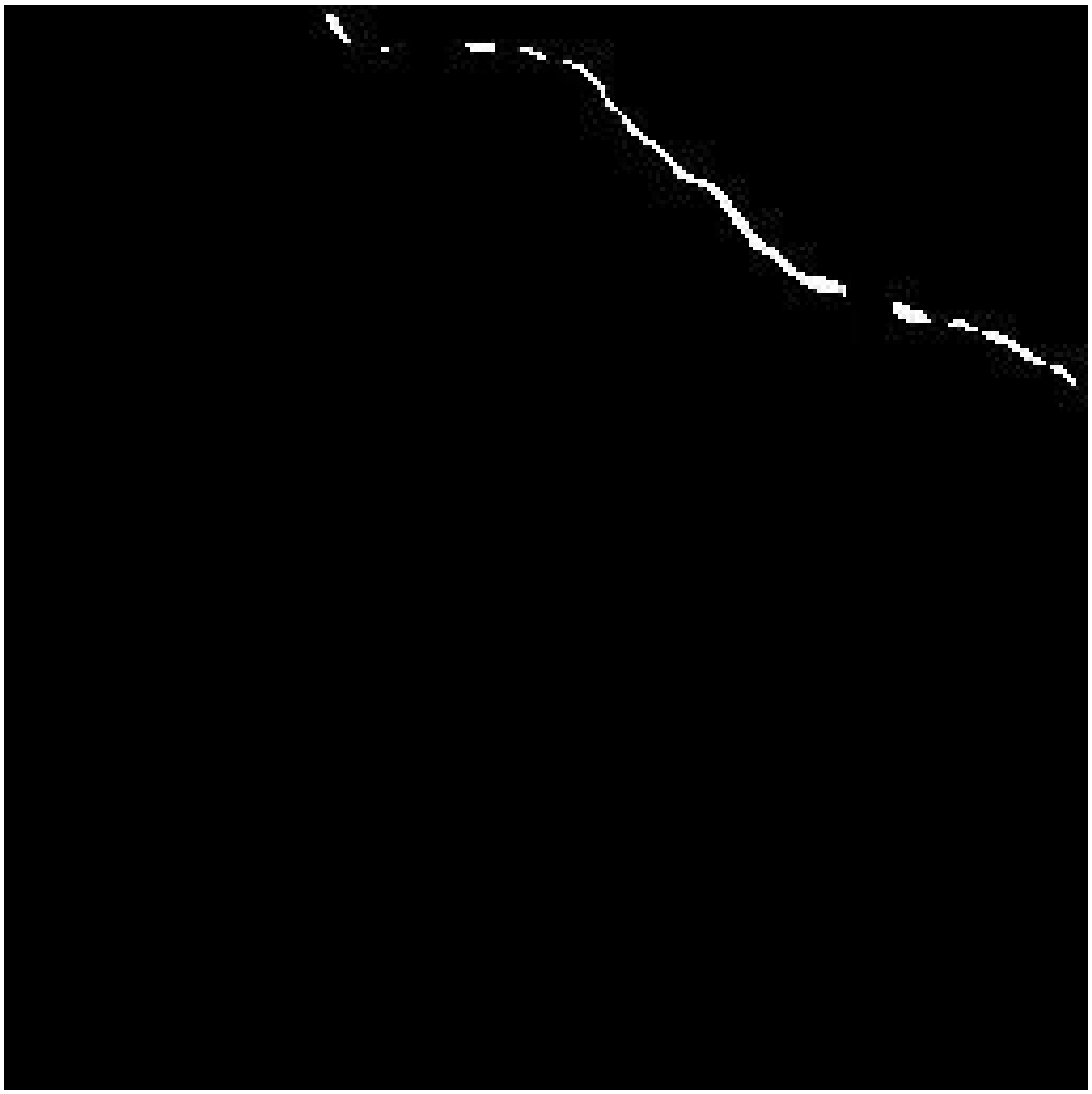

Figure 7.11: Image of Upper Portion of Edge Map for Wall Crack

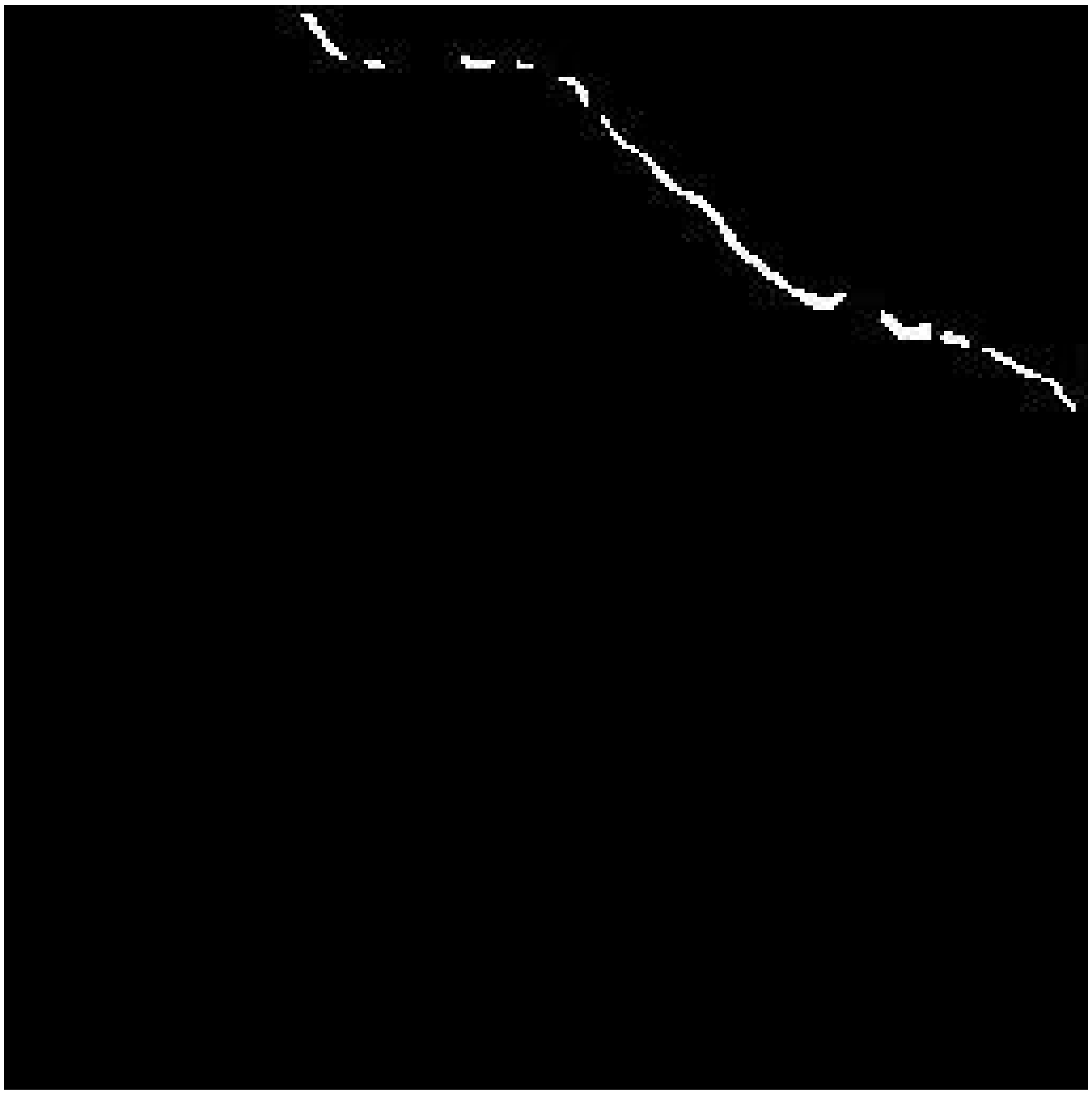

Figure 7.12: Image of Lower Portion of Edge Map for Wall Crack

Figure 7.13 shows the result of applying the optical flow warping code to merge the two crack portions of the wall crack into one crack

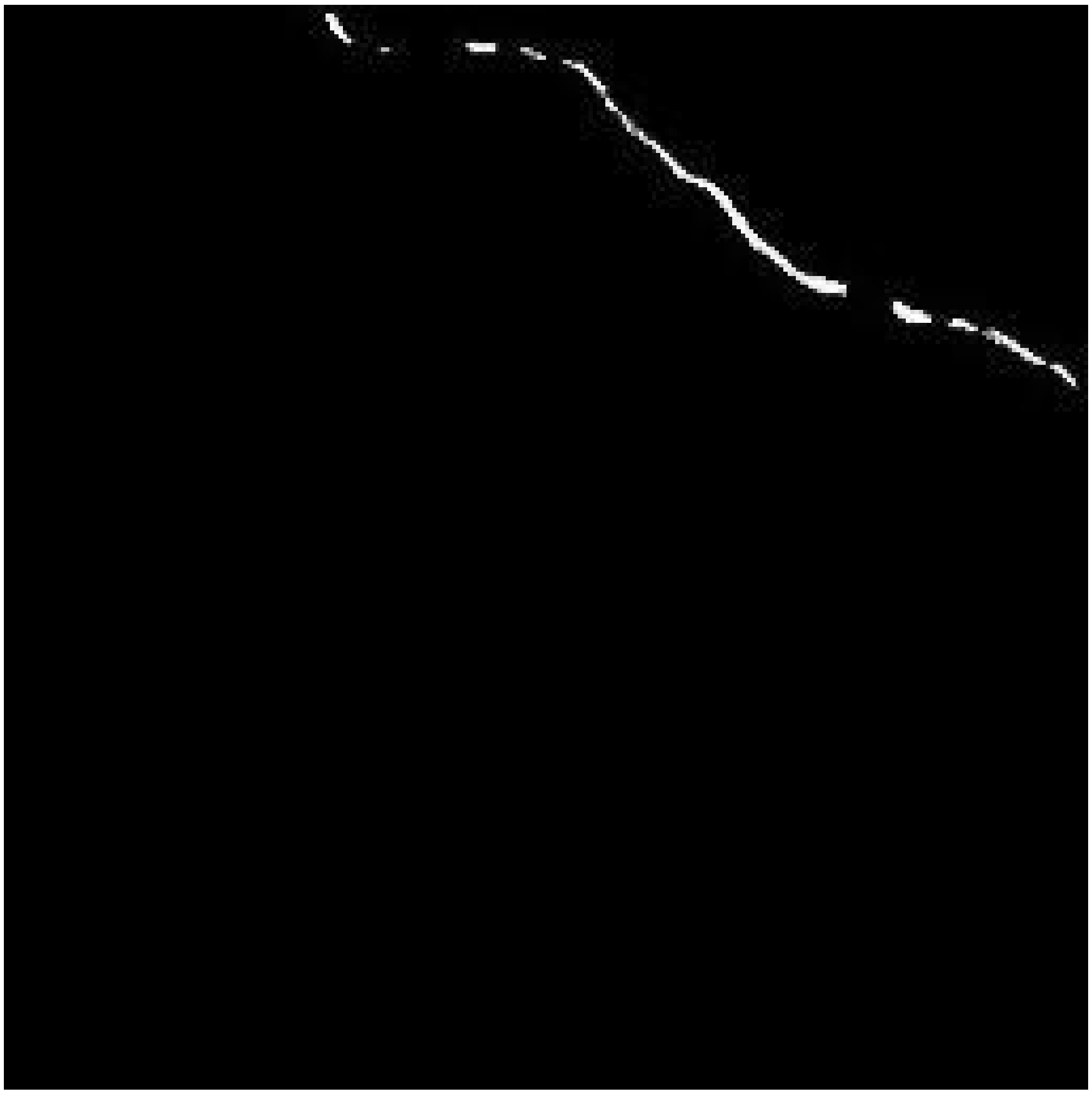

Figure 7.13: Merged Upper and Lower Edges of Wall Crack After Applying Optical Flow Warping

The fourth image, in figure 7.14, was of a pavement crack. Its edge map was formed by thresholding the the magnitude of the intensity gradient at 22% of the maximum gradient intensity value in the image.

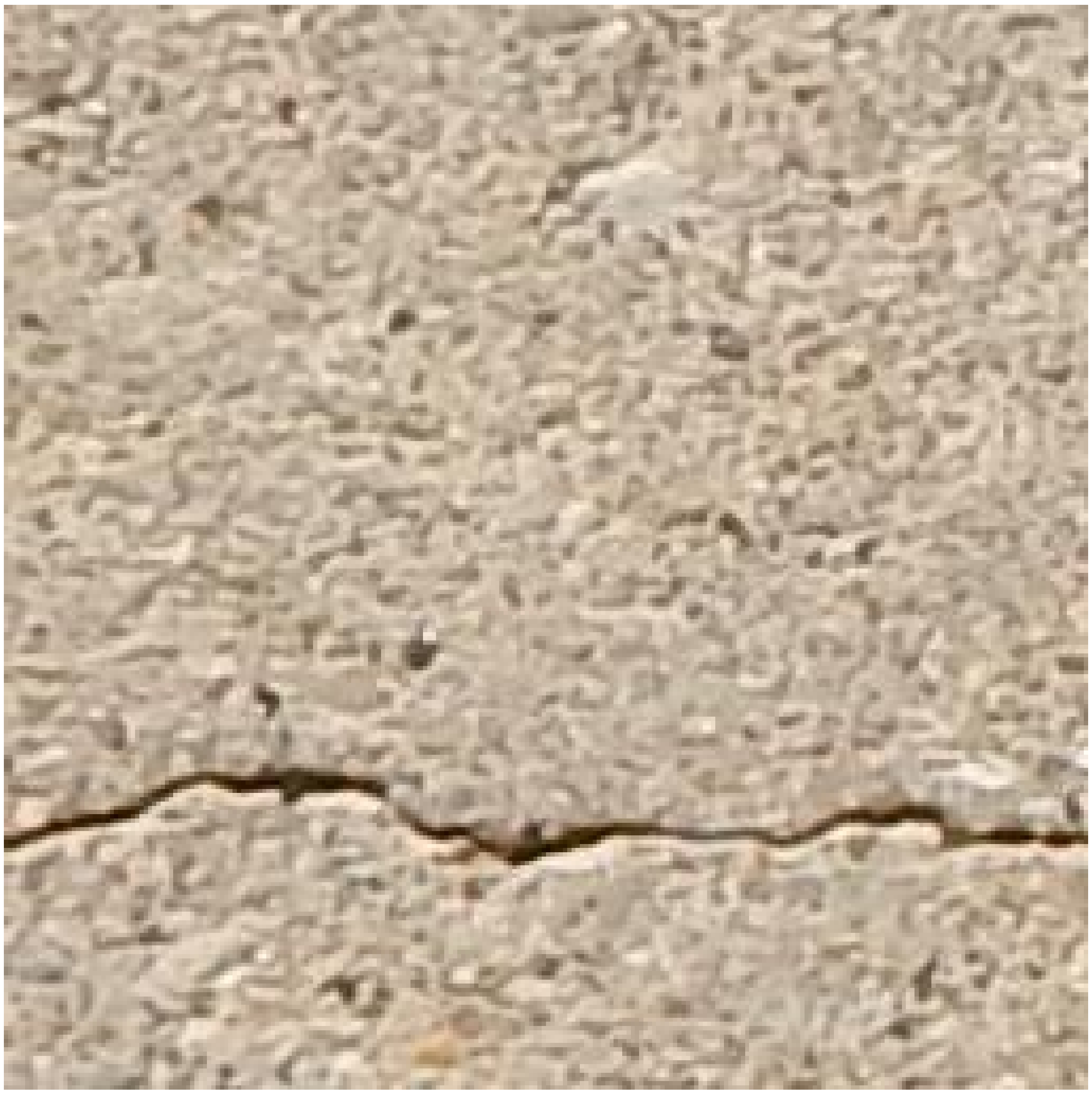

Figure 7.14: Image 004-114 of a Pavement Crack

Figures 7.15 and 7.16 show the edge map after it as been separated into two images, an upper image and a lower image.

Figure 7.15: Image of Upper Portion of Edge Map for Pavement Crack

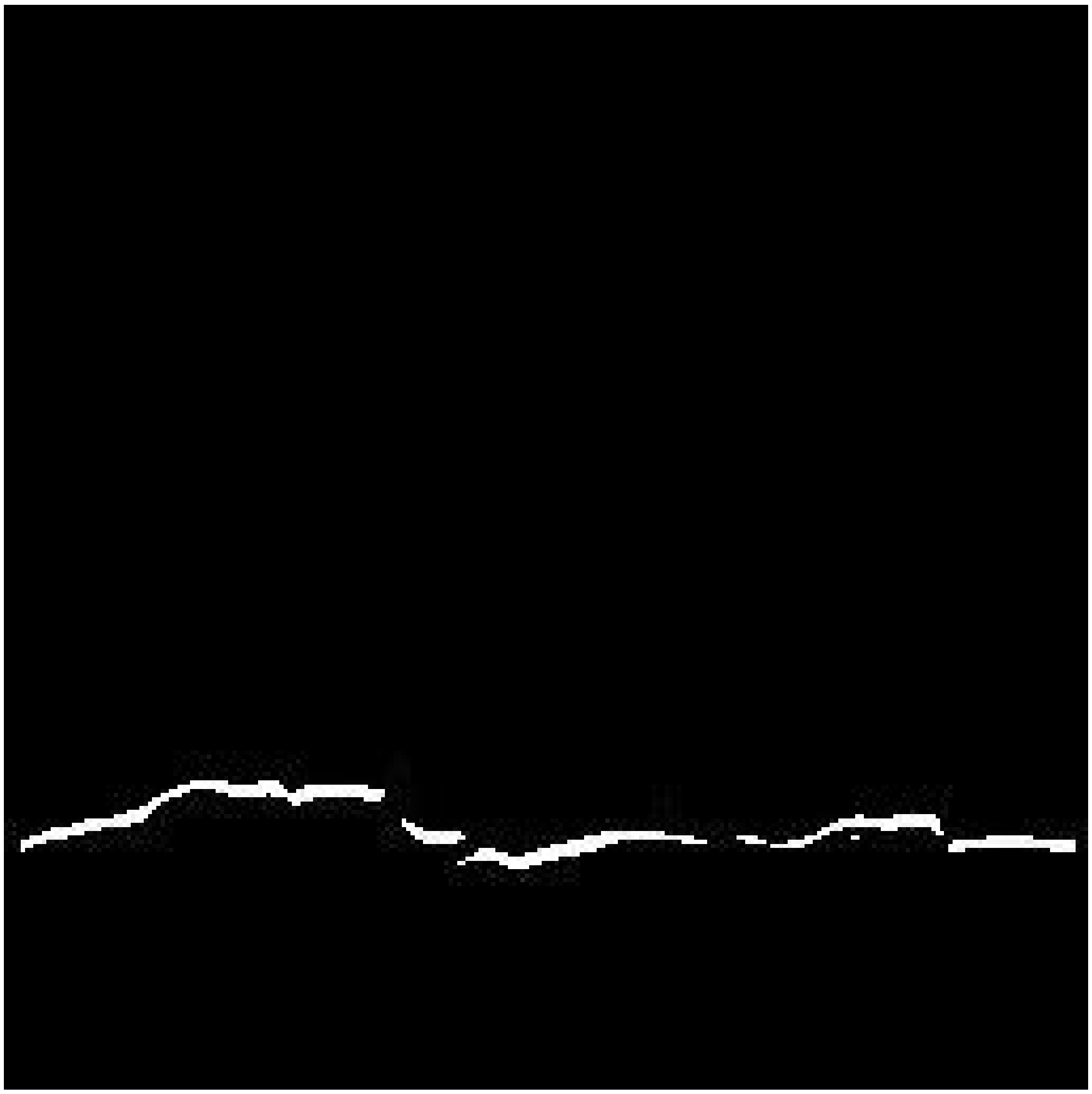
Figure 7.16: Image of Lower Portion of Edge Map for Bridge Deck Crack

Figure 7.17 shows the result of applying the optical flow warping code to merge the two crack portions of the pavement crack into one crack

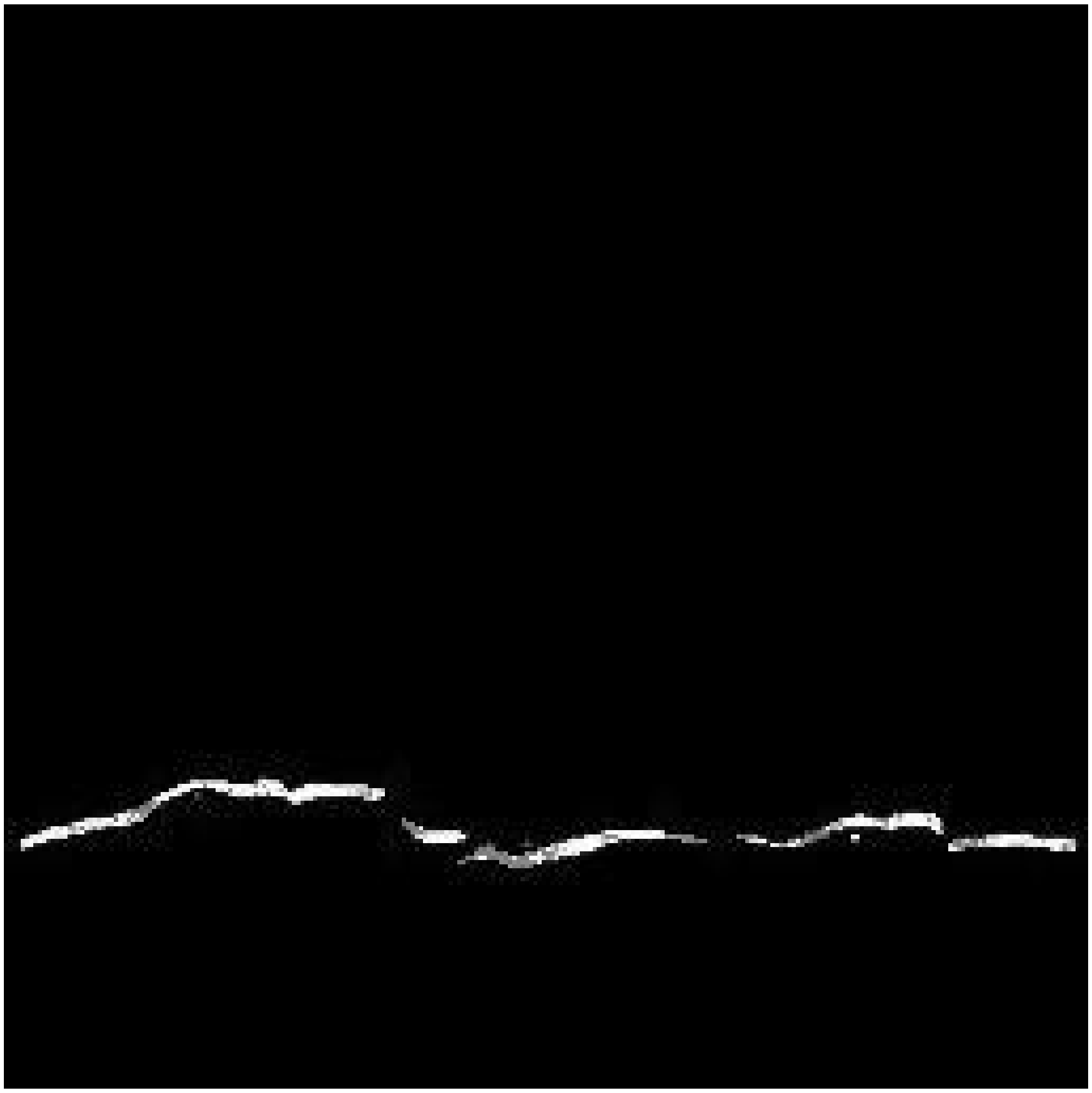

Figure 7.17: Merged Upper and Lower Edges of Pavement Crack After Applying Optical Flow Warping

The fifth image, in figure 7.18, was of a bridge deck crack. Its edge map was formed by thresholding the the magnitude of the intensity gradient at 8% of the maximum gradient intensity value in the image.

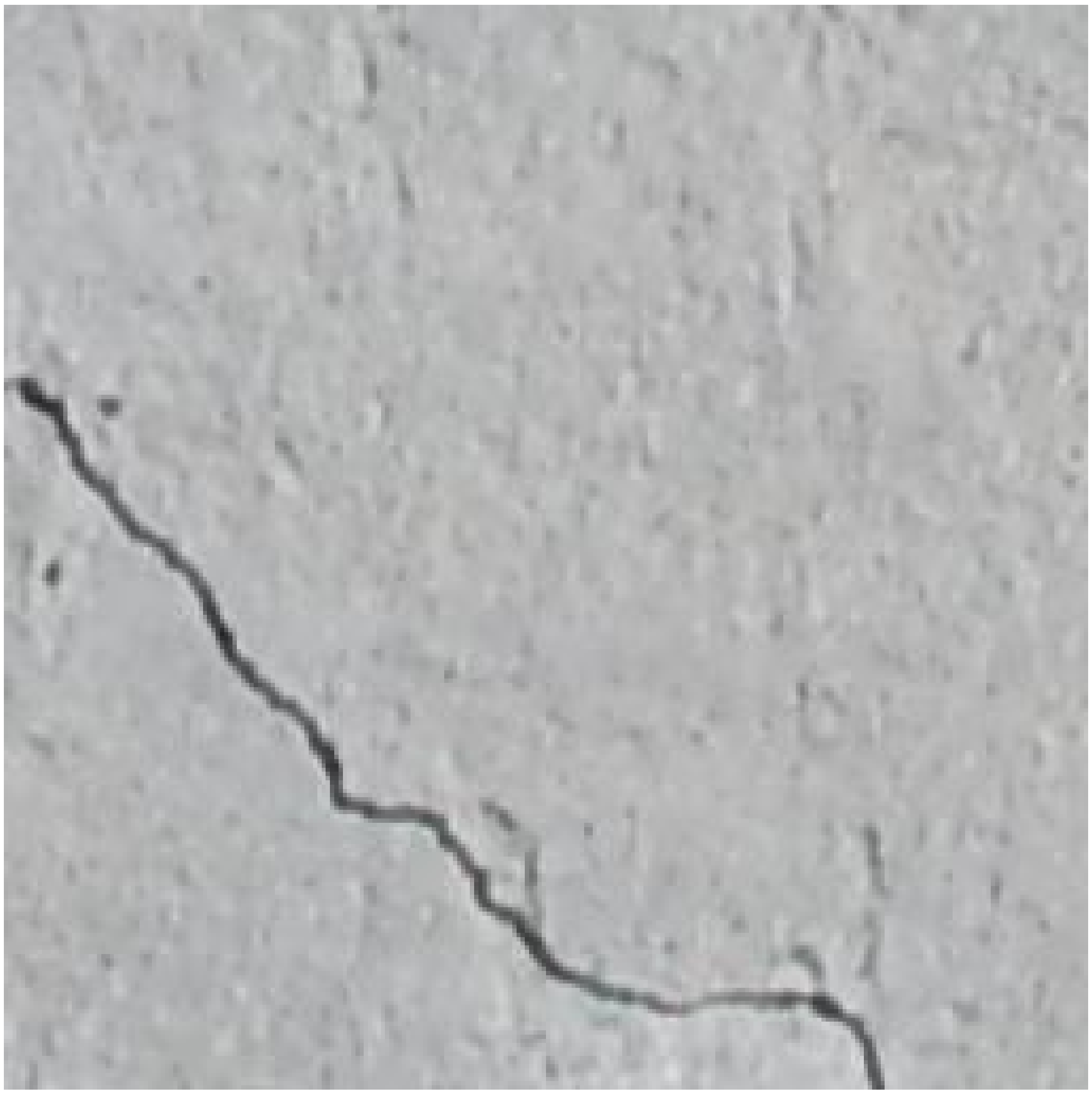

Figure 7.18: Image 7002-28 of a Bridge Deck Crack

Figures 7.19 and 7.20 show the edge map after it as been separated into two images, an upper image and a lower image.

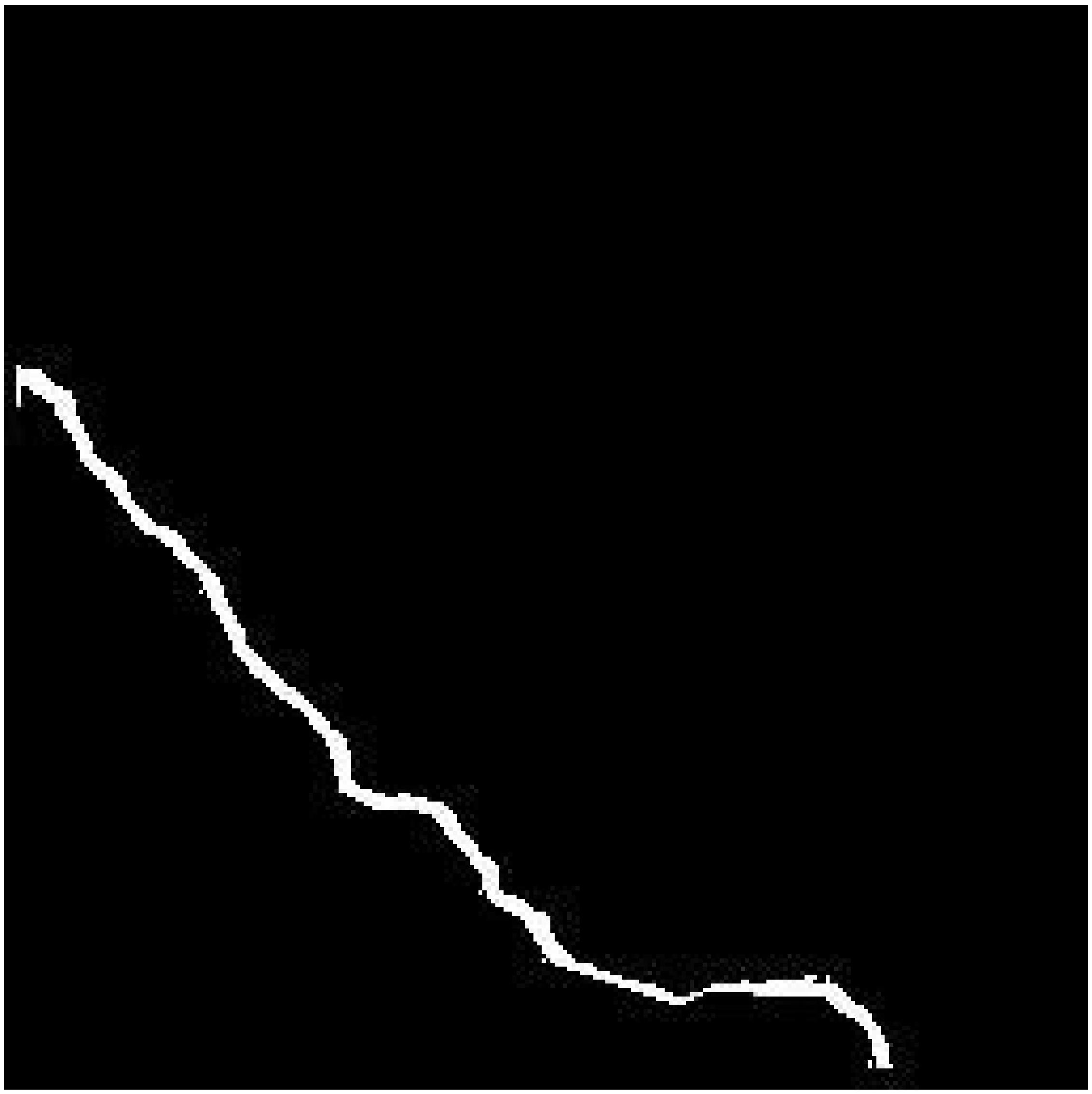

Figure 7.19: Image of Upper Portion of Edge Map for Bridge Deck Crack

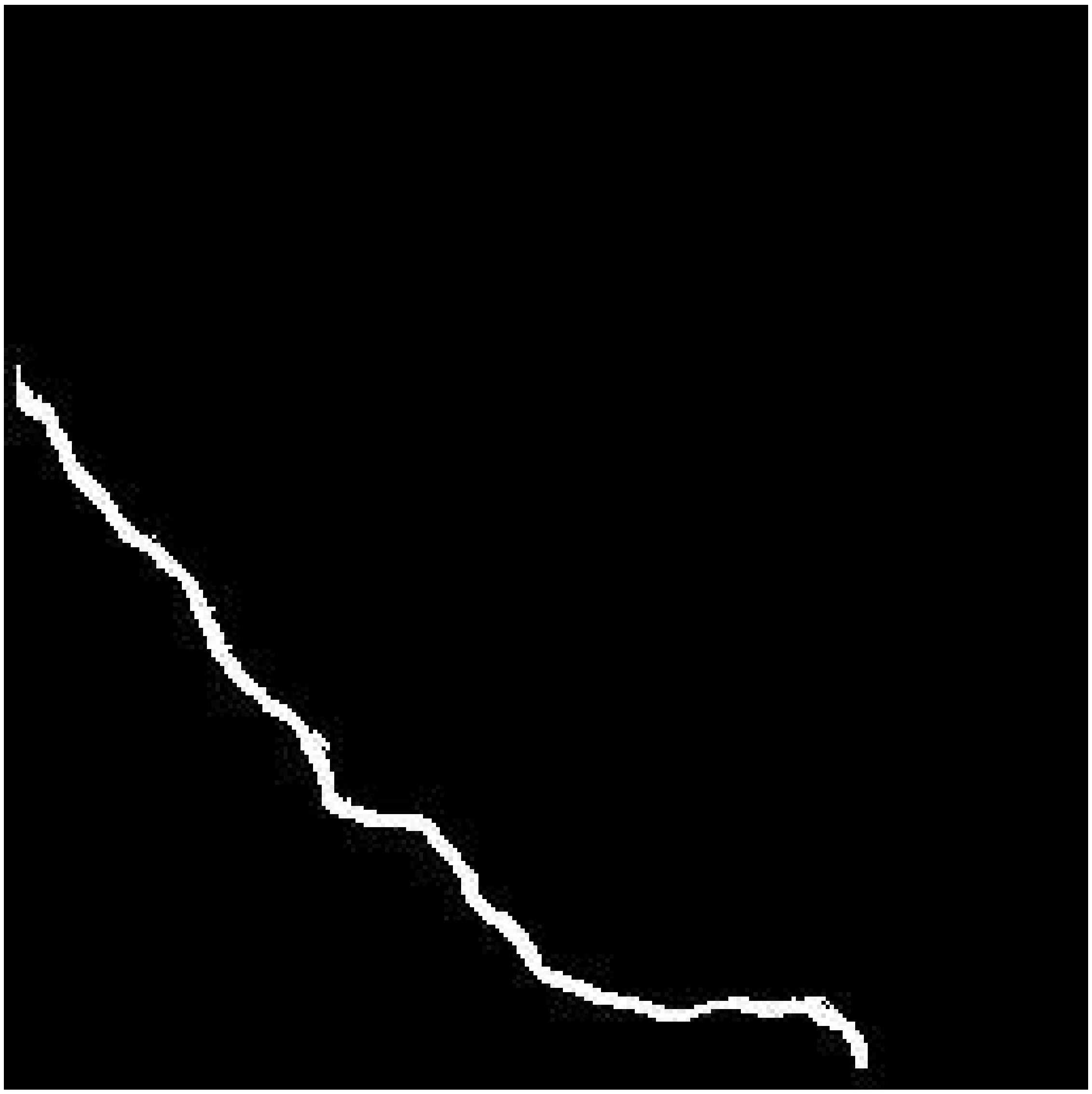

Figure 7.20: Image of Lower Portion of Edge Map for Bridge Deck Crack

Figure 7.21 shows the result of applying the optical flow warping code to merge the two crack portions of the bridge deck crack into one crack

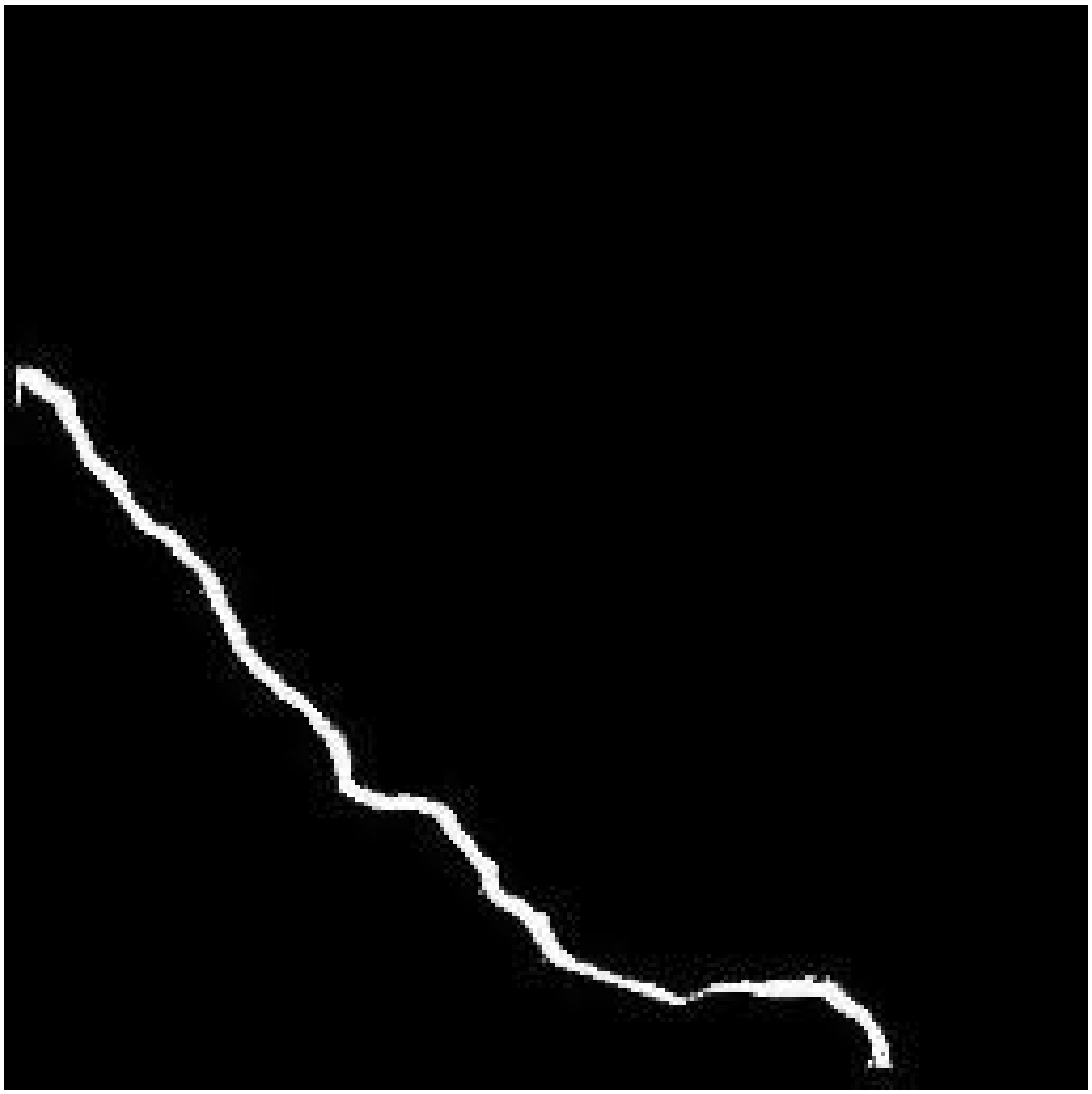

Figure 7.21: Merged Upper and Lower Edges of Bridge Deck Crack After Applying Optical Flow Warping

Chapter 8

Image Processing Algorithms for Feature Detection

8.1 Crack Endpoint Detection

A crack endpoint detection algorithm was generated that works for the upper and lower portions of a crack image. Different code is required for different cracks. MATLAB code was written which works for the upper and lower portions of the crack image 7070-24 which is a wall crack image. Separate MATLAB code was written for the upper and lower portions of the crack image 7003-18 which is a bridge deck crack.

The MATLAB code works by scanning across a crack image, which is a binary image, and first finding the left endpoints. Then, the right endpoints are found. A separate set of code within this MATLAB script works on the special case where a line segment portion of a crack which has other crack line segments beneath it, ends and then begins again.

Let I denote the image array. In the search for the left endpoints, a pixel (i,j) such that I(i,j)=255 is sought while simultaneously checking that I(i,j-1) = 0 and I(i+1,j-1) = 0, where i denotes the row and j the column in the image array. Then, if simultaneously, I(i-1,j) = 0, I(i-1,j-1) = 0, I(i+1,j-1) = 0 while I(i+1,j-1) ≠ 0 or I(i,j-1) ≠ 0 or I(i+1,j) = 0, then we declare that (i,j) is a crack left endpoint. Alternatively, if this last "if" condition is not satisfied, then if I(i+1,j) = 0 and I(i+1,j-1) = 0 and I(i-1,j) ≠ 0 and I(i-1,j-1) = 0 and I(i-2,j-1) = 0, then we declare that (i,j) is a crack left endpoint. This last if condition is necessary so that one of the left endpoints in the lower crack 7070-24-det-low will be detected an two endpoints in the upper crack 7070-24-det-up will be detected.

For the detection of the right endpoints, set a counting variable k = 0. Then, while I(i,j+k) > 0, we increment k. Then if I(i+1,j+k) = 0 and I(i+1,j+k+1) = 0 and I(i-1,j+k+1) = 0 and I(i-1,j+k+1) ≠ 0 or I(i,j+k+1) ≠ 0 or I(i-1,j+k) = 0, then we declare (i,j+k) to be a crack right endpoint. Alternatively, if this last "if" condition is not satisfied, then if I(i-1,j+k) ≠ 0 and I(i,j+k) ≠ 0 and I(i-2,j+k+1) = 0 and I(i,j+k) = 0 or I(i-1,j+k) = 0 or I(i-2,j+k) ≠ 0 or I(i+1,j+k) = 0 or I(i,j+k) = 0 and I(i,j+k) = 0 or I(i,j+k-1) ≠ 0 or I(i-1,j+k) = 0 or I(i-2,j+k) = 0 or I(i+1,j+k+1) = 0 and I(i,j+k+1) ≠ 0 or I(i-1,j+k+1) = 0 or I(i+1,j+k+1) = 0 and I(i,j+k) = 0 or I(i-1,j+k) = 0 or I(i+1,j+k) = 0 or I(i+1,j+k+1) = 0, then we declare (i,j+k) to be a crack right endpoint. Without this alternative path, not all of the right endpoints would be detected. Finally, there is a special case to deal with one of the right endpoints in the upper portion of crack 7070-24-det or in 7070-24-det-up. There, we have a situation where a line segment stops and then restarts while the segments below it continue without interruption. To address this case, consider the vector of crack endpoints, ep. Let z = size(ep,2). Consider whether or not $|i - ep(z-1)| > |j + k - ep(z)|$ This expression checks whether or not the absolute value of the difference between the current row and the previous row coordinate of the crack endpoint is greater than the difference between the current column (j+k) and the previous column coordinate.If the row difference is greater than the column difference, then if I(i-1,j+k) = 0 and I(i-1,j+k+1) = 0 and I(i+1,j+k) ≠ 0, then we declare a crack right endpoint at (i,j+k).

For the bridge deck crack 7003-18, the same basic algorithm can be used to detect the crack endpoints if we add the following alternative to the search logic for the left endpoints: If I(i,j) = 255 and I(i,j-1) = 0 and I(i-1,j+1) = 0 and I(i-1,j) = 0, then execute the next set of statements.

Inserting this alternative condition into the search logic alters nothing in the detection of endpoints for the wall crack 7070-24 while allowing for the detection of the left endpoints (and thereby the associated right endpoints) for the bridge deck crack 7003-18. The image in Figure 8.1 was used

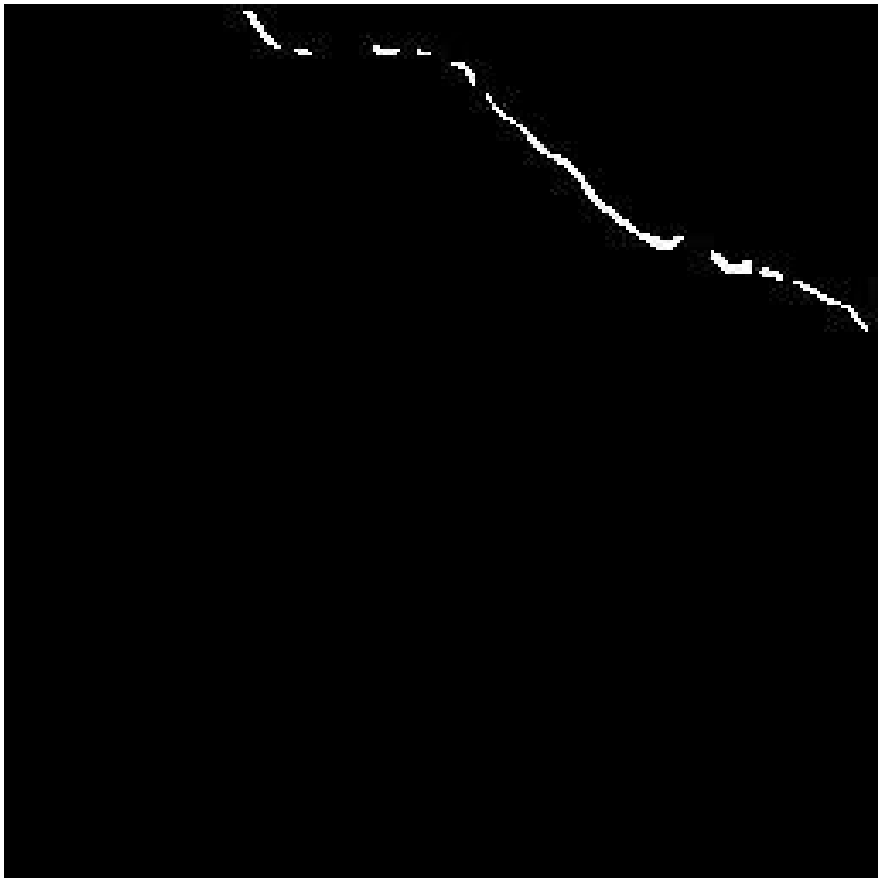

Figure 8.1: Input Image 7070-24-det-up.jpg

as an input to the crack endpoint detection algorithm. The result was the set of crack endpoints (4,72), (13,81), (14,86), (15,90), (14,109), (14,116), (15,122), (15,125), (18,132), (23,138), (28,142), (69,199), (75,208), (78,219), (79,222), (81,228), (82,232), and (96,253). The image in figure 8.2 shows the same crack as in figure 8.1, but the addition of horizontal and vertical axes against which the computed endpoints can be checked.

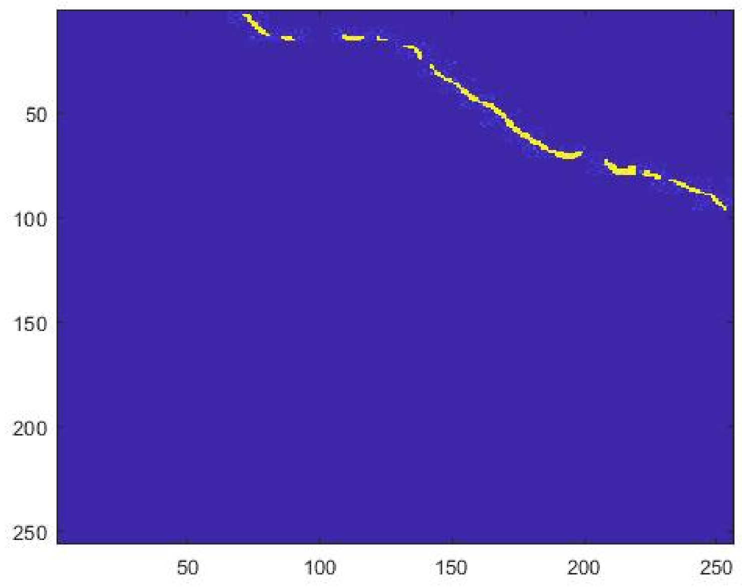

Figure 8.2: Input Image of figure 8.1 with Horizontal and Vertical Axes Added

The image in Figure 8.3 was used as input to the endpoint crack detection algorithm. The result was the set of crack endpoints (4,77), (9,82), (11,90), (11,91), (10,110), (11,116), (11,123), (13,128), (14,133), (69,199), (73,211), (75,219), (76,224), (77,230), (78,232), (85,246), (86,248), and (90,253). The image in figure 8.4 shows the same crack as in image 8.3, but with the addition of horizontal and vertical axes against which the computed endpoints can be checked.

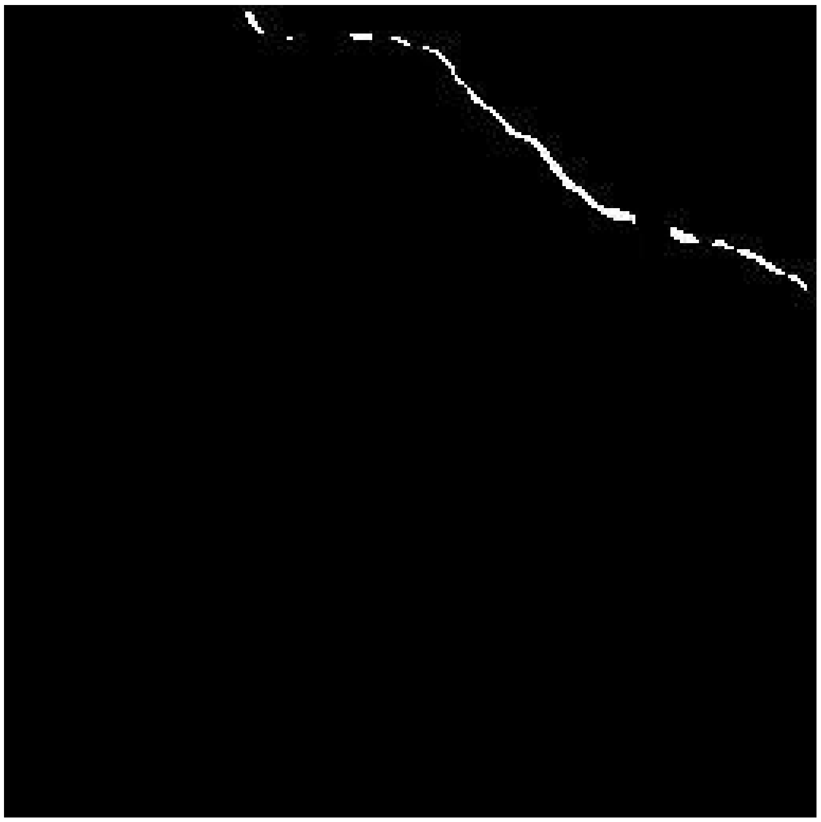

Figure 8.3: Input Image 7070-24-det-low

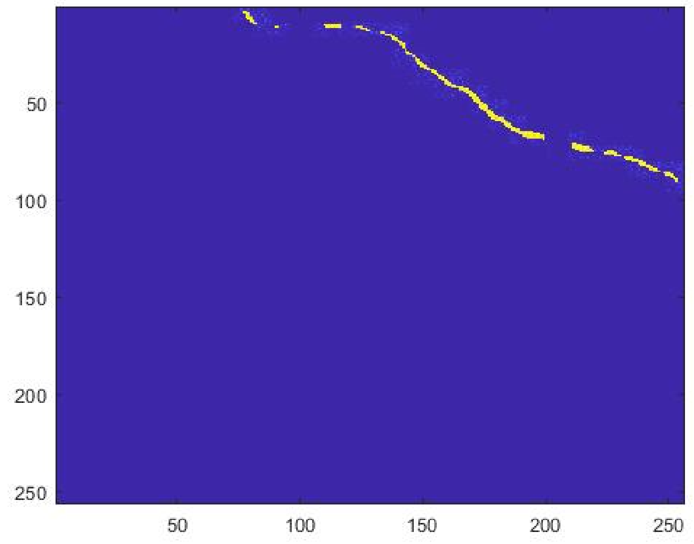

Figure 8.4: Input Image of figure 8.3 with Horizontal and Vertical Axes Added

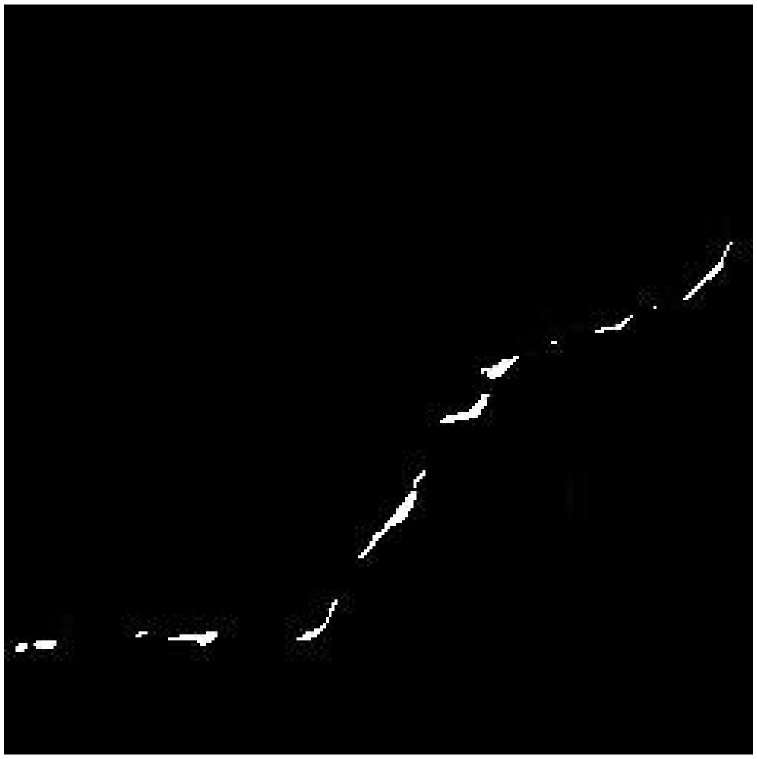

Figure 8.5: Input Image 7003-18-det-up.jpg

The image in Figure 8.5 was used as input to the endpoint crack detection algorithm. The result was the following set of crack endpoints: (82,249), (101,233) ,(104,223), (107,215), (112,203), (116,188), (116,189), (121,176), (126,164), (134,166), (143,150), (160,144), (165,141), (189,122), (204,114), (215,49), (215,73), (216,46), (217,57), (217,101), (218,18), (219,8), (219,11), and (221,4). The image in figure 8.6 is the same as the image in figure 8.5 but with horizontal and vertical axes added so that the endpoints can be checked.

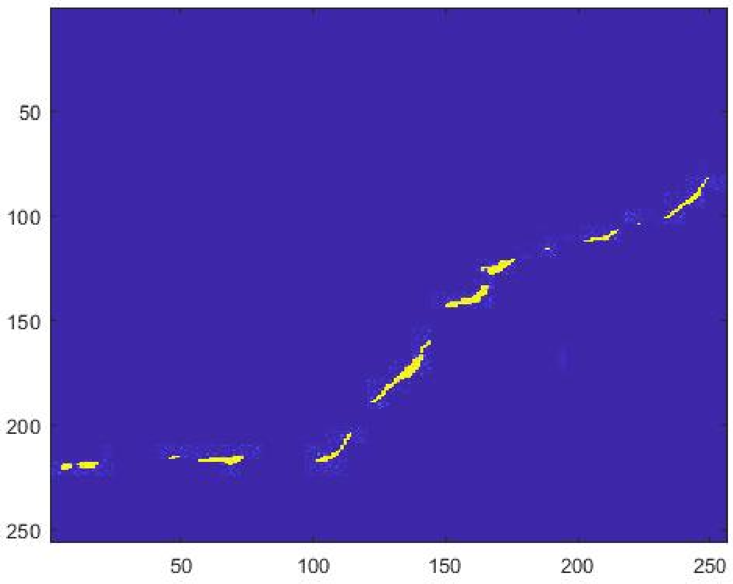

Figure 8.6: Input Image in figure 8.5 with Horizontal and Vertical Axes Added

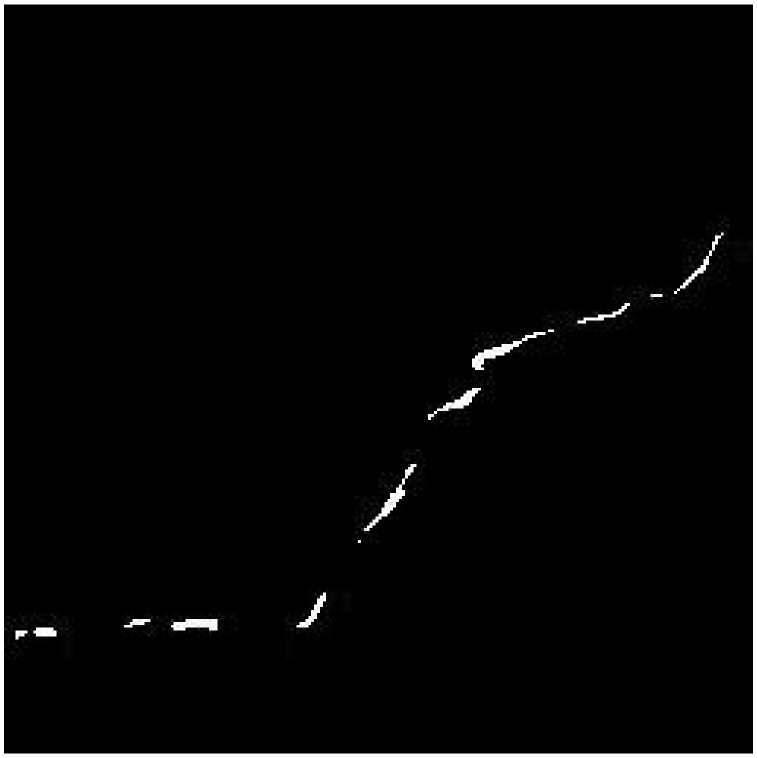

Figure 8.7: Input Image 7003-18-det-low.jpg

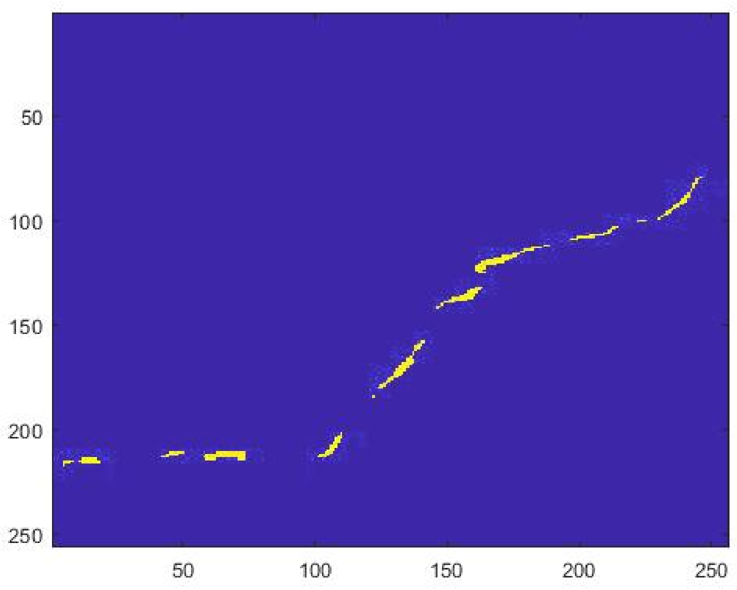

Figure 8.8: Input Image in figure 8.8 with Horizontal and Vertical Axes Added

The image in Figure 8.7 was used as input to the endpoint crack detection algorithm. The result was the following set of crack endpoints: (79,246), (99,230), (100,222), (100,225), (103,214), (109,197), (112,187), (112,188), (113,185), (124,161), (132,163), (142,146), (158,141), (180,124), (184,122), (202,110), (211,50), (211,73), (213,42), (213,58), (213,101), (215,8), (215,11), (215,18), and (217,4). The image in 8.8 is the same as the image in figure 8.7 with horizontal and vertical axes added against which the endpoints can be compared.

8.2 Detection of Crack Intersection Points

The crack intersection point algorithm reads in an input image. The algorithm uses vertical differences between two branches or parts of a crack to determine where the two parts intersect. The pixel distances were computed by determining the pixel location (i,j) of the last pixel with value 255 before the pixel in the next row (i+1,j) would have value 0, starting an index k at 0, and then determining the next pixel with value 255 at pixel location (i+1+k,j). The pixel distance is k. Horizontal differences were used where the two crack branches had more of a vertical characteristic.

When one-pixel differences are found between two parts of a crack for pixels where I(i+k,j+1)=255 and I(i+1+k,j+1)=255, and I(i+1+k,j+2) =255, then for the particular row, i, each column j+k+1 is noted for each k>= 0. Then the crack intersection points are assigned as (i,j+1+k). This worked for the crack, 7070-24-det-clean.tiff. By adding some code to check for horizontal differences between two parts of a crack for pixels where I(i-1,j+k) = 255 and I(i-1,j+1+k)=255, this worked for the crack, 7003-18-det-clean.tiff. In particular, when one-pixel differences are found between two parts of a crack. Then for the particular column, j, each row i-1-k is noted for each k>= 0. Then, the crack intersection points are assigned as (i-1-k,j).

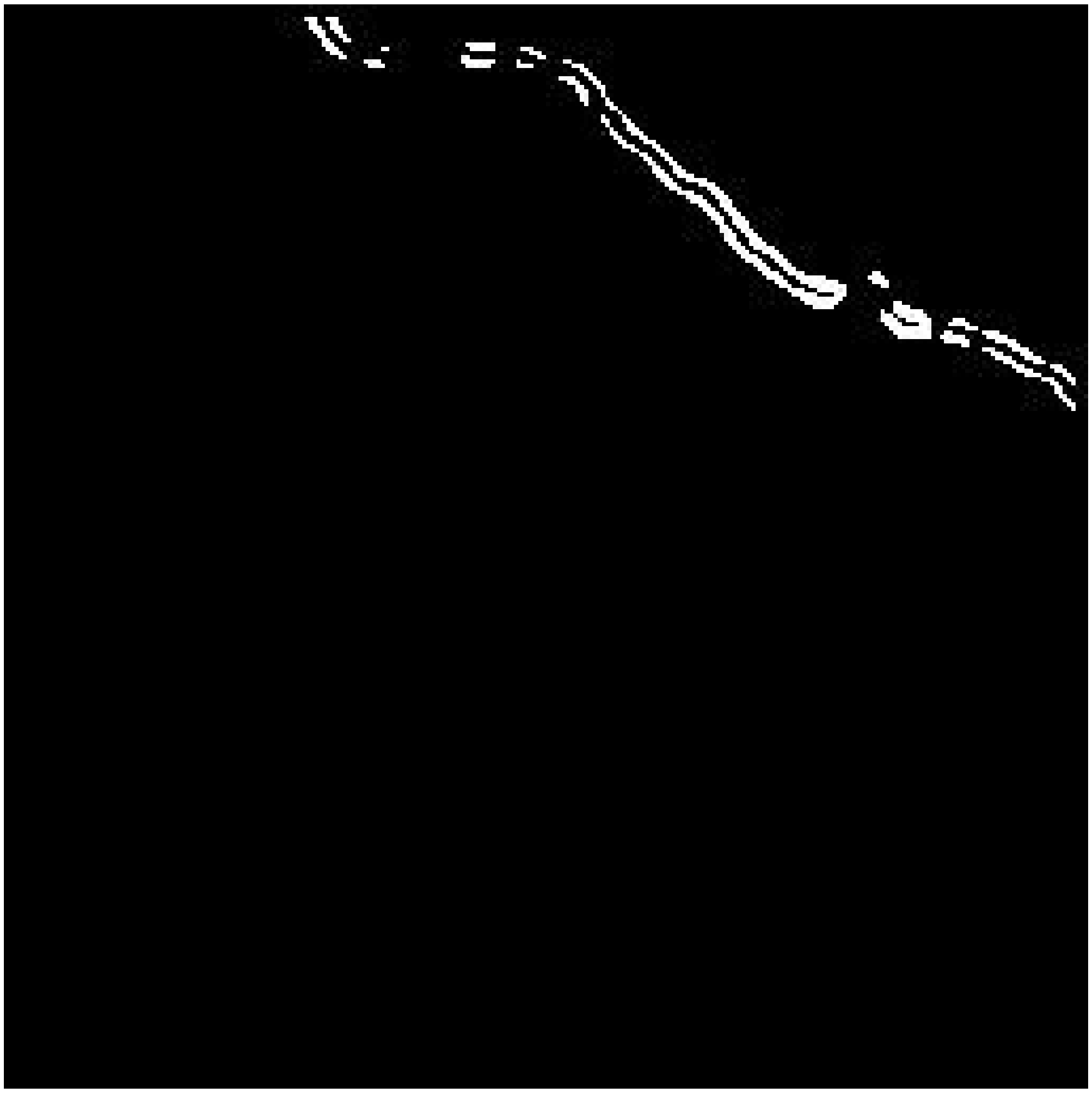

Figure 8.9: Input Image 7070-24-det-clean.jpg

The image in Figure 8.9 was used as input into the crack intersection algorithm. The output was the set of crack intersection points (68,198), (68,199), (68,200), (75,218), (75,219), and (75,220). The image in figure 8.10 is the same as the image in figure 8.9, but with horizontal and vertical axes added so that the intersection points can be checked.

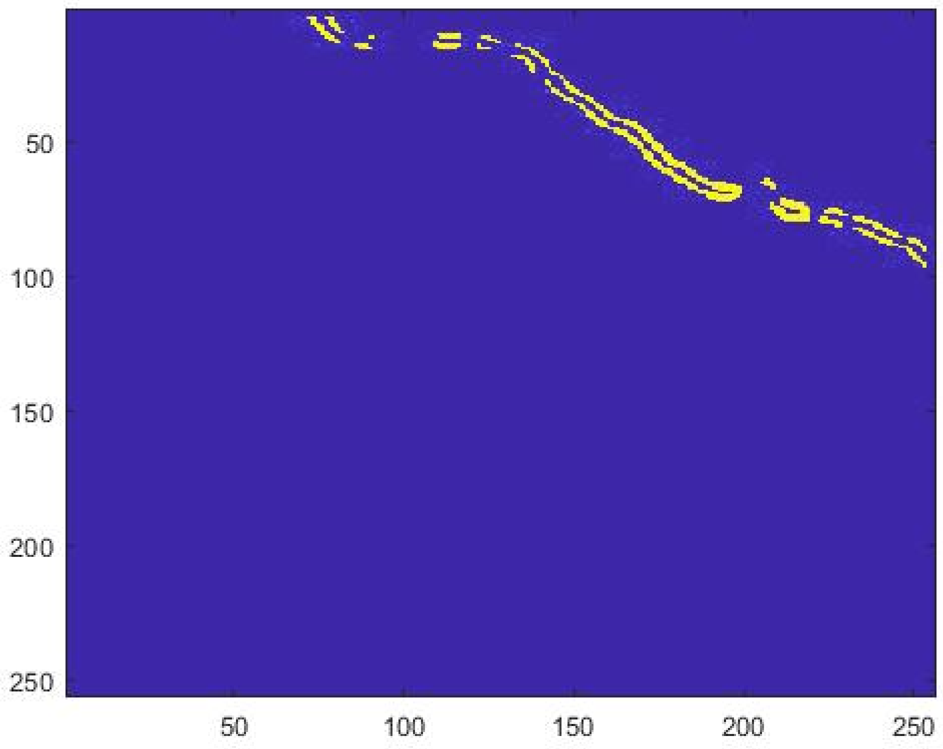

Figure 8.10: Input Image in figure 8.9 with Horizontal and Vertical Axes Added

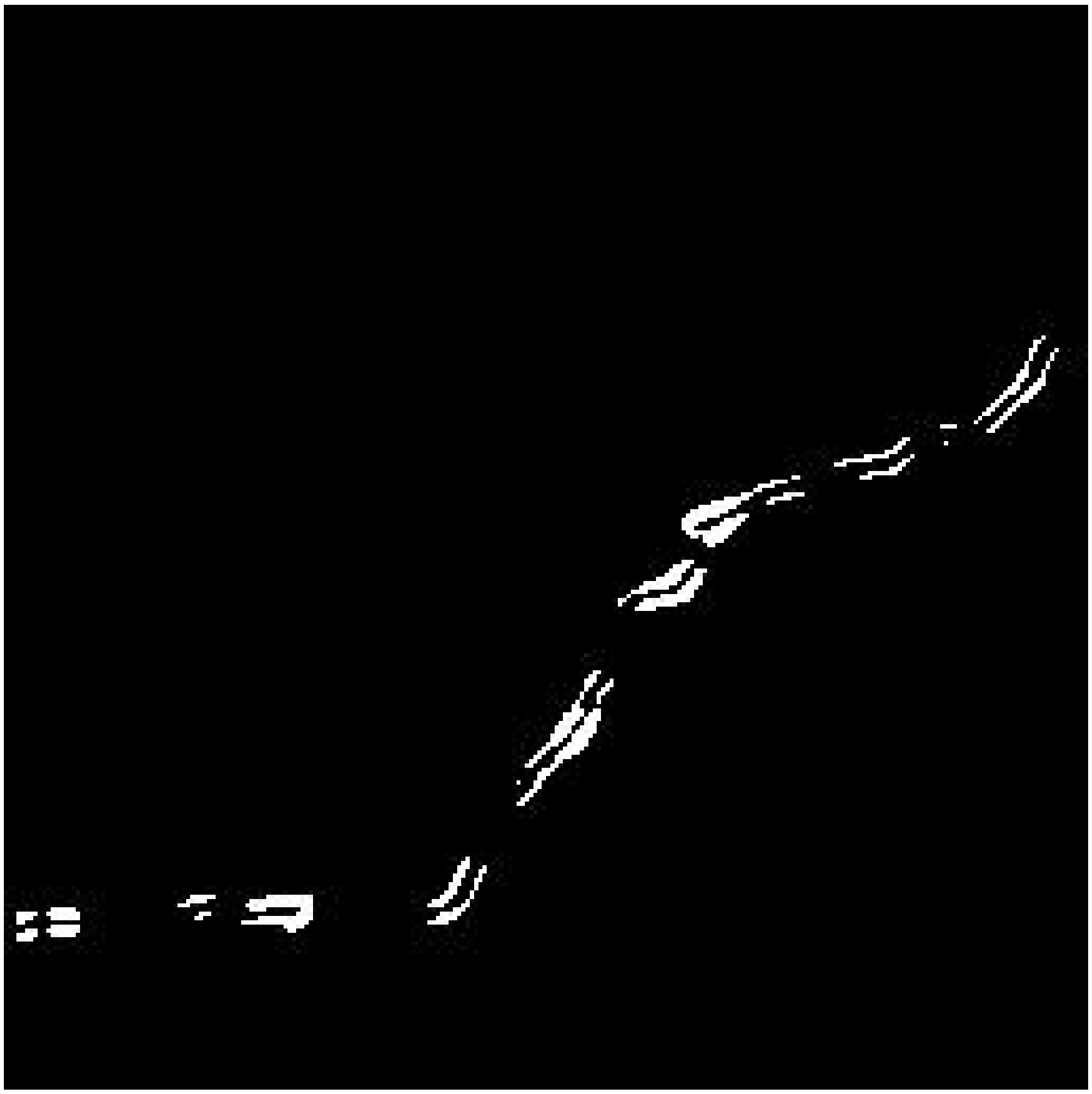

Figure 8.11: Input Image 7003-18-det-clean.jpg

The image in Figure 8.11 was used as input into the crack intersection algorithm. The output was the set of crack intersection points (124,164), (213,72), (213,73), and (213,74). The image in figure 8.12 is the same as the image in figure 8.11, but with horizontal and vertical axes added so that the intersection points can be checked.

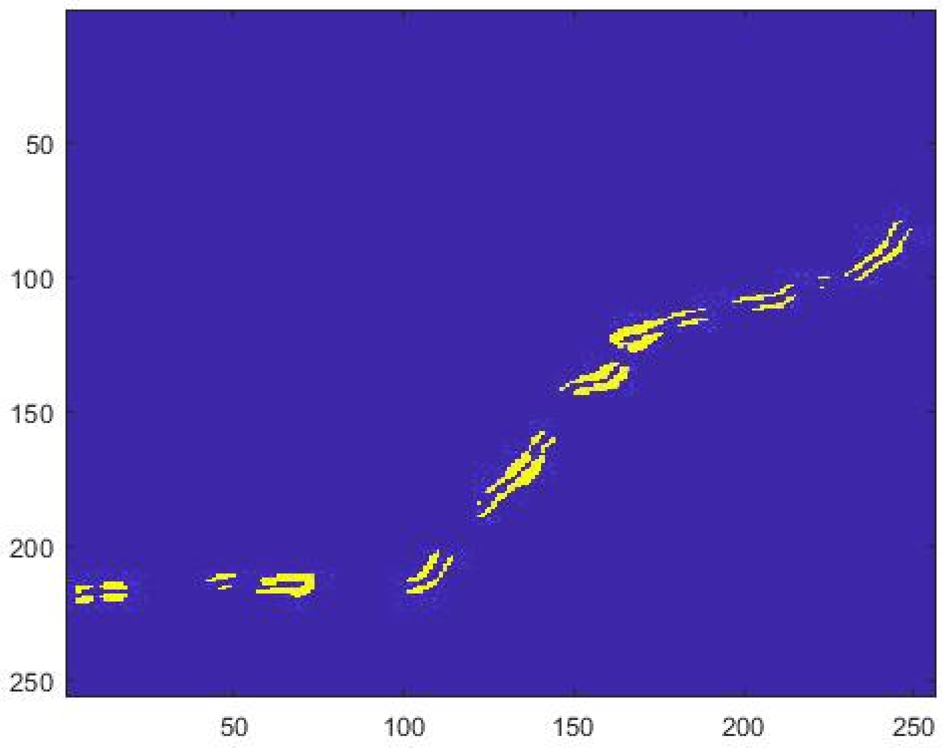

Figure 8.12: Input Image in figure 8.11 with Horizontal and Vertical Axes Added

8.3 Orientation Detection of Lines

The orientation detection algorithm computes the Radon transform (sinogram) of a image of lines at various angles. Then, the maximum value of the sinogram is computed at each angle and is plotted as a function of angle. This allows for the selection of a threshold that will separate the actual Radon transform-detected angles from the rest of the values in the sinogram, or the clutter level. The clutter in the image is caused by the number of lines at different angles being in close proximity to each other as they emerge from one point. The clutter level is evident when the histogram of the maximum values of the sinogram is plotted. The fact that the maximum sinogram value obtained at the angle of each line in the image is much greater than the level of the clutter allows for the selection of a threshold that will separate the actual Radon transform-detected angles from the clutter values in the sinogram. That threshold is chosen to be the mean of the maximum sinogram values. If the maximum value of the sinogram at a particular angle $\in$ the interval [0,180] degrees is greater than the mean of the maximum values of the sinogram and if ms(θ-1) $<$ ms(θ) and if ms(θ+1) $<$ ms(θ), where θ is the current angle and θ-1 and θ+1 denote one degree less and one degree greater than the current angle and where ms denotes the vector of maximum sinogram values, then the output matrix at location (i,j) is set to 1. Otherwise, the output matrix at location (i,j) is set to 0. Next, the values in each column are summed, and if the result is greater than 1, it is set to 1. Next, the values in each column are added up, and if the result is greater than 1, it is set to 1. If the column sum is nonzero, then the orientation count (which counts the number of angles in the sinogram) is incremented by 1. Finally, the mean and the standard deviation are computed. Now, the column index in the sinogram denotes the angle in degrees. However, since the established angles in the sinogram lie in the interval [0,179] degrees, and since vectors in MATLAB do not have an index of 0, but instead start at 1, it is necessary to subtract 1 from the column index, j.

For the three-line image shown in figure 8.13 , the plotted maximum sinogram value versus orientation is shown in figure 8.14.

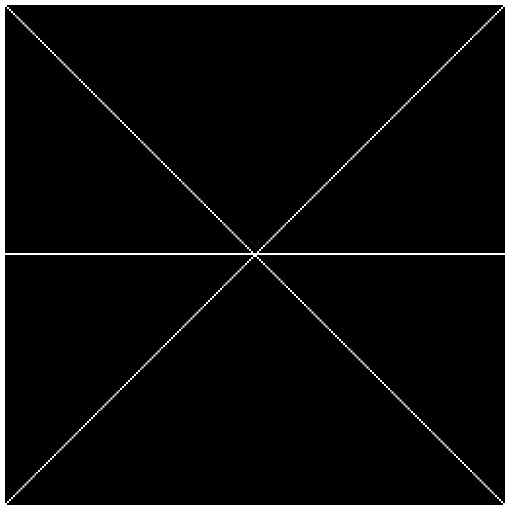

Figure 8.13: Input Image for Orientation Detection Algorithm

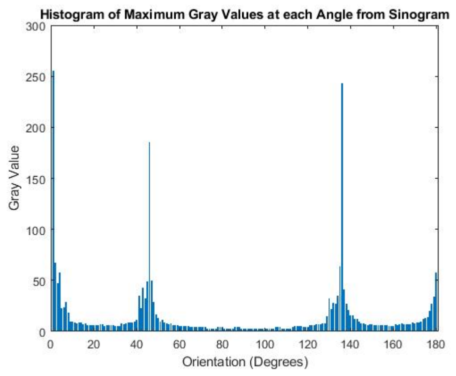

Figure 8.14: Sinogram Plot for Threshold Determination

For this three-line image, with lines at angles of 0 degrees, 45 degrees, and 135 degrees, the computed mean was 60 degrees with a standard deviation of 12.6491 degrees.

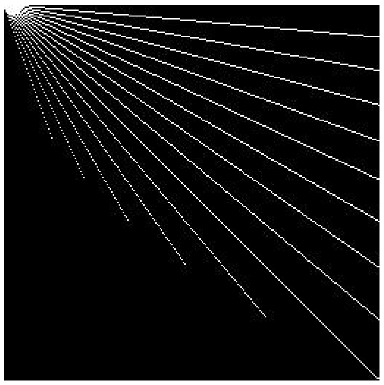

Figure 8.15: Input Image with Lines at 14 Different Angles

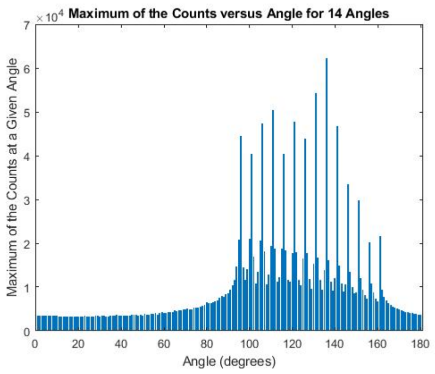

Figure 8.16: Histogram of Maximum Sinogram at each Angle for 14 Input Angles

Figure 8.15 above shows the input image with 14 lines at different angles, starting at 5 degrees and continuing in 5-degree increments up to 70 degrees. Figure 8.16 above shows a histogram of the maximum of the sinogram at each angle. What appears clearly is 14 angles associated with lines that are perpendicular to the lines at the original 14 angles in the input image.

Figure 8.17 shows the number of lines for each angle for the case of 14 lines. In this case of 14 lines, the number of computed lines at each angle is equal to the number of true lines at each angle.

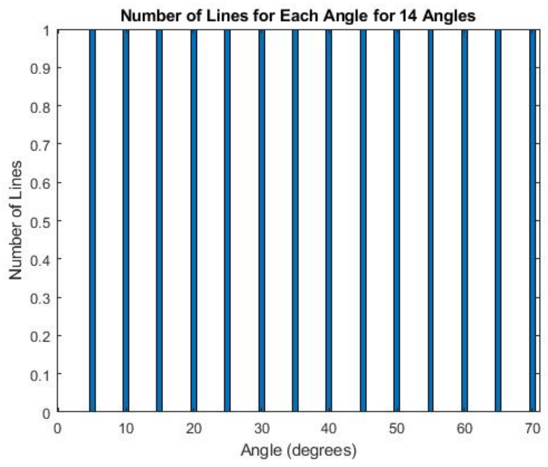

Figure 8.17: The Number of Lines per Angle for 14 Angles

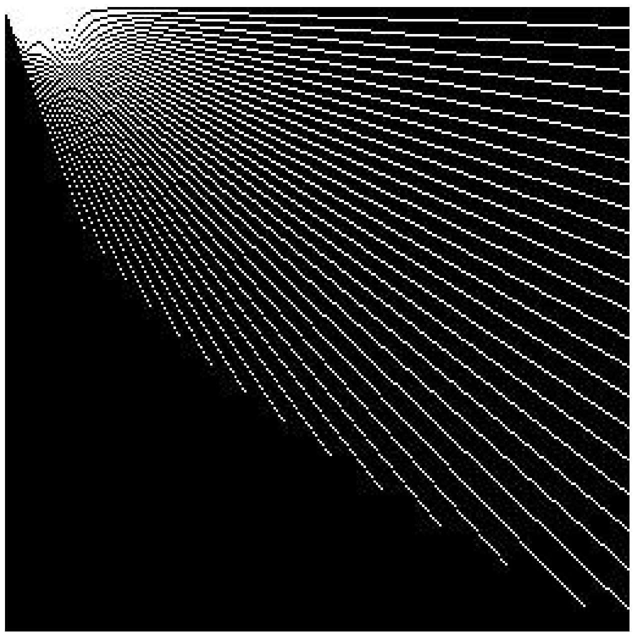

Figure 8.18: Input Image Showing Lines at 35 Different Angles

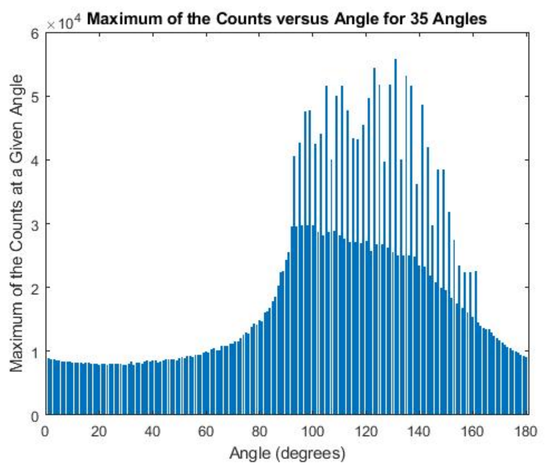

Figure 8.19: Histogram of Maximum of Sinogram at Each Angle for 35 Input Angles

Figure 8.18 above shows the input image with 35 lines at different angles (starting at 2 degrees, with 2-degree increments, up to 70 degrees). Figure 8.19 above shows a histogram of the maximum of the sinogram at each angle. 31 of 35 angles can be counted from the histogram. However, the sinogram of the image shown in figure 8.20 shows 35 peaks which correspond to 35 detected lines.

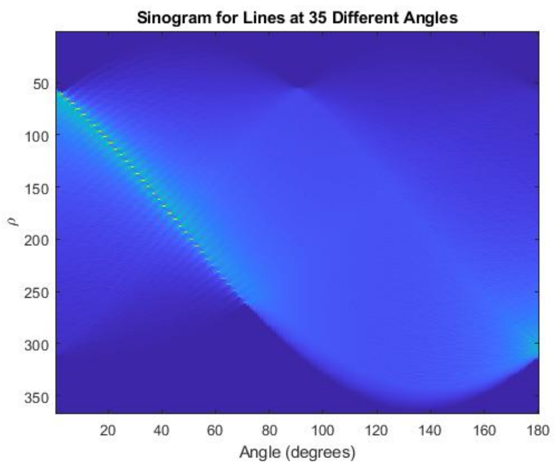

Figure 8.20: Sinogram of 35 Lines at 35 Angles Showing the Peaks of the 35 Detected Lines

Figure 8.21 shows the number of lines for each angle for 35 lines. In the case of 35 lines, the number of computed lines at each angle is equal to the true number of lines at each angle.

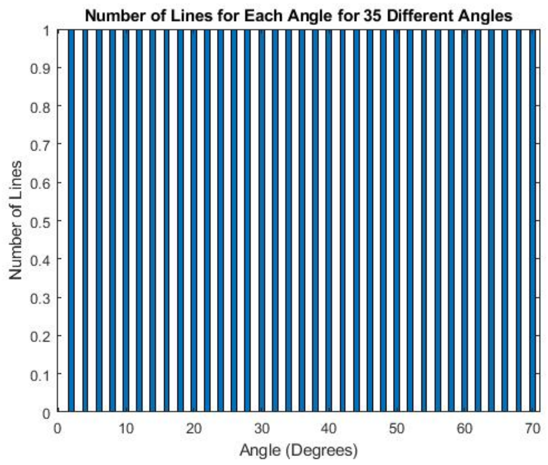

Figure 8.21: The Number of Lines per Angle for 35 Angles

The mean of the 14 detected angles was computed to be 37.5 degrees which is exactly the mean of the 14 input angles. Also, the standard deviation of those 14 detected angles is 20.1556 degrees. Similarly, the mean of the 35 detected angles was computed to be 36 degrees which is precisely the mean of the 35 input angles. Also, the standard deviation of those 35 detected angles is 20.199 degrees.

Consider the following input image in figure 8.22 of 8 lines at angles of 50, 55, 60, 65, 70, 75, 80, and 85 degrees. Figure 8.23 is the corresponding sinogram.

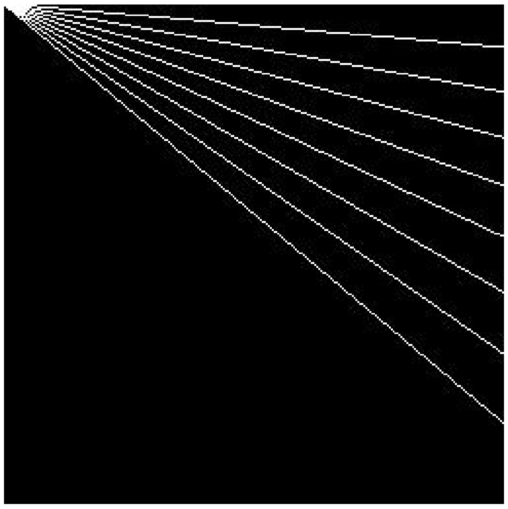

Figure 8.22: Input Image with Lines at Eight Different Angles

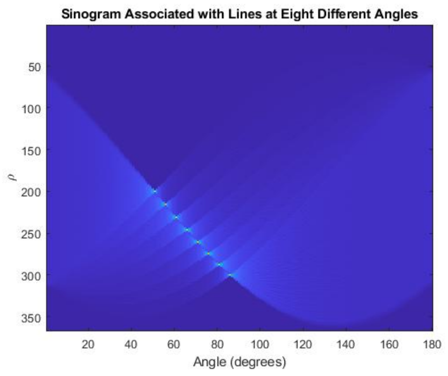

Figure 8.23: Sinogram Corresponding to Image with Lines at Eight Different Angles

Table 8.1 shows the numerically recovered angles and the true angles after applying the Radon transform to the image of eight different lines.

Numerically Recovered Angle	True Angle
50	50
55	55
60	60
65	65
70	70
75	75
80	80
85	85

Table 8.1: Numerically Recovered Angles versus True Angles

Figure 8.24 shows an image with 10 lines at an angle of 25 degrees and 5 lines at an angle of 60 degrees.

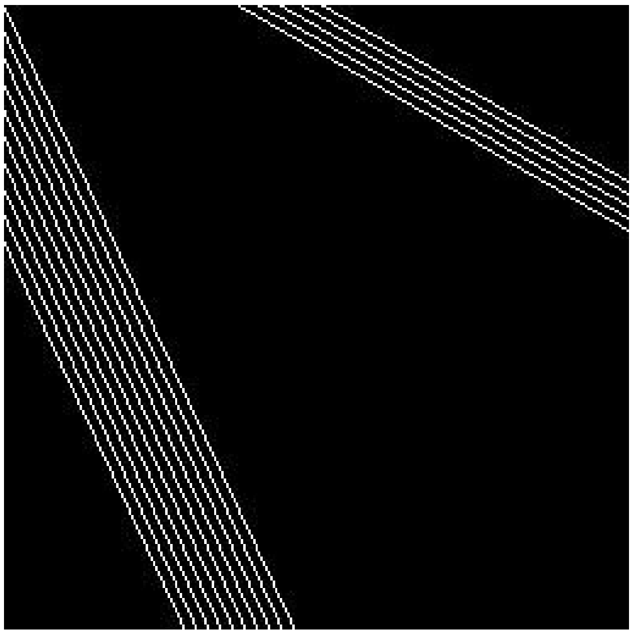

Figure 8.24: Image with 10 Lines at 25 degrees and 5 lines at 60 degrees, with angles measured with respect to the vertical

When the Radon transform processed the image in figure 8.24, it should produce peaks at the two angles, equal to the number of lines at each angle. The resulting sinogram shown in figure 8.25 shows those peaks.

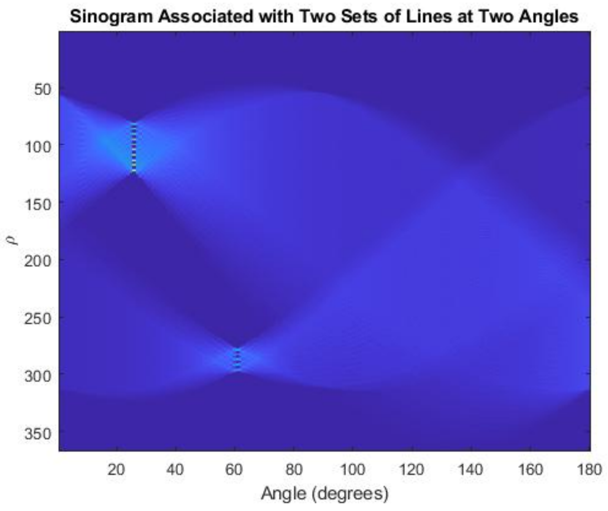

Figure 8.25: Sinogram Showing Peaks at Two Different Angles

When processed by the orientation detection algorithm, it should detect lines at those two angles, and this is what it does. Figure 8.26 shows the number of lines at each of the two angles.

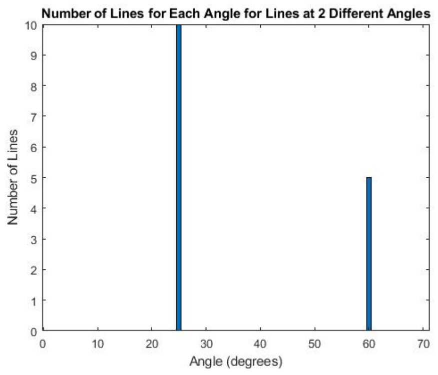

Figure 8.26: The Number of Lines at Each Angle for Two Different Angles

Figure 8.27 shows an image with 18 lines at 6 different angles. The corresponding sinogram is shown in figure 8.28

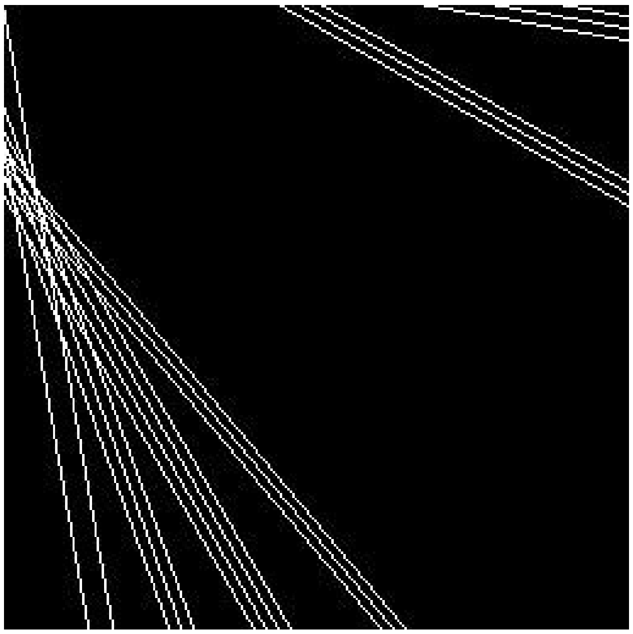

Figure 8.27: Image with 18 Lines at 6 Different Angles

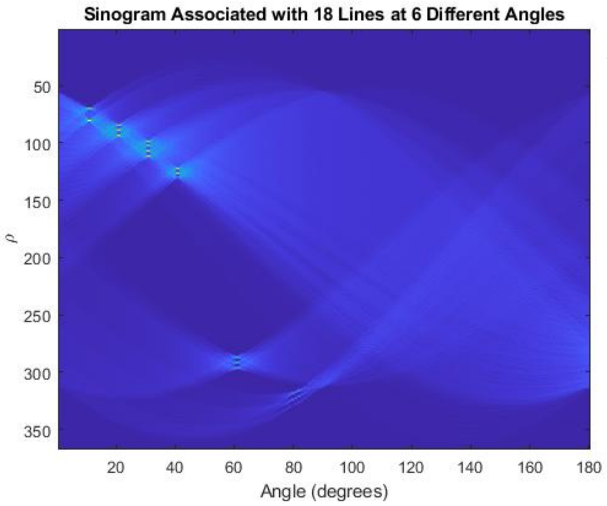

Figure 8.28: Sinogram Associated with Image of 18 Lines at 6 Different Angles

The bar plot in figure 8.29 shows the maximum of the sinogram values in figure 8.28. The bar

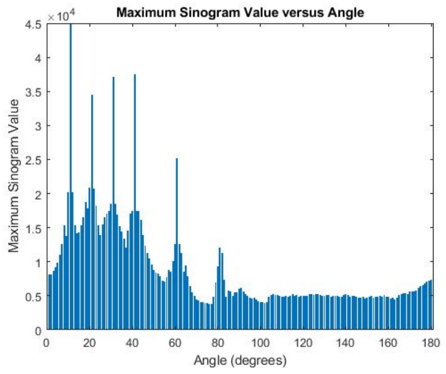

Figure 8.29: Maximum Value of the Sinogram for 18 Lines

plot in figure 8.30 shows the true number of lines at each angle in the image of 18 lines at 6 different angles. The bar plot in figure 8.31 shows the computed number of lines at each angle in the image of 18 lines at 6 different angles. The three lines at the last of the six angles were not detected, since the length of the lines relative to the size of the image was too small.

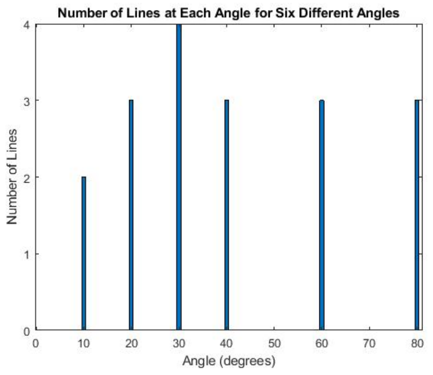

Figure 8.30: True Number of Lines at Each Angle for 18 Lines at 6 Different Angles

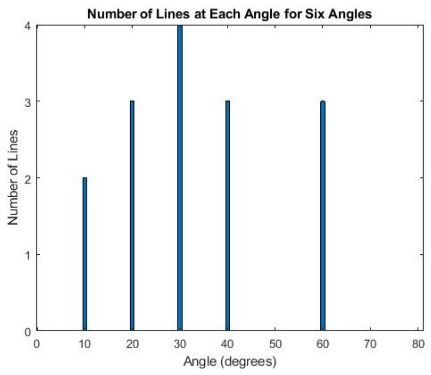

Figure 8.31: Computed Number of Lines at Each Angle for 18 Lines at 6 Different Angles

8.4 Determination of Line Thickness

The line thickness algorithm consisted of four parts, a part to detect horizontal lines and determine their thickness, a part to detect vertical lines and determine their thickness, a part to detect lines at 45 degrees and -45 degrees and determine their thickness, and parts to determine lines at other selected angles.

To detect a horizontal line, the algorithm determined that a pixel located at (i,j) was on a horizontal line if I(i,j) $\neq$ 0 and if I(i,j-1) = 0 and if I(i,j+1) $\neq$ 0 and if I(i-1,j) = 0 and if I(i,j+5) $\neq$ 0 and if I(i,j+10) $\neq$ 0, where I denotes the pixel value or intensity. Then, to determine the thickness, while I(i+k,j) $\neq$ 0, increment k (k=k+1) and thick (thick=thick+1) where k is the row increment and thick is the line thickness.

This is illustrated in figure 8.32

$$\begin{array}{ccccccccccc}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 255 & 255 & \cdot & \cdot & \cdot & 255 & \cdot & \cdot & \cdot & 255 \\
0 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
0 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
0 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
0 & 255 & 255 & \cdot & \cdot & \cdot & 255 & \cdot & \cdot & \cdot & 255 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{array}$$

Figure 8.32: Depiction of a Horizontal Line

A pixel at location (i ,j) was determined to lie on a vertical line if I(i,j) $\neq$ 0 and if I(i-1,j) = 0 and if I(i,j-1) = 0 and if I(i+1,j) $\neq$ 0 and if I(i+5,j) $\neq$ 0 and if I(i+10,j) $\neq$ 0. Then, to determine the line thickness, while I(i,j+k) $\neq$ 0, increment k (k=k+1) and thick (thick=thick+1).

This is depicted in figure 8.33

0	0	0	0	0	0	0
0	255	·	·	·	255	0
0	255	·	·	·	255	0
0	·	·	·	·	·	·
0	·	·	·	·	·	·
0	·	·	·	·	·	·
0	255	·	·	·	255	0
0	·	·	·	·	·	·
0	·	·	·	·	·	·
0	·	·	·	·	·	·
0	255	·	·	·	255	0

Figure 8.33: Depiction of a Vertical Line

A pixel at location (i,j) is determined to lie on a line (which is increasing with j) at an angle of multiple pixel thickness if I(i,j) $\neq$ 0 and if I(i,j-1) = 0 and if I(i,j+1) = 0 and if I(i+1,j) $\neq$ 0 and if I(i+1,j+1) $\neq$ 0 and if I(i+1,j+2) = 0. To determine the line thickness, while I(i,j+k) $\neq$ 0, then increment k (k=k+1) and thick (thick=thick+1).

A pixel at location (i,j) is determined to lie on a line (which is decreasing with j) at an angle of multiple pixel thickness if I(i,J) $\neq$ 0 and if I(i,j-1) = 0 and if I(i,j+1) = 0 and if I(i+1,j-1) $\neq$ 0 and if I(i+1,j) $\neq$ 0 and if I(i+1,j+1) = 0. To determine the line thickness, while I(i+k,j) $\neq$ 0, then increment k (k=k+1) and thick (thick=thick+1).

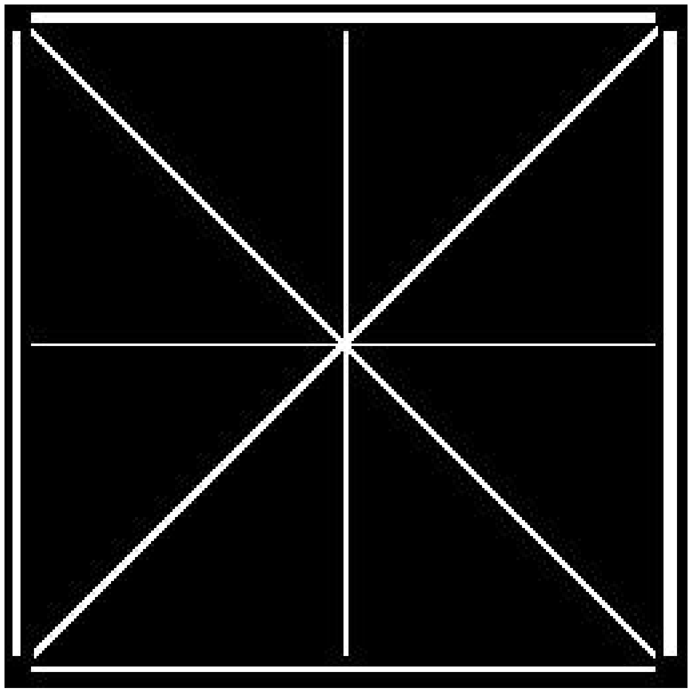

Figure 8.34: Input Image with Eight Lines of Different Thicknesses

The input image in figure 8.34 shows eight lines of varying thicknesses. The far-right vertical line has a thickness of 5 pixels. The vertical line in the middle of the image has a thickness of 2 pixels. The far-left vertical line has a thickness of 3 pixels. The top horizontal line has a thickness of 4 pixels. The horizontal line in the middle of the image has a thickness of 1 pixel. The bottom horizontal line has a thickness of 2 pixels. The diagonal line that is decreasing from left to right has a thickness of 3 pixels. The diagonal line that is increasing from left to right has a thickness of 4 pixels. The sum of all of the pixel widths is equal to 24. The number of lines is 8. Therefore, the average pixel thickness is 3 with a standard deviation of 1.2247. This is exactly what the MATLAB code determined.

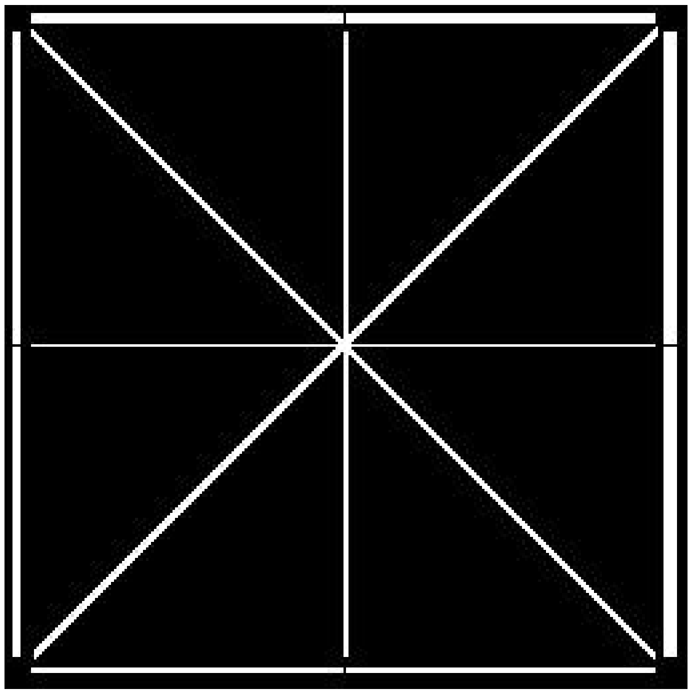

Figure 8.35: Input Image with Twelve Lines of Different Thicknesses

The input image in figure 8.35 shows twelve lines of different thicknesses. The two far-right vertical lines have a thickness of 5 pixels. The two vertical lines in the middle of the image have a thickness of 2 pixels. The vertical lines on the far-left have a thickness of 3 pixels. The horizontal lines at the top have a thickness of 4 pixels. The horizontal line in the middle of the figure has a thickness of 1 pixel. The horizontal lines at the bottom of the figure have a thickness of 2 pixels. The diagonal line decreasing from left to right has a thickness of 3 pixels. The diagonal line increasing from left to right has a thickness of 4 pixels. The sum total of the line thicknesses is 38. Therefore, since the number of lines is 12, the average line thickness is 3.1667, and the standard deviation is 1.21335. This is exactly what the MATLAB code determines.

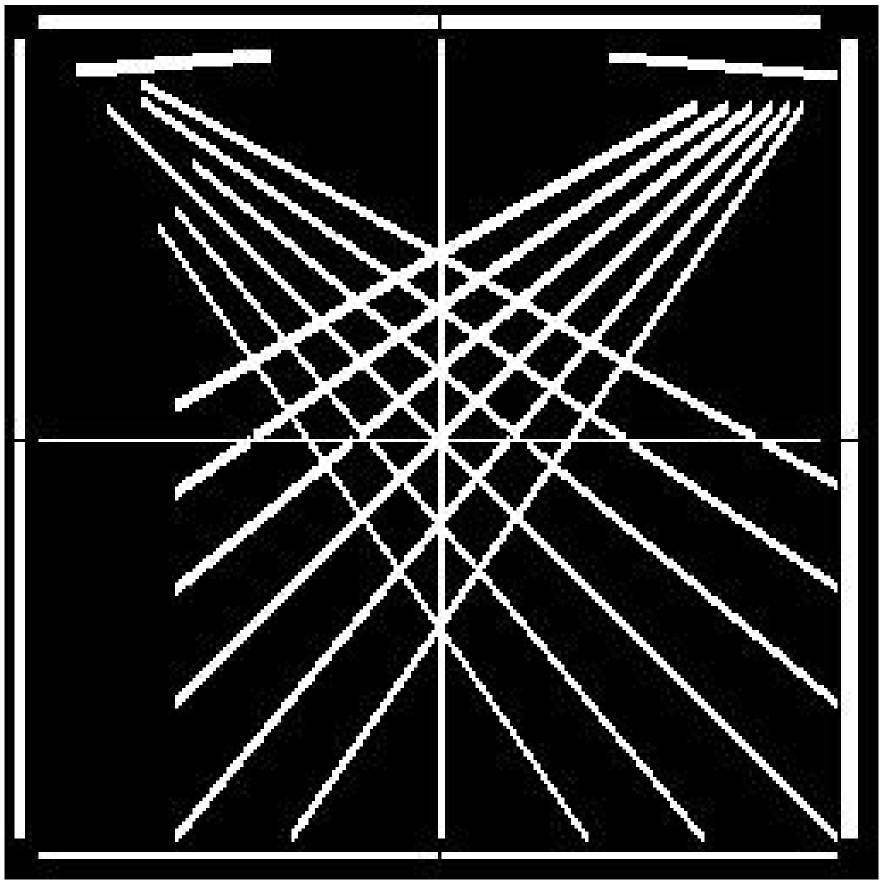

Figure 8.36: Input Image with 24 Lines of Different Thicknesses

The input image in figure 8.36 shows 24 lines of different thicknesses used as input into the line width algorithm. In addition to the horizontal and vertical lines, there are lines at angles of 5, 35, 40, 45, 50, and 55 degrees. Each line decreasing from left to right had a width of 3 pixels, and each line increasing from left to right had a width of 4 pixels. The algorithm determined that the average thickness is 3.333 pixels and that the standard deviation is 0.942809 pixels.

Figure 8.37shows an input image with 7 parallel lines with 4 different thicknesses. Figure 8.38 shows its corresponding sinogram.

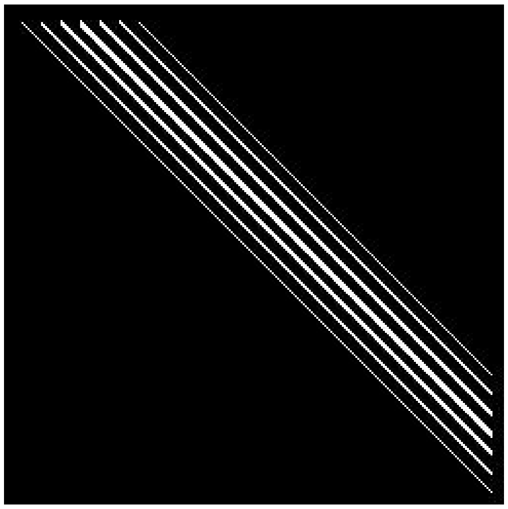
Figure 8.37: Input Image with Seven Parallel Lines With Four Different Thicknesses

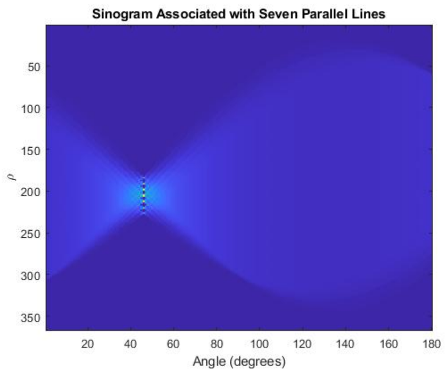

Figure 8.38: Sinogram Corresponding to the Input Image with Seven Parallel Lines with Four Different Thicknesses

Line Number	True Thickness	Detected Thickness
1	1	1
2	2	2
3	3	3
4	4	4
5	3	3
6	2	2
7	1	1

Table 8.2: True versus Detected Thickness for 7 parallel lines with 4 Thicknesses

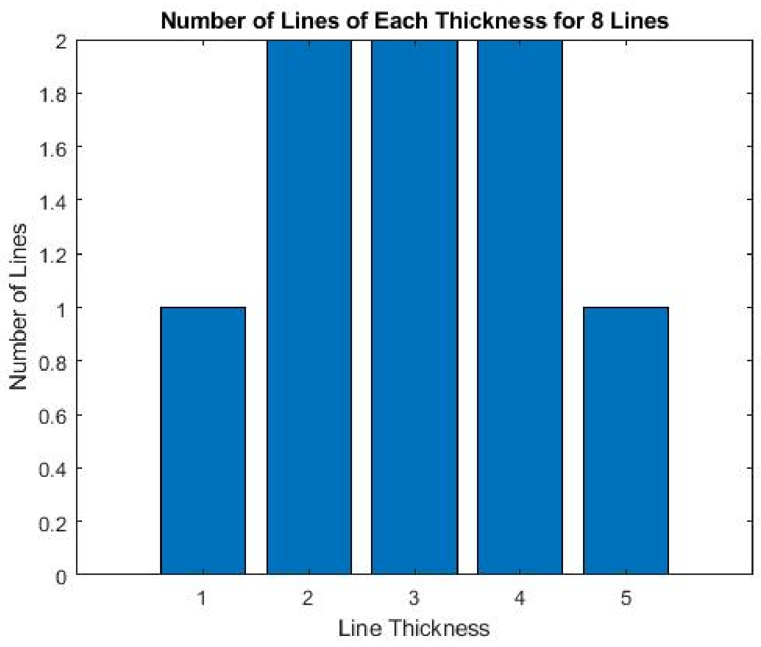

Figure 8.39: The Number of Lines at Each Thickness for 8 Lines

Figure 8.39 shows the number of lines at each thickness when an image with 8 lines was used. In the case of 8 lines, the number of computed lines for each thickness is equal to the number of true lines for each thickness.

Figure 8.40 shows the number of lines at each thickness when an image with 12 lines was used. In the case of 12 lines, the number of computed lines at each thickness is equal to the number of true lines at each thickness.

Figure 8.41 shows the number of lines at each thickness when an image with 18 lines was used. In the case of 18 lines, the number of computed lines at each thickness is equal to the true number of lines at each thickness.

Figure 8.42 shows the number of lines at each thickness when an image with 24 lines was used. In the case of 24 lines, the number of computed lines at each thickness is equal to the true number of lines at each thickness.

The algorithm designed for 24 lines at at different angles was successfully tested on 8, 12, 13, 15,

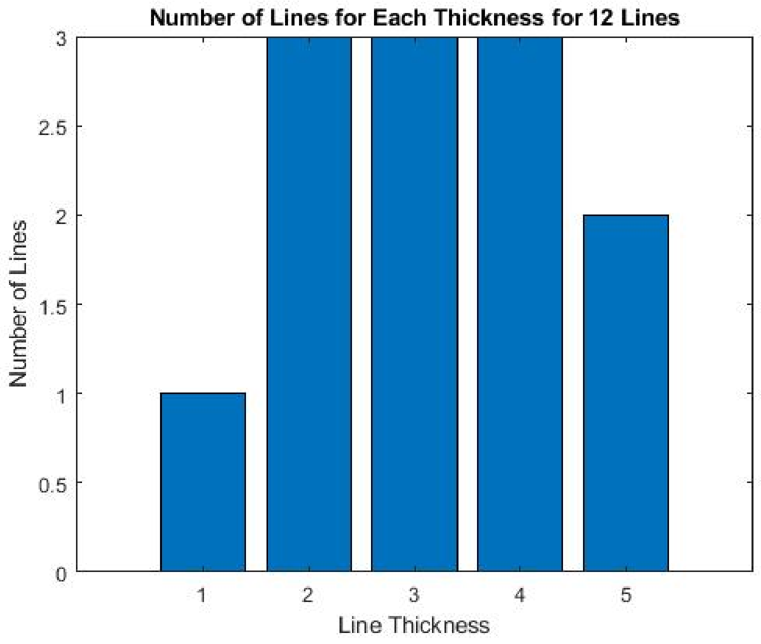

Figure 8.40: The Number of Lines at Each Thickness for 12 Lines

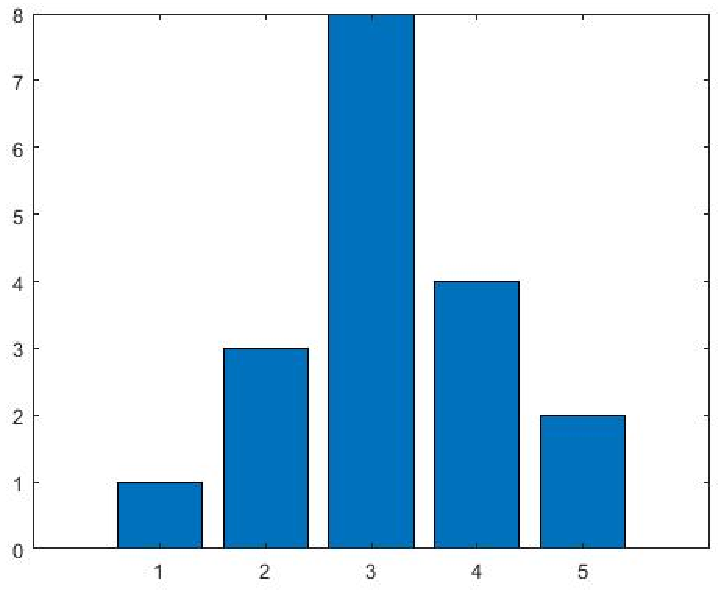

Figure 8.41: The Number of Lines at Each Thickness for 18 Lines

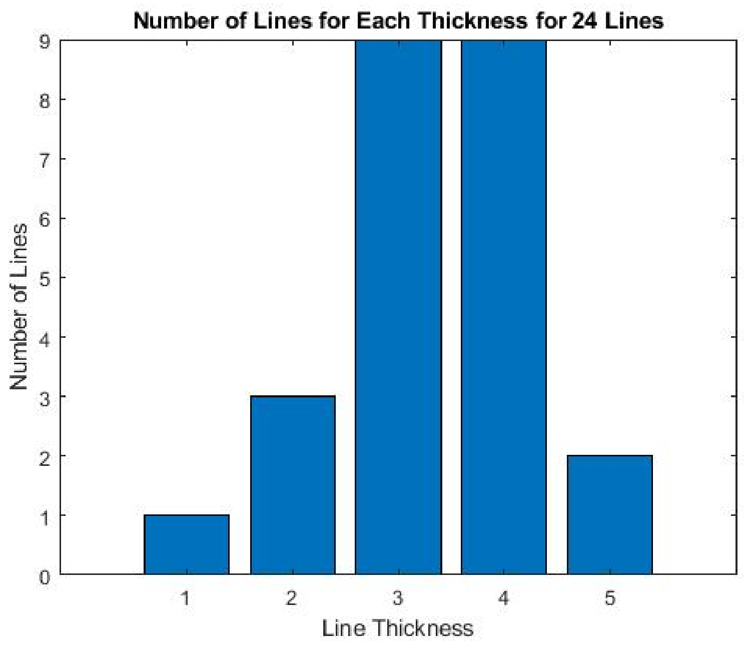

Figure 8.42: The Number of Lines at Each Thickness for 24 Lines

16, 18, 18, 20, 21, 22, 23, and 24 lines respectively. There were other test cases with 23 and 8 lines on which the algorithm did not successfully detect all of the lines. Another image with 8 lines in which only 7 were detected had the end of one line touching another line. Since the algorithm detects the endpoints and uses those to determine a line has been detected as well as its thickness, the ends of lines need to be free and not touching other lines. One image with 23 lines had 25 lines detected. This may be due to the fact that lines were introduced at angles not programmed into the algorithm.

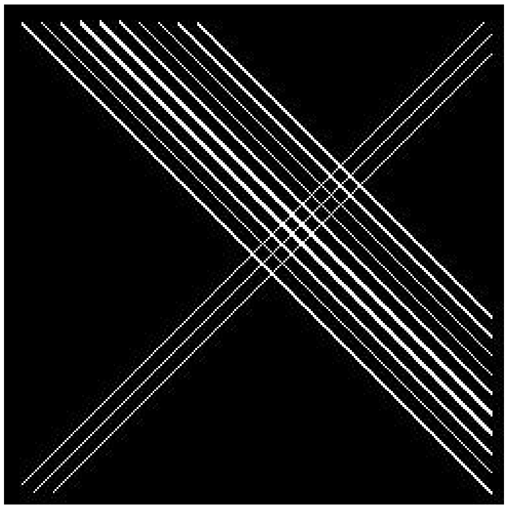

Figure 8.43: Input Image with Two Sets of Parallel Lines, 13 Lines Total

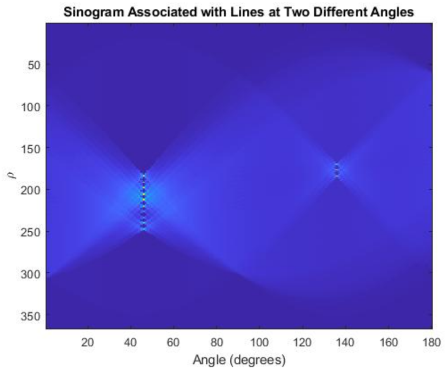

Figure 8.44: Sinogram Corresponding to the Input Image with Two Sets of Parallel Lines, 13 Total Lines

Figure 8.43 shows an input image with 13 lines, 10 parallel lines of different thicknesses at 45 degrees and 3 parallel lines at -45 degrees.

Figure 8.44 shows the sinogram corresponding to the image in figure 8.43.

Figure 8.45 shows a bar plot of the true line thickness distribution corresponding to the input image in figure 8.43. Figure 8.46 shows a

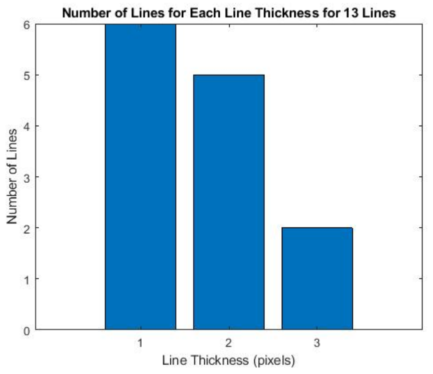

Figure 8.45: Bar Plot of the True Line Width Distribution for Parallel Lines at 2 Different Angles

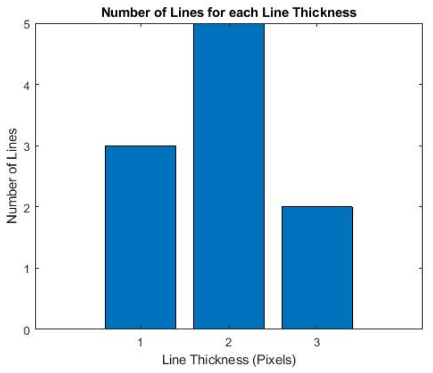

Figure 8.46: Bar Plot of the Computed Line Width Distribution for Parallel Lines at 2 Different Angles

bar plot of the computed line thickness distribution corresponding to the input image in figure 8.43. The difference in the true andn computed bar plots is due to the fact that the three lines with single pixel thickness were not detected by the line width algorithm.

8.5 Filtering of Reconstructed Images

I had previously looked at using a Kuwahara filter to process a reconstructed image, that is an image which had been transformed with the Radon transform into a sinogram and then reconstructed using filtered back projection. Since the adaptive Kuwahara filter starts with four processing windows and increases the size of those windows depending upon whether or not the variance of the newly sized window is less than the variance of the previously sized window, and uses the mean of the window with the smallest variance, I had proposed using instead the interquartile range (IQR) and median of the processing windows. However, this combination failed to produce any output. I then tried other combinations of IQR/variance and median/mean. I was able obtain output for the combination of IQR and mean. In addition, I was able to generate an adaptive (in the sense of a changing window size) version for IQR and mean.

In his paper, [47], Bartyzel describes how to construct an adaptive Kuwahara filter so that the window size changes automatically based upon the satisfaction of certain criteria. In this work, his method is used with IQR replacing variance. First, the filter window or mask is divided into four subregions as discussed previously. These subregions will initially consist of four pixels (2x2 subregions) Next,for each subregion, the mean and IQR are computed. For a given subregion, the size of the subregion window is increased by 1 (i.e. 2x2 becomes 3x3) Next, for the new window size, the mean and IQR are calculated. If the IQR of the newly sized subregion is smaller than the IQR of the window before the resizing of the window (previous subregion size), then the newly calculated values are assigned as the new subregion values. Then, the size of the window for the subregion continues to be increased until its size reaches the maximum allowable size or until the IQR of the newly enlarged subregion is greater than that calculated for the previous size of the subregion. The minimum IQR and corresponding mean will be attained The computations in this step are repeated for each of the subregions. Finally, the IQRs of each of the four subregions are compared. The subregion with the lowest IQR is sought. The value of the current pixel is then replaced with the mean of the subregion with the lowest IQR. Figure 8.47 shows an example of increasing the size of the upper left subregion. Figure 8.48 shows an example of changing (or leaving

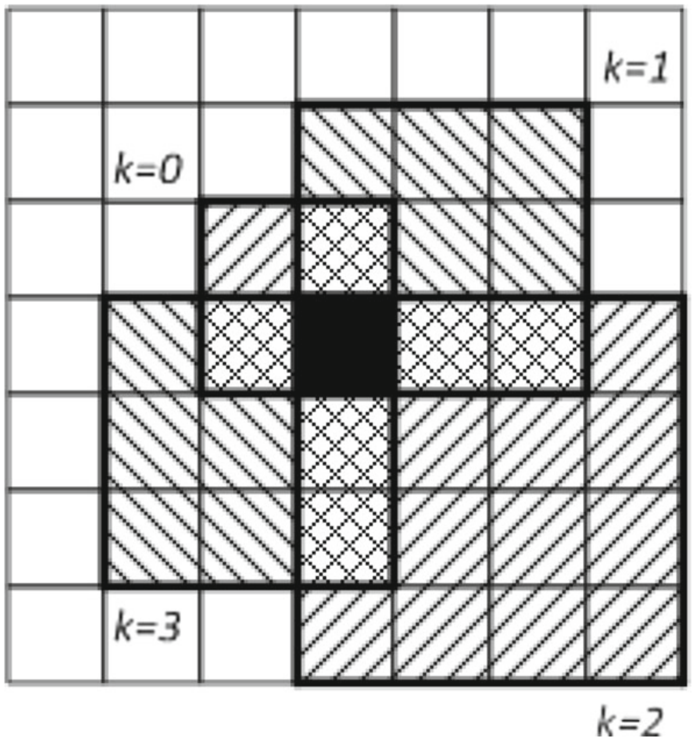

Figure 8.48: Example of Changing Windows in All Four Subregions

it as it is) the size of all four subregions.

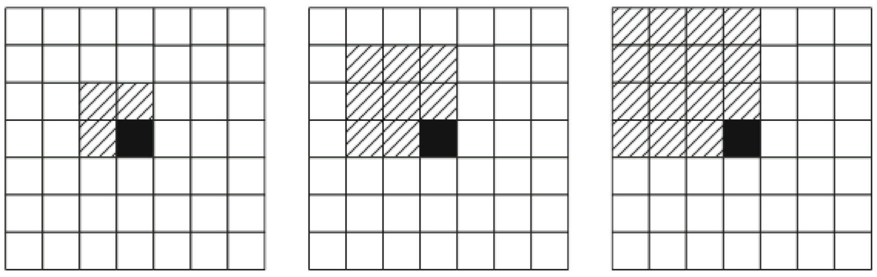

Figure 8.47: Example of Changing Window Size in a Subregion

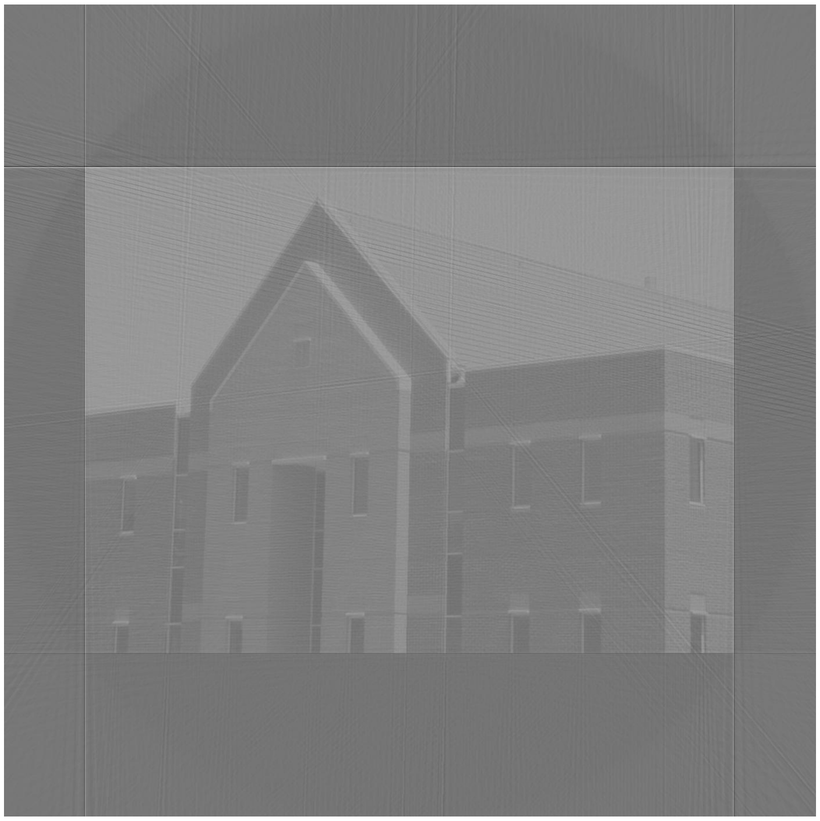

Figure 8.49: Reconstructed Image Using Filtered Back Projection with a Ram-Lak Filter

Consider the input image in figure 8.49. The image was produced by using filtered back projection with the Ram-Lak filter on a Radon-transformed image (sinogram). This image was filtered using a 9x9 Tomita-Tsuji filter with the window variance replaced by the interquartile range (IQR) yielding the image shown in figure 8.50.

Finally, an adaptive version (automatically changing the processing window size) of the Kuwahara filter was applied to the RL reconstructed image using the IQR instead of the window variance, producing the image shown in figure 8.51.

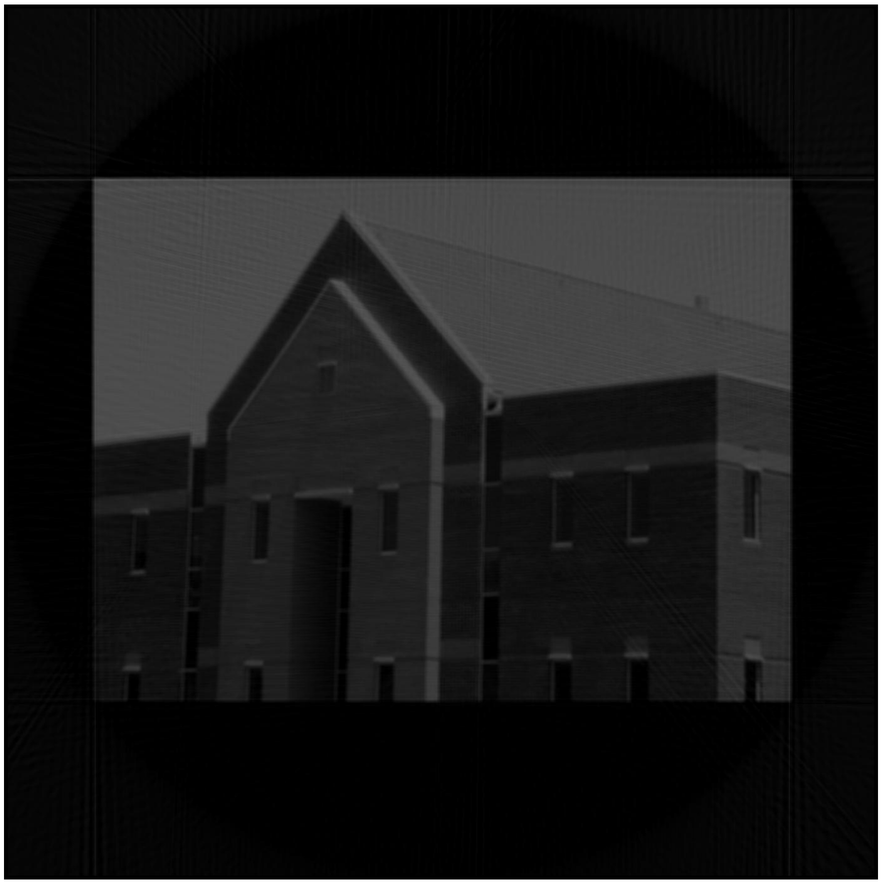

Figure 8.50: Filtered RL Reconstructed Image Using 9x9 Tomita-Tsuji Filter with IQR and Mean

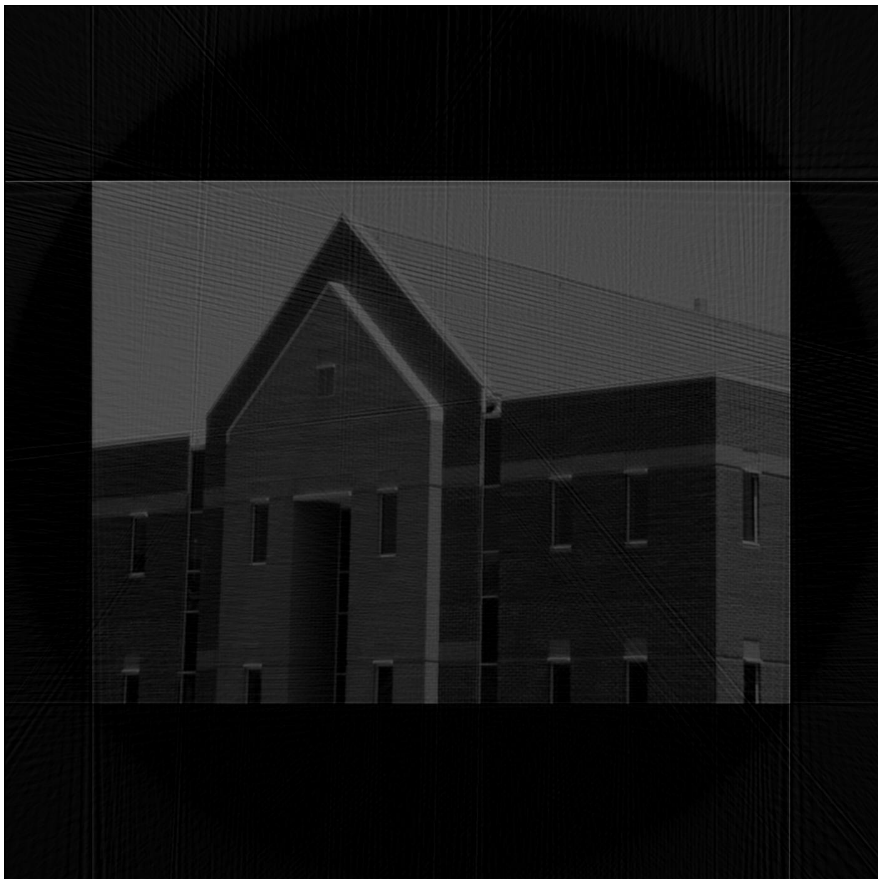

Figure 8.51: Adaptive Kuwahara Applied to RL Reconstructed Image Using IQR and Mean

Consider the image in figure 8.52 reconstructed using filtered back projection and a Shepp- Logan filter.

Figure 8.53 shows the result of applying the 9x9 Tomita-Tsuji filter with IQR replacing the window variance to the SL-reconstructed image in figure 8.52.

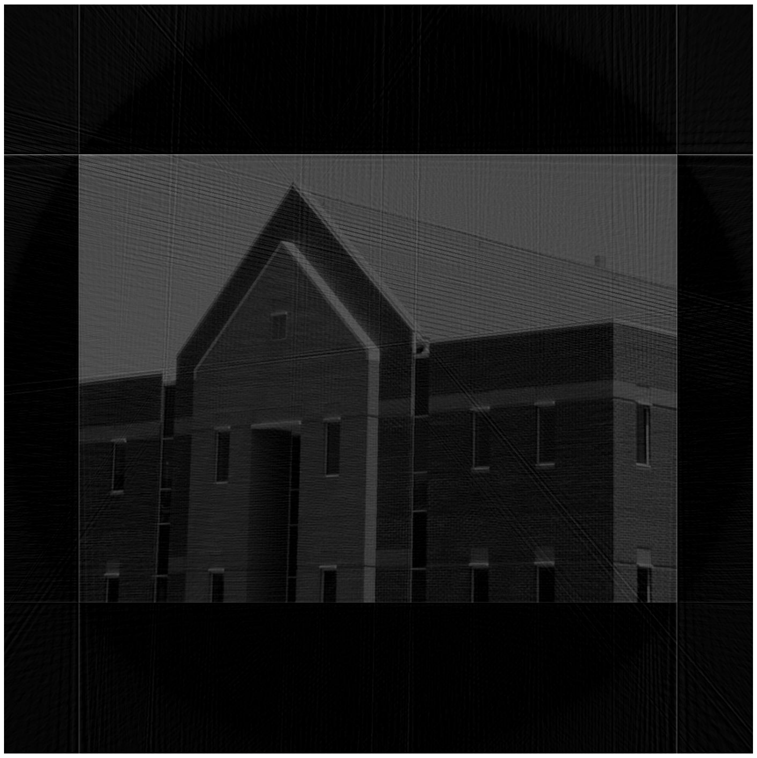

Figure 8.52: Reconstructed Image Using Filtered Back Projection with a Shepp-Logan Filter

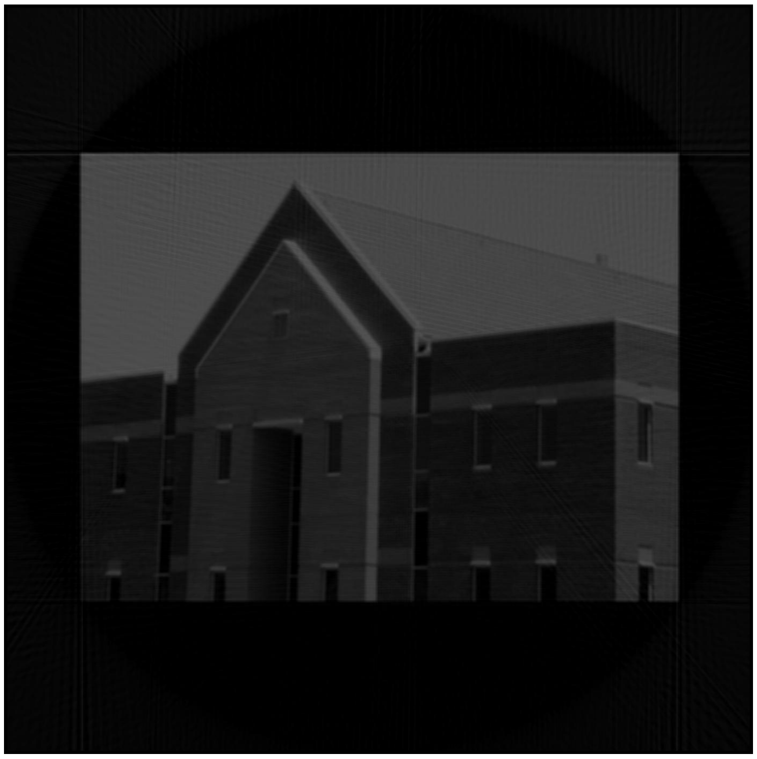

Figure 8.53: Filtered SL Reconstructed Image Using 9x9 Tomita-Tsuji Filter with IQR and Mean

Finally, figure 8.54 shows the result of applying an adaptive version of Kuwahara filter to an SL reconstructed image in figure 8.52 using the IQR instead of the window variance.

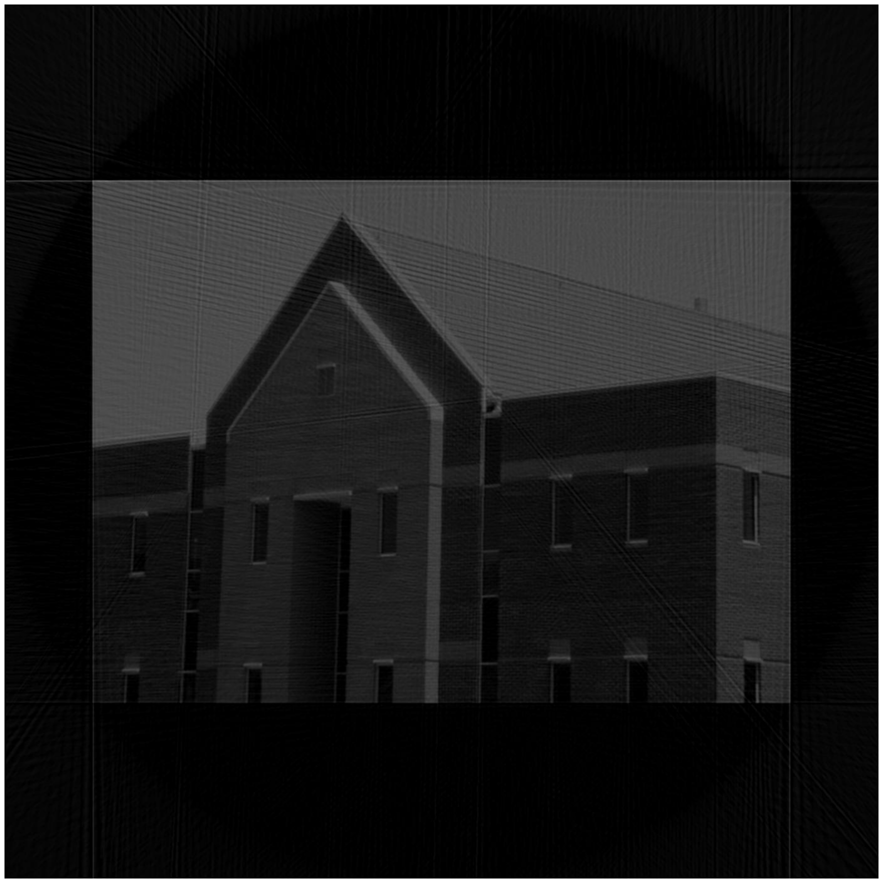

Figure 8.54: Adaptive Kuwahara Filter Applied to SL Reconstructed Image Using IQR and Mean

The RL reconstructed image of the moon in figure 8.55 was processed with a 9x9 Tomita-Tsuji filter with the window variance replaced by the IQR resulting in the image shown in figure 8.56

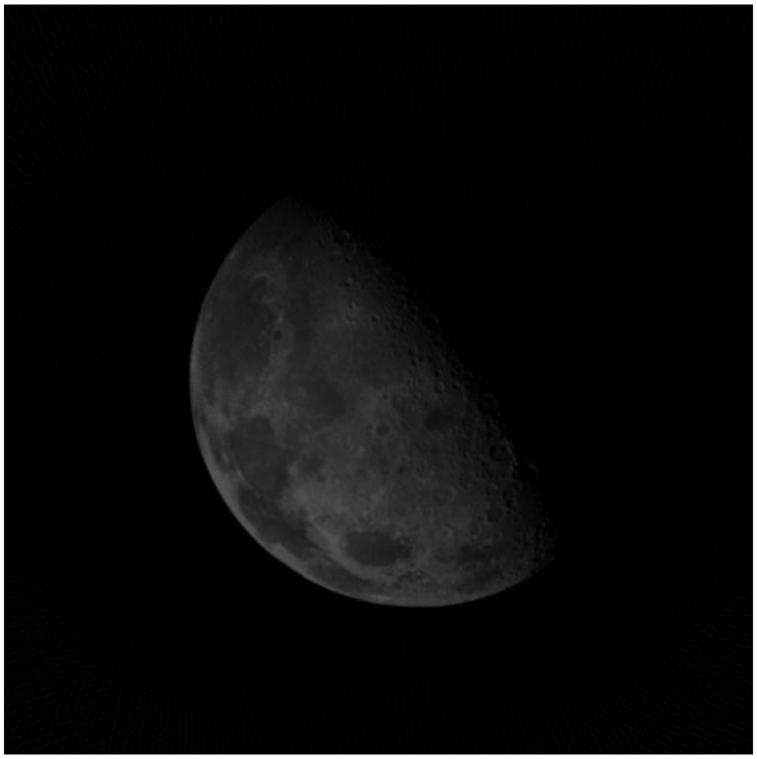

Figure 8.55: Reconstructed Image Using Filtered Back Projection with Ram-Lak Filter

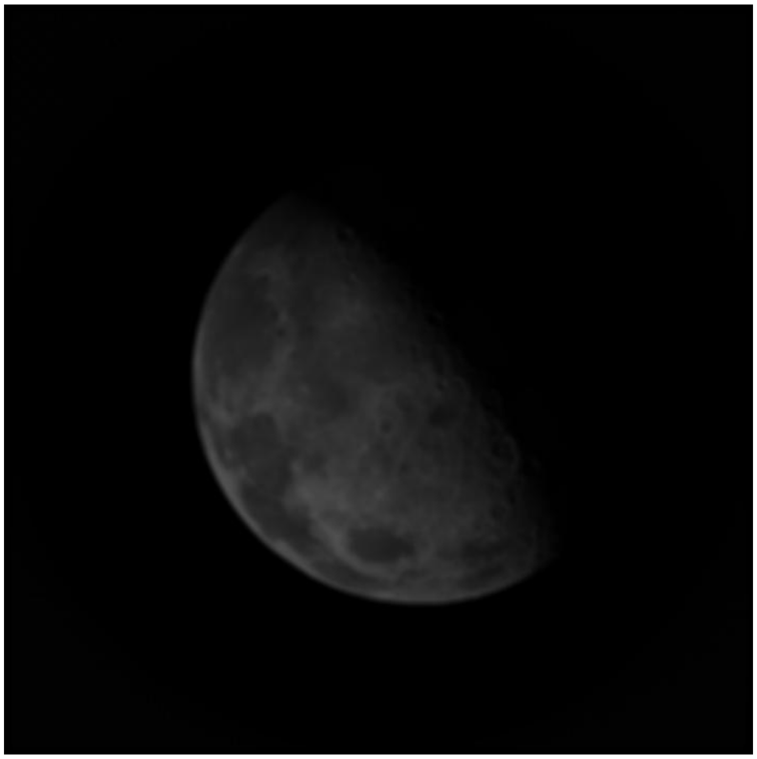

Figure 8.56: Filtered RL Resconstructed Image Using 9x9 Tomita-Tsuji Filter with IQR

Finally, the image in figure 8.55 was processed with an adaptive Kuwahara filter resulting in the image in figure 8.57

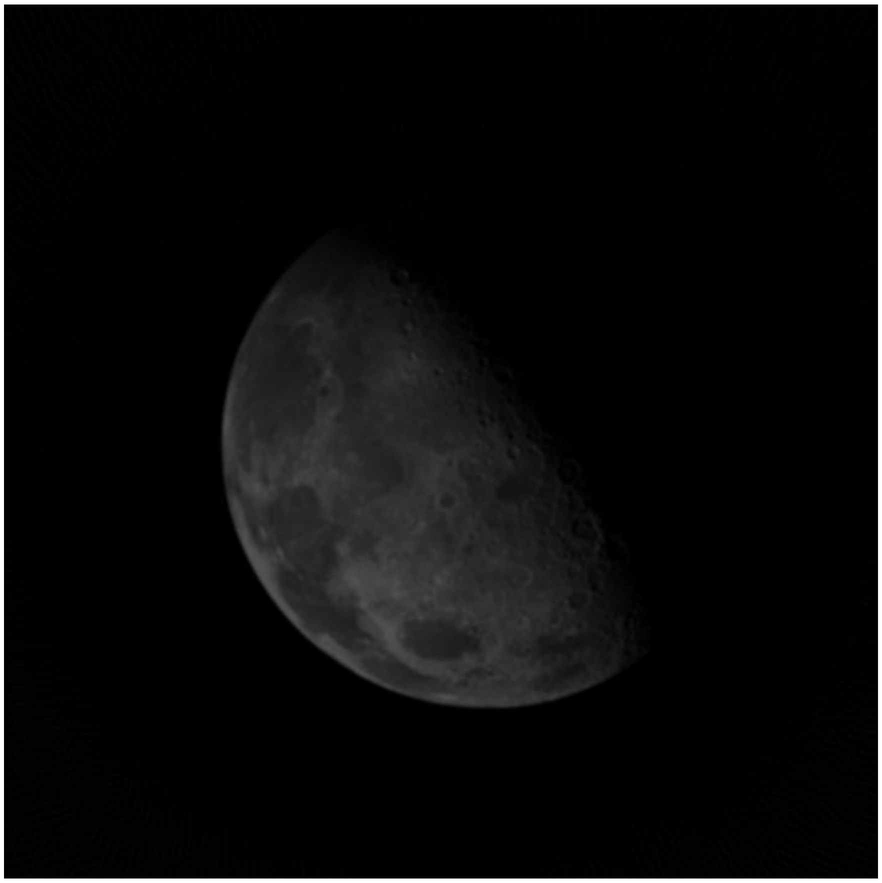

Figure 8.57: Filtered RL Reconstructed Image Using Adaptive Kuwahara Filter with IQR

Consider the image in figure 8.58. This reconstructed image was produced by filtered backprojection using the Shepp-Logan filter.

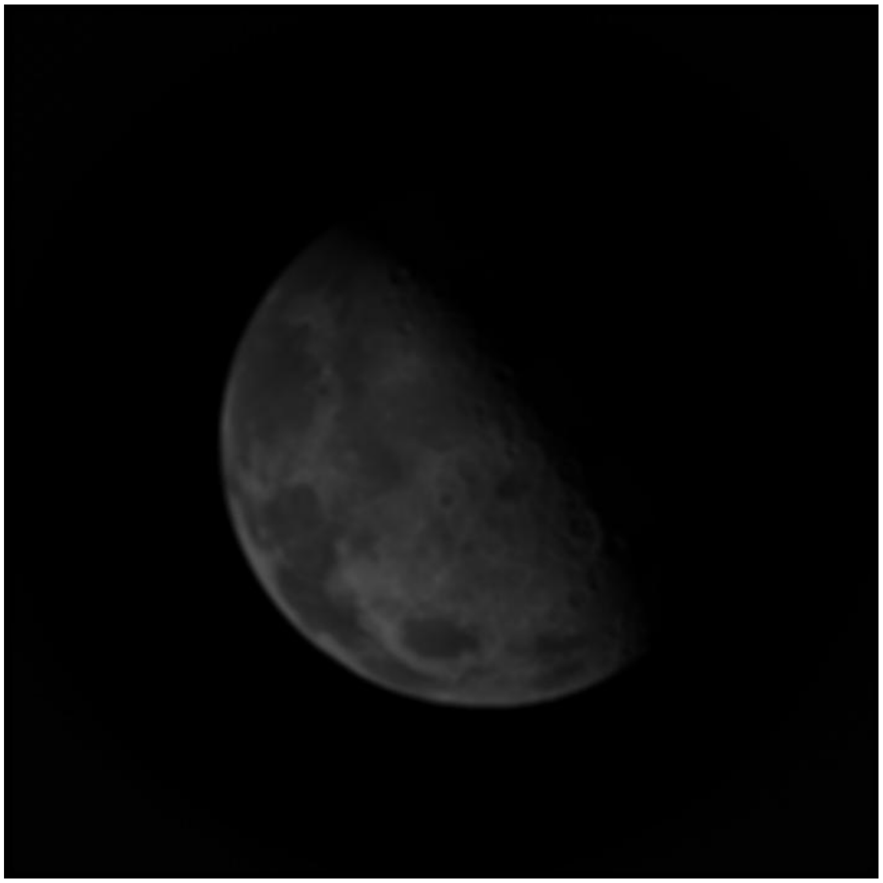

Figure 8.58: Image Reconstructed with Filtered Back Projection and the Shepp-Logan Filter

The image in figure 8.59 is the result of applying a 9x9 Tomita-Tsuji filter with IQR replacing the window variance to the image in figure 8.58.

Finally, the image in figure 8.58 was processed using an adaptive Kuwahara filter with IQR replacing the window variance resulting in the image shown in figure 8.60.

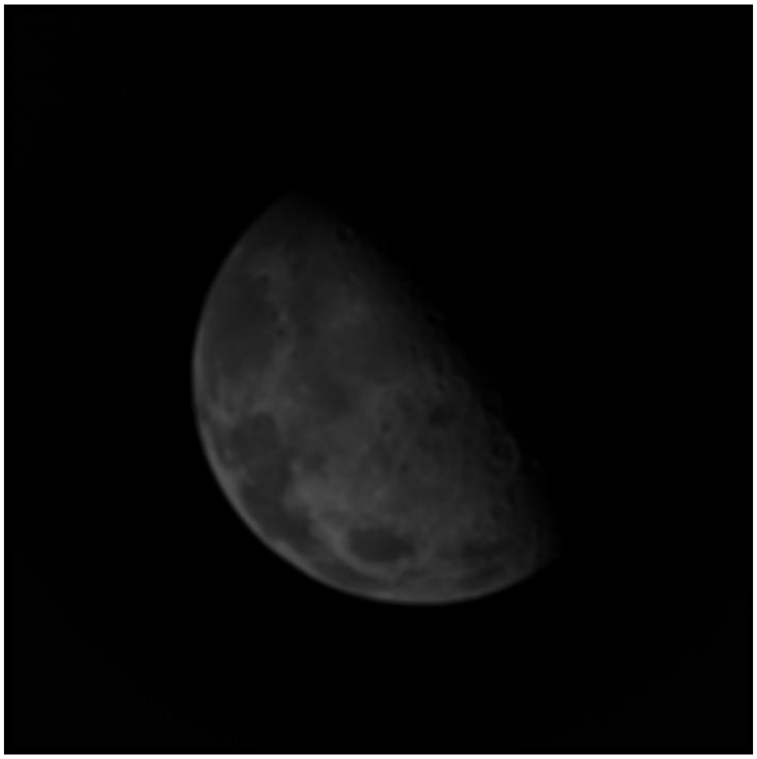

Figure 8.59: SL-Reconstructed Image Filtered with 9x9 Tomita-Tsuji Filter

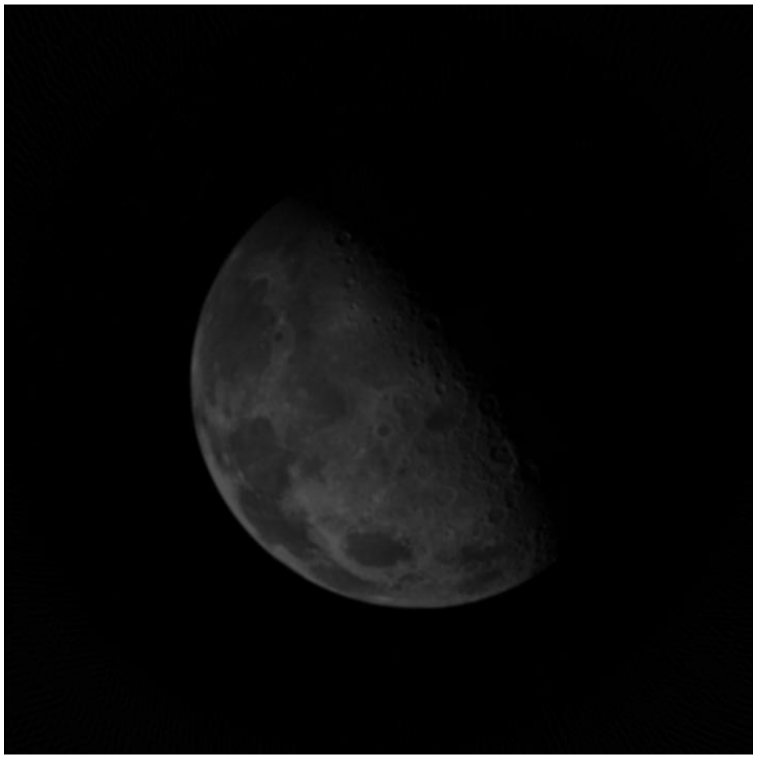

Figure 8.60: SL-Reconstructed Image After Filtering with Adaptive Kuwahara Filter

Consider the image in figure 8.61. This reconstructed image was produced by filtered back projection using the Ram-Lak filter.

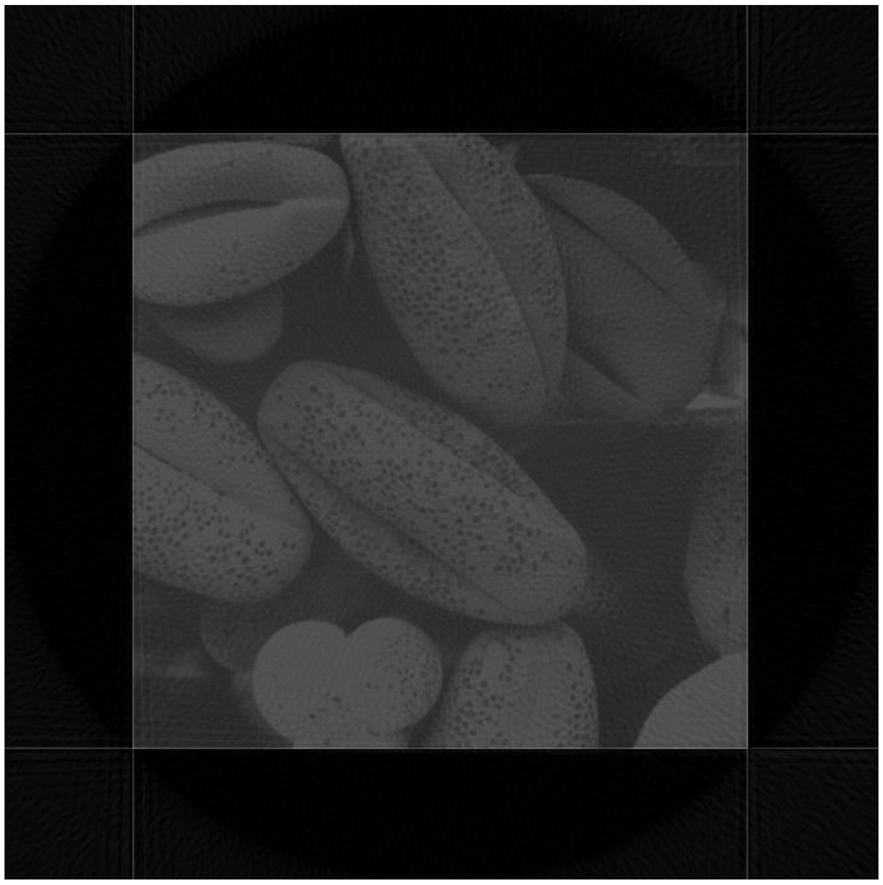

Figure 8.61: Reconstructed Image Using Filtered Back Projection with Ram-Lak Filter

This image was then processed by the 9x9 Tomita-Tsuji filter with IQR replacing window variance resulting in the image shown in figure 8.62

Then, the image in figure 8.61 was processed using an adaptive Kuwahara filter with IQR replacing the window variance resulting in the image shown in figure 8.63

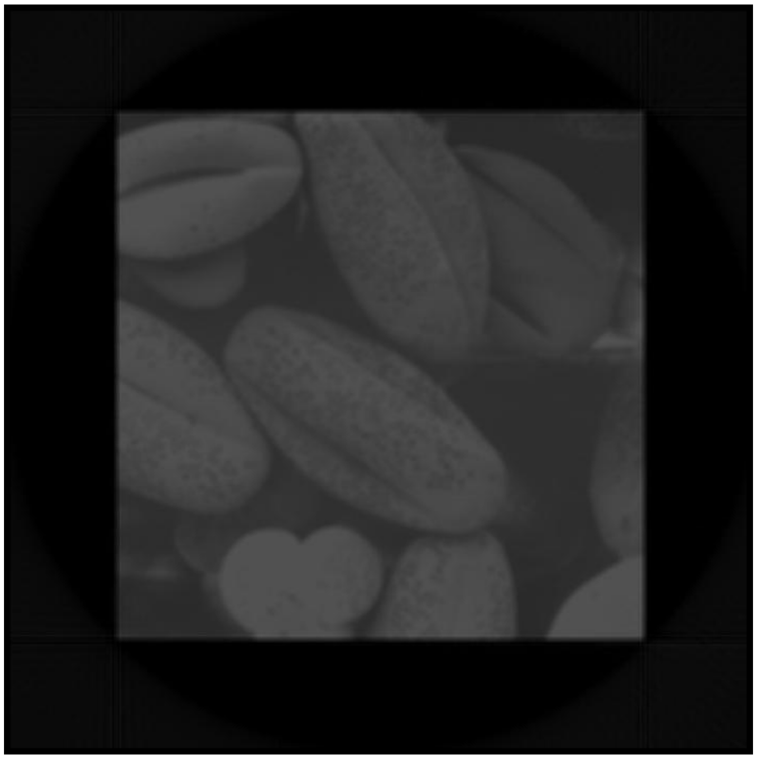

Figure 8.62: Filtered RL Reconstructed Image Using 9x9 Tomita-Tsuji Filter and IQR

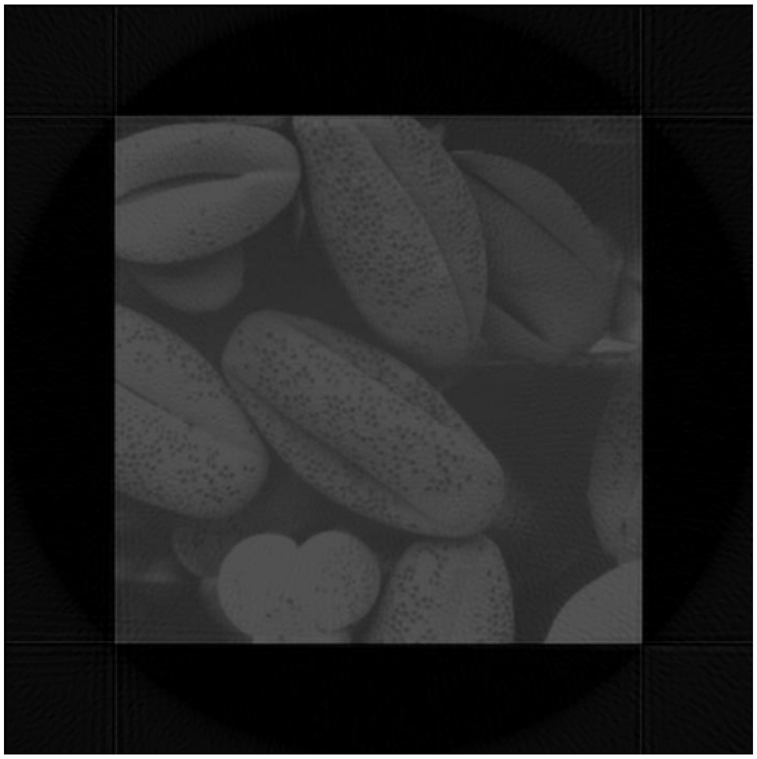

Figure 8.63: RL- Reconstructed Image Filtered with Adaptive Kuwahara Filter with IQR

Consider the image in figure 8.64 which was produced by filtered back projection using the Shepp-Logan filter.

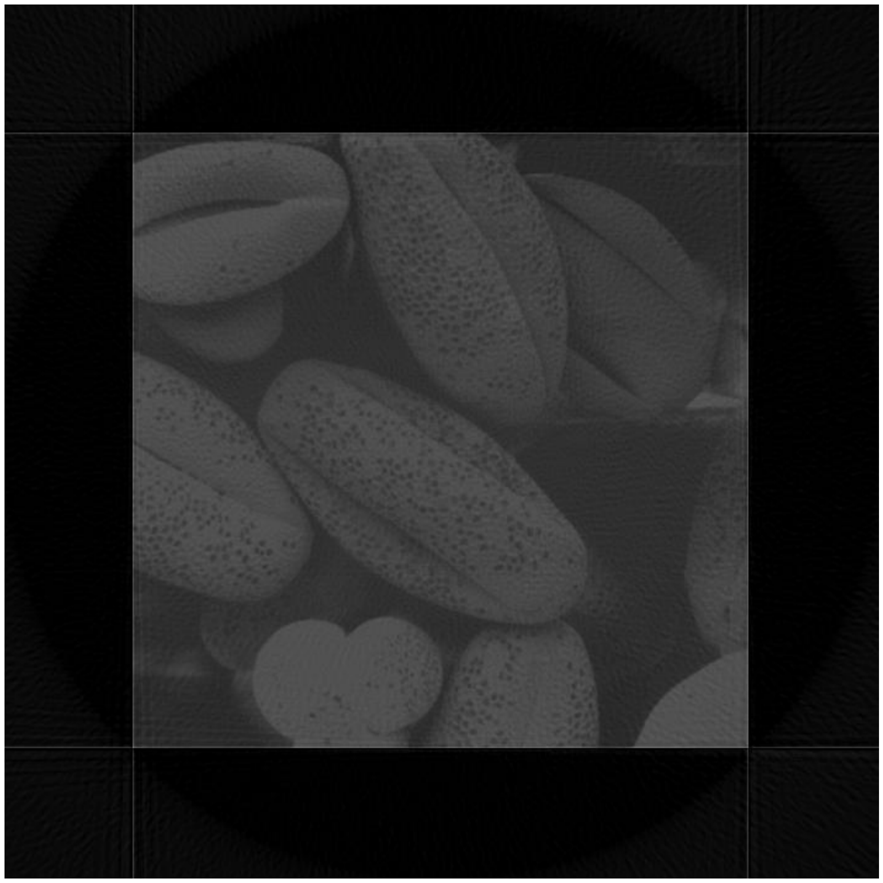

Figure 8.64: Reconstructed Image Using Filtered Back Projection and the Shepp-Logan Filter

The image in figure 8.64 was processed using a 9x9 Tomita-Tsuji filter with IQR replacing the window variance, resulting in the image shown in figure 8.65.

Next, the image in figure 8.64 was processed with an adaptive Kuwahara filter with IQR replacing the window variance, resulting in the image shown in figure 8.66.

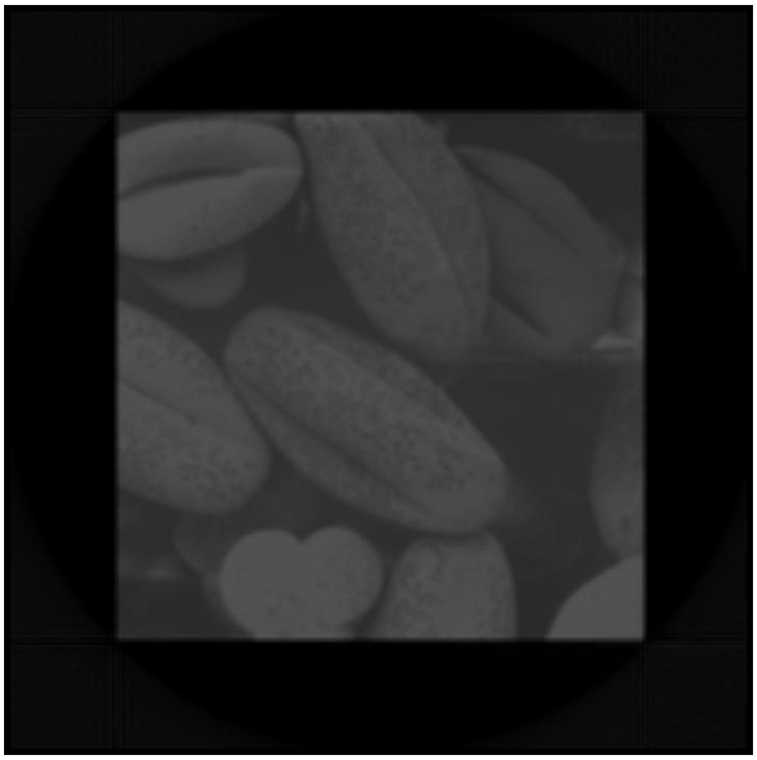

Figure 8.65: SL-Reconstructed Image Filtered with 9x9 Tomita-Tsuji Filter with IQR

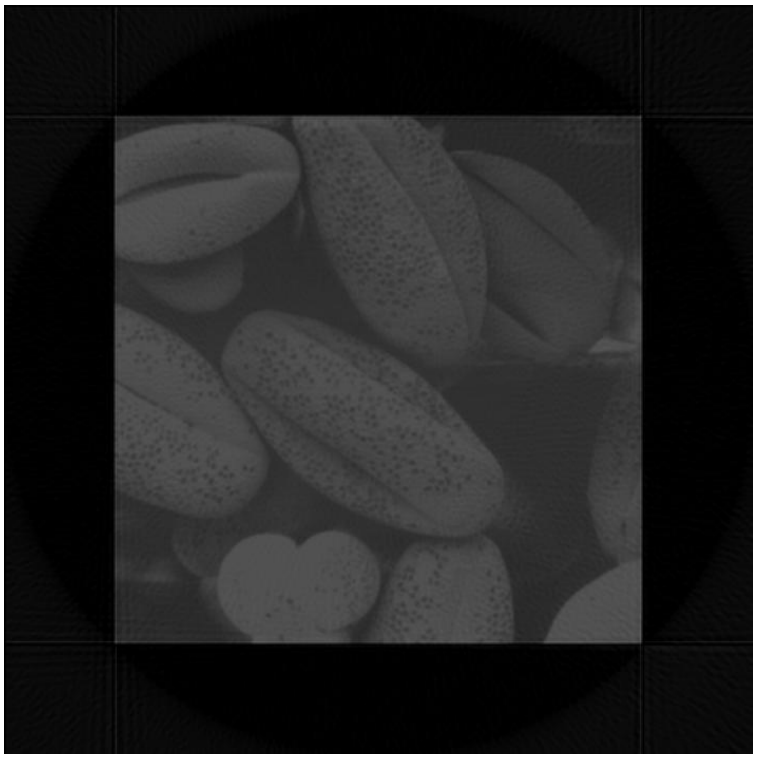

Figure 8.66: SL-Reconstructed Image Filtered with Adaptive Kuwahara Filter with IQR

Consider the image shown in figure 8.67.

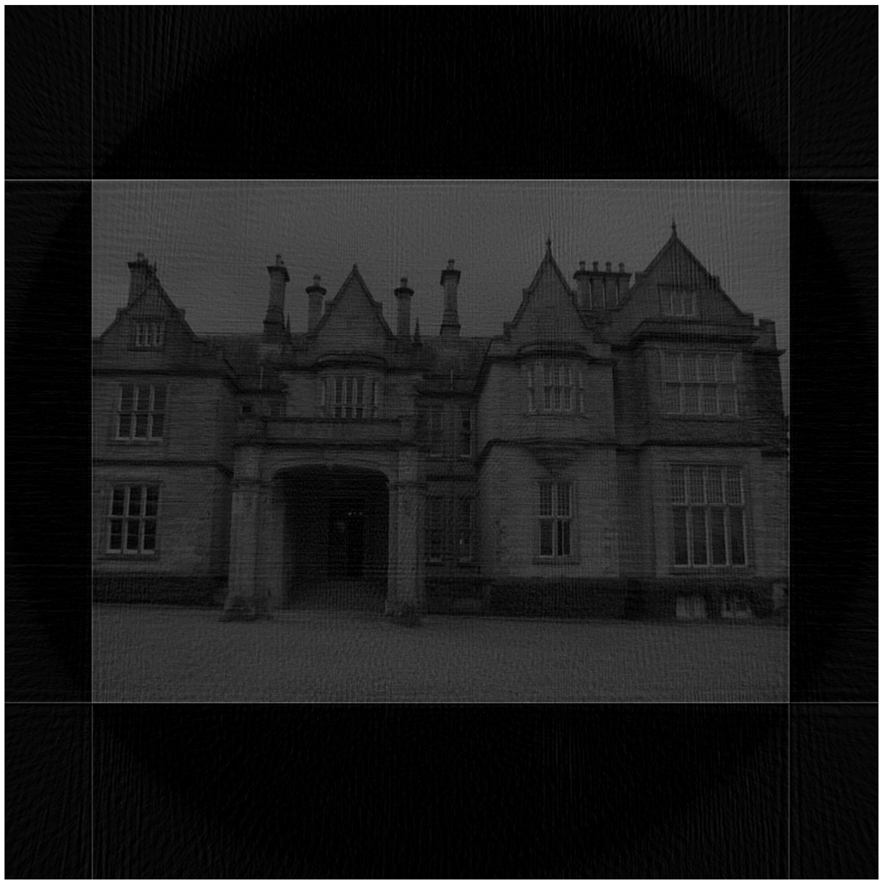

Figure 8.67: Reconstructed Image Produced by Filtered Back Projection with Ram-Lak Filter

The reconstructed image in figure 8.67 was produced by filtered back projection using the Ram-Lak filter. This image was filtered using a 9x9 Tomita-Tsuji filter with IQR replacing the window variance resulting in the image shown in figure 8.68

Next, the image in figure was processed by an adaptive Kuwahara filter with IQR replacing the window variance, resulting in the image in figure 8.69

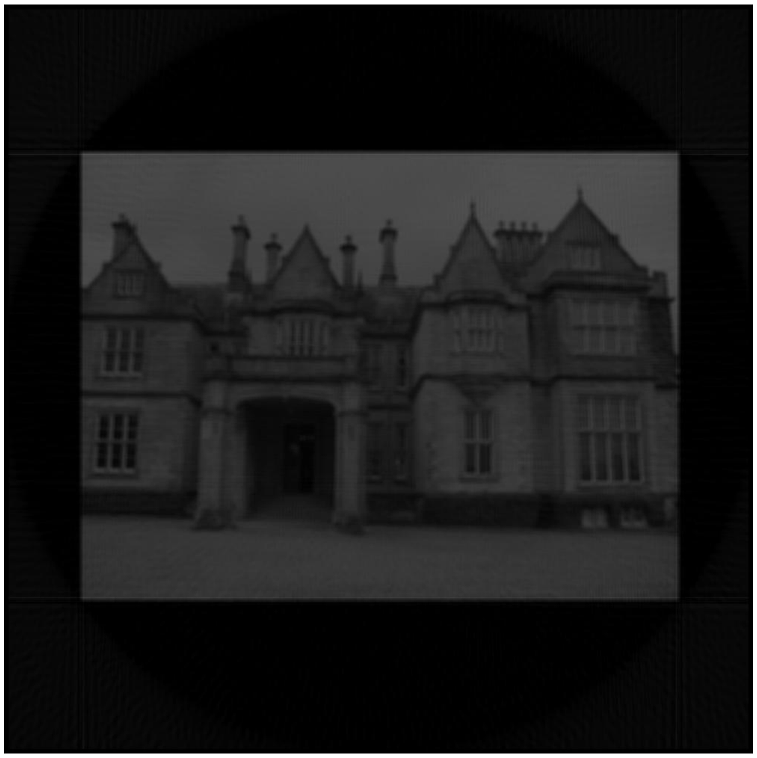

Figure 8.68: RL-Reconstructed Image Using 9x9 Tomita-Tsuji with IQR

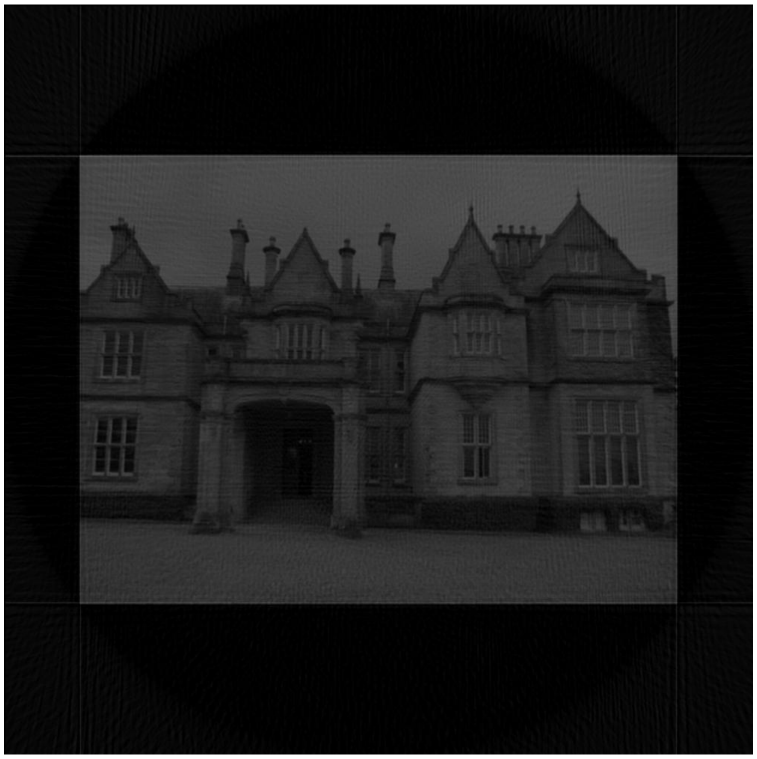

Figure 8.69: RL-Reconstructed Image Filtered with Adaptive Kuwahara Filter with IQR

Consider the image in figure which was produced by filtered back projection with the Shepp-Logan filter.

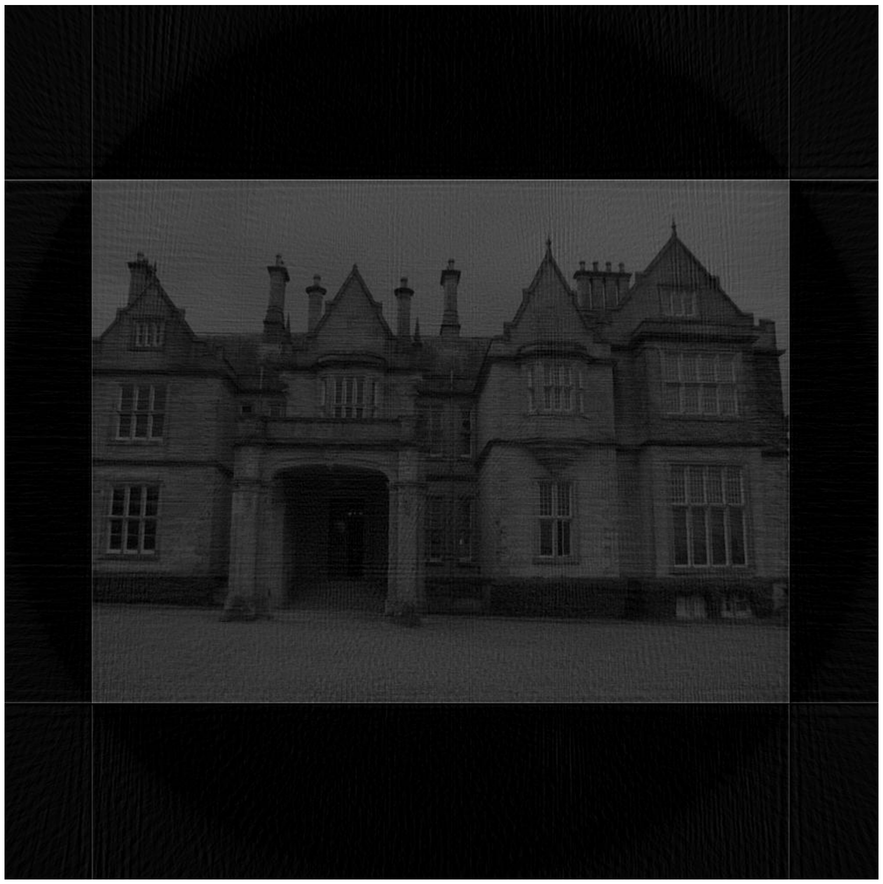

Figure 8.70: Reconstructed Image Produced by Filtered Back Projection with Shepp-Logan Filter

The image in figure 8.70 was processed using a 9x9 Tomita-Tsuji filter with IQR replacing the window variance, resulting in the image shown in figure 8.71

The image in figure was then processed by an adaptive Kuwahara filter with IQR replacing window variance, resulting in the image shown in figure 8.72.

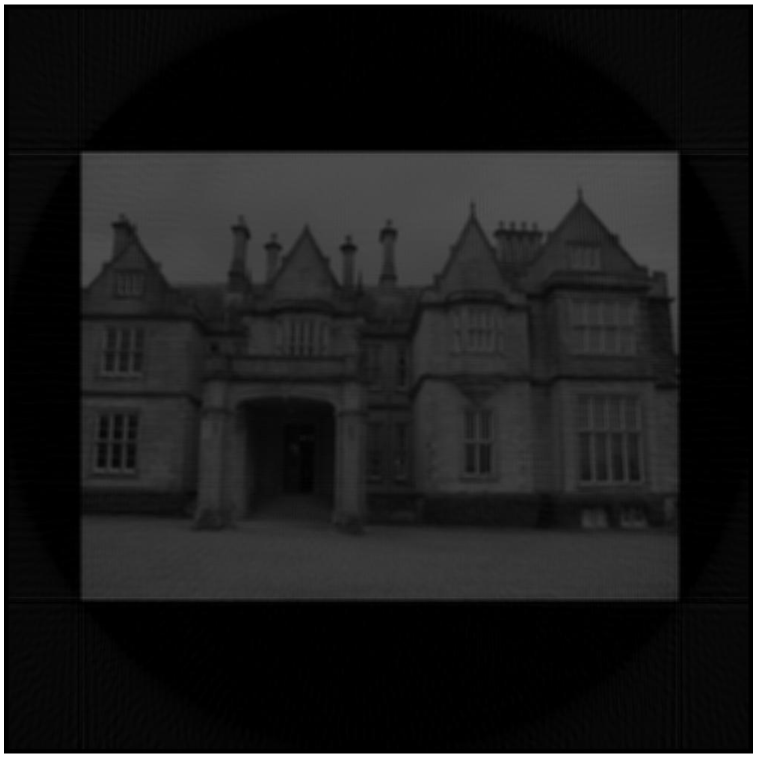

Figure 8.71: SL-Reconstructed Image Filtered with 9x9 Tomita-Tsuji Filter with IQR

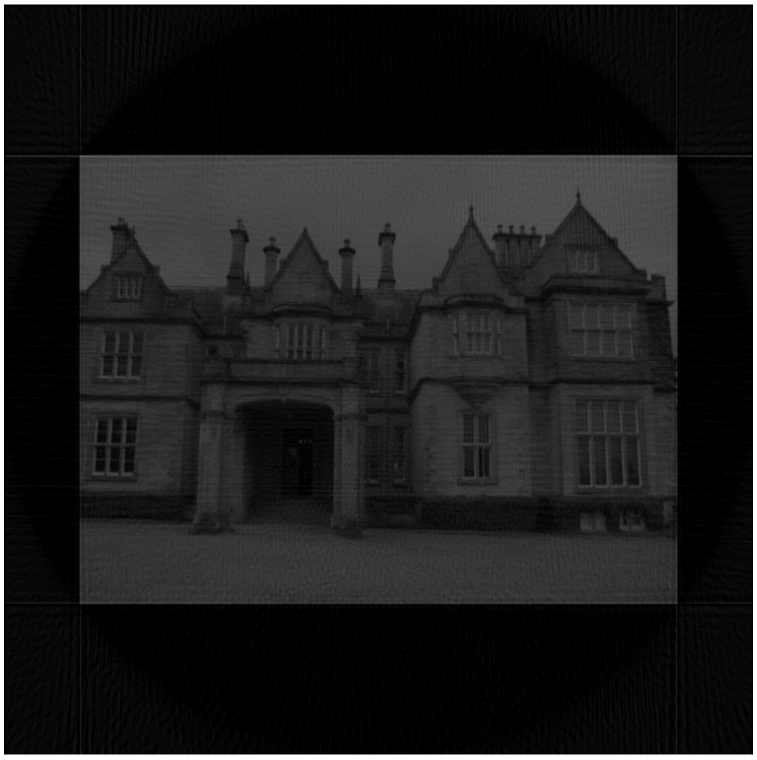

Figure 8.72: SL-Reconstructed Image Filtered with Adaptive Kuwahara Filter with IQR

Table 8.3: Total Variation Values for Grayscale Images

Image	TV Norm
Reconstructed (with RL filter) image-building	3.4244
Reconstructed (with RL filter) and filtered with 3x3 Kuwahara (with IQR)	2.6322
Reconstructed (with RL filter) and filtered with Adaptive Kuwahara (with IQR)	2.1245
Reconstructed (with RL filter) and filtered with 5x5 Kuwahara (with IQR)	1.6886
Reconstructed (with RL filter) and filtered with 9x9 Tomita-Tsuji (with IQR)	0.903
Reconstructed (with SL filter) image-building	3.2268
Reconstructed (with SL filter) and filtered with 3x3 Kuwahara (with IQR)	2.5070
Reconstructed (with SL filter) and filtered with Adaptive Kuwahara (with IQR)	2.0460
Reconstructed (with SL filter) and filtered with 5x5 Kuwahara (with IQR)	1.6407
Reconstructed (with SL filter) and filtered with 9x9 Tomita-Tsuji (with IQR)	0.8796
Reconstructed (with RL filter) image-moon	0.8537
Reconstructed (with RL filter) and filtered with 3x3 Kuwahara (with IQR)	0.7681
Reconstructed (with RL filter) and filtered with Adaptive Kuwahara (with IQR)	0.5907
Reconstructed (with RL filter) and filtered with 5x5 Kuwahara (with IQR)	0.5488
Reconstructed (with RL filter) and filtered with 9x9 Tomita-Tsuji (with IQR)	0.2615
Reconstructed (with SL filter) image-moon	0.8164
Reconstructed (with SL filter) and filtered with 3x3 Kuwahara (with IQR)	0.7403
Reconstructed (with SL filter) and filtered with Adaptive Kuwahara (with IQR)	0.5729
Reconstructed (with SL filter) and filtered with 5x5 Kuwahara (with IQR)	0.5351
Reconstructed (with SL filter) and filtered with 9x9 Tomita-Tsuji (with IQR)	0.2588

The entries in Table 8.3 show that the total variation (TV) norm decreases with increasing window

Table 8.4: Total Variation for Grayscale Images (Continued)

Image	TV Norm
Reconstructed (with RL filter) image-fruits	2.4273
Reconstructed (with RL filter) and filtered with 3x3 Kuwahara (with IQR)	1.9219
Reconstructed (with RL filter) and filtered with Adaptive Kuwahara (with IQR)	1.5776
Reconstructed (with RL filter) and filtered with 5x5 Kuwahara (with IQR)	1.3236
Reconstructed (with RL filter) and filtered with 9x9 Tomita-Tsuji (with IQR)	0.8043
Reconstructed (with SL filter) image-fruits	2.2868
Reconstructed (with SL filter) and filtered with 3x3 Kuwahara (with IQR)	1.8357
Reconstructed (with SL filter) and filtered with Adaptive Kuwahara (with IQR)	1.5303
Reconstructed (with SL filter) and filtered with 5x5 Kuwahara (with IQR)	1.2859
Reconstructed (with SL filter) and filtered with 9x9 Tomita-Tsuji (with IQR)	0.7858
Reconstructed (with RL filter) image-irish	3.8177
Reconstructed (with RL filter) and filtered with 3x3 Kuwahara (with IQR)	2.7818
Reconstructed (with RL filter) and filtered with Adaptive Kuwahara (with IQR)	2.0953
Reconstructed (with RL filter) and filtered with 5x5 Kuwahara (with IQR)	1.9635
Reconstructed (with RL filter) and filtered with 9x9 Tomita-Tsuji (with IQR)	1.026
Reconstructed (with SL filter) image-irish	3.5019
Reconstructed (with SL filter) and filtered with 3x3 Kuwahara (with IQR)	2.6148
Reconstructed (with SL filter) and filtered with Adaptive Kuwahara (with IQR)	2.0142
Reconstructed (with SL filter) and filtered with 5x5 Kuwahara (with IQR)	1.7045
Reconstructed (with SL filter) and filtered with 9x9 Tomita-Tsuji (with IQR)	1.0013

size. Also, the adaptive Kuwahara filter which starts with a 3x3 Kuwahara filter performs better than the standard 3x3 IQR Kuwahara filter.

8.5.1 Filtering of Reconstructed Color Images

We can apply the Kuwahara filter to color images using the IQR instead of the variance of a processing window.

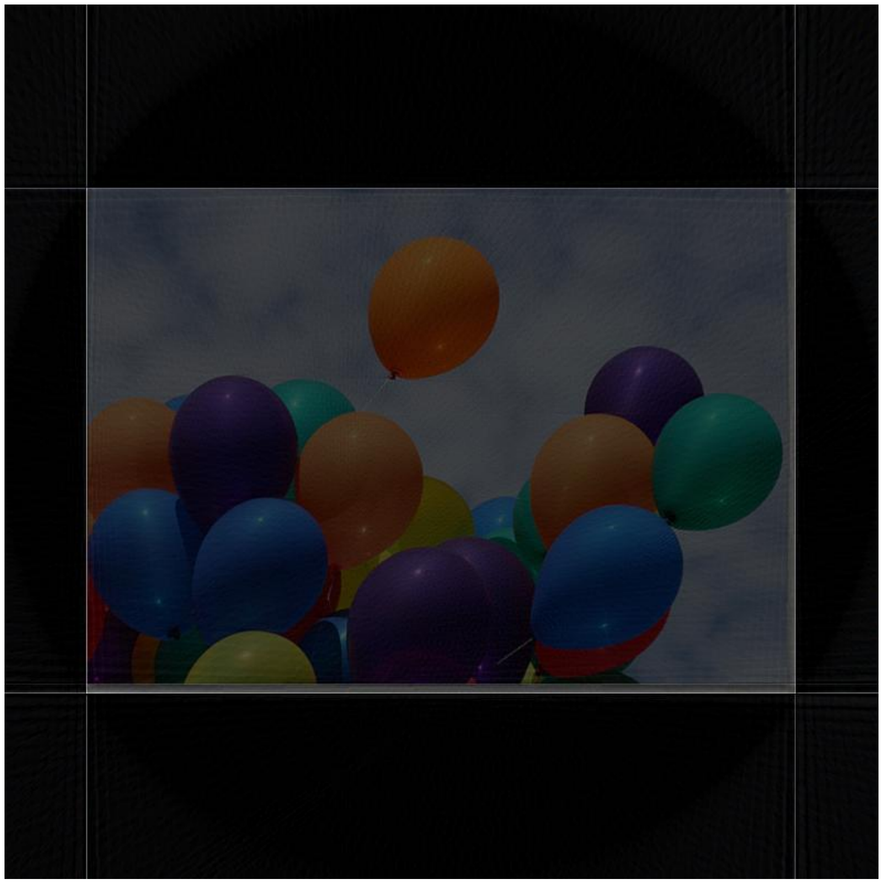

Figure 8.73: Reconstructed Balloons by Filtered Back Projection with the Ram-Lak Filter

When the Ram-Lak reconstructed image in figure 8.73 is processed by the color Kuwahara filter using IQR instead of variance, the resulting image is shown in figure 8.74 below.

The image in figure 8.75 below shows the result of applying an adaptive 3x3 color Kuwahara filter using IQR instead of variance.

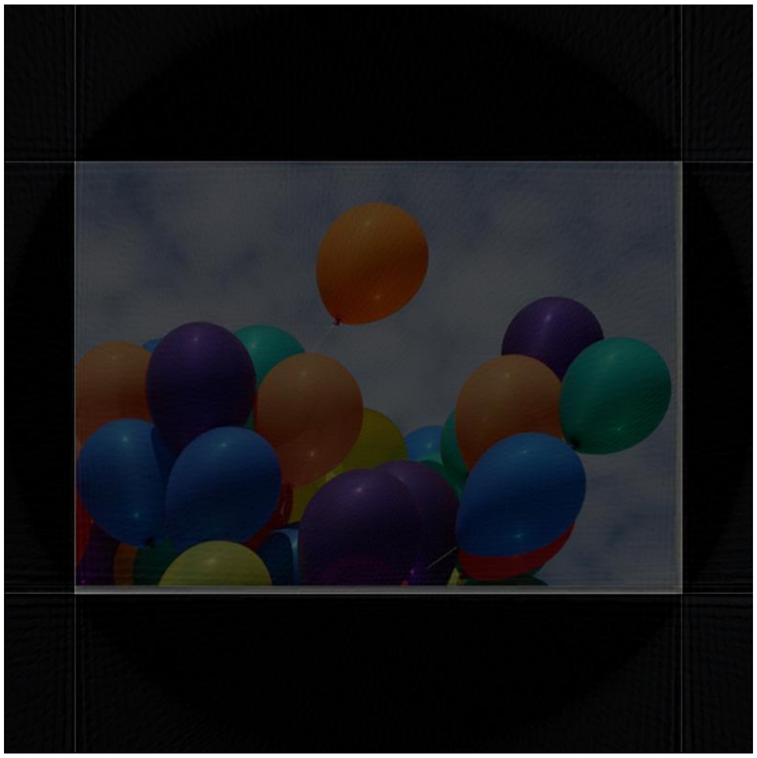

Figure 8.74: RL Color Kuwahara with IQR

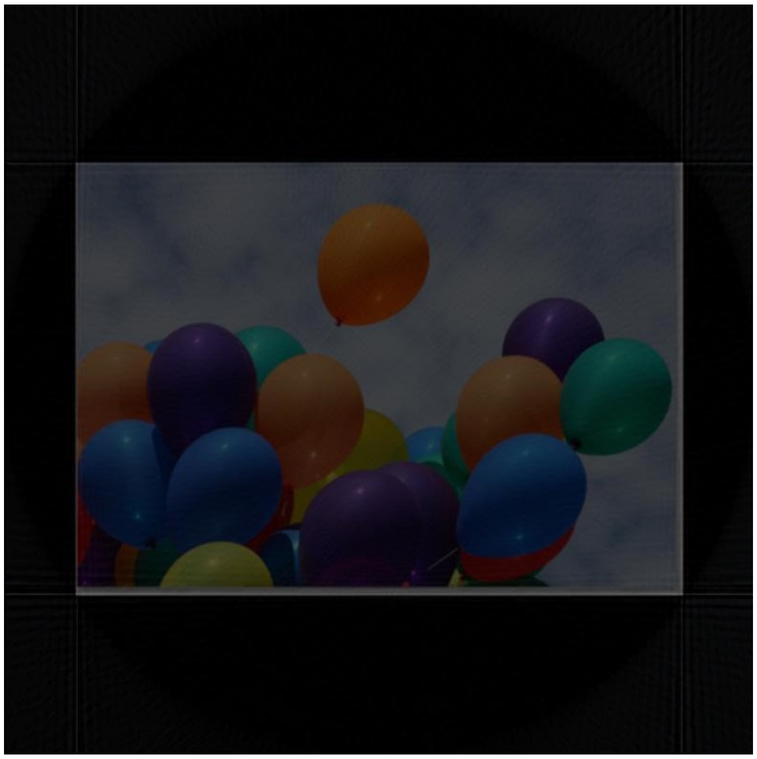

Figure 8.75: RL Adaptive Color 3x3 Kuwahara Filter Using IQR

Consider now the image in figure 8.76 It was produced by filtered back projection using the Shepp-Logan filter. Then, this image was filtered using a 3x3 Kuwahara filter with IQR replacing the window variance, resulting in the image shown in figure 8.77

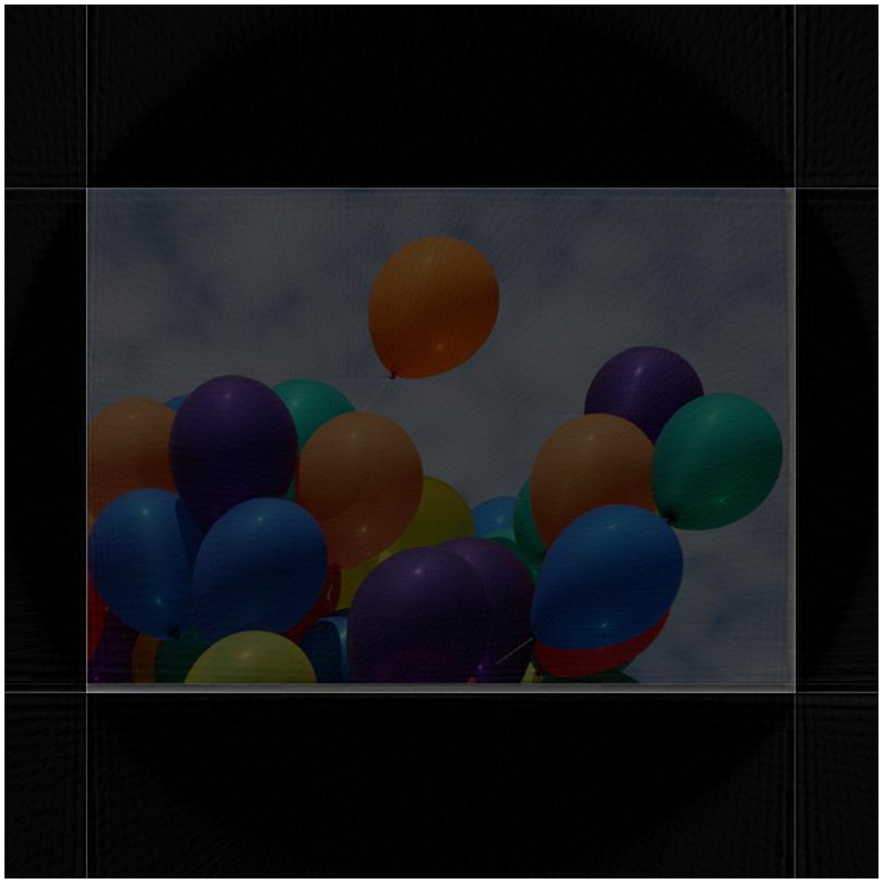

Figure 8.76: Reconstructed Image Produced by Filtered Back Projection with Shepp-Logan Filter

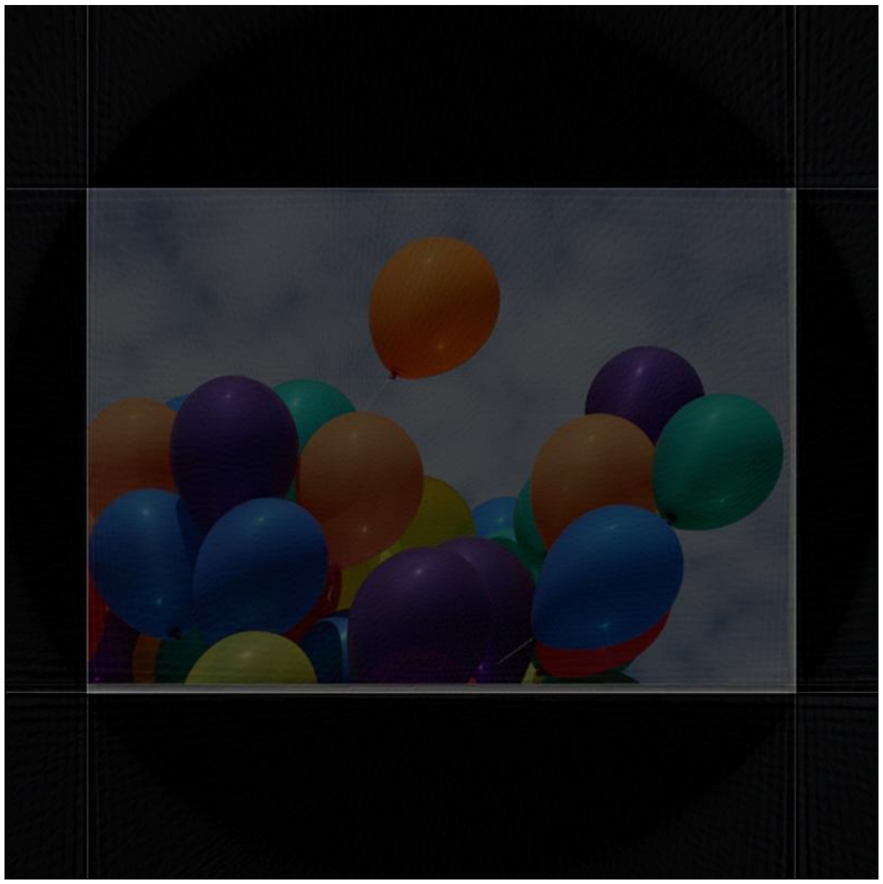

Figure 8.77: SL-Reconstructed Image Filtered with 3x3 Kuwahara Filter with IQR

Next, the image in figure 8.76 was filtered with an adaptive Kuwahara filter with IQR replacing the window variance, resulting in the image shown in figure 8.78

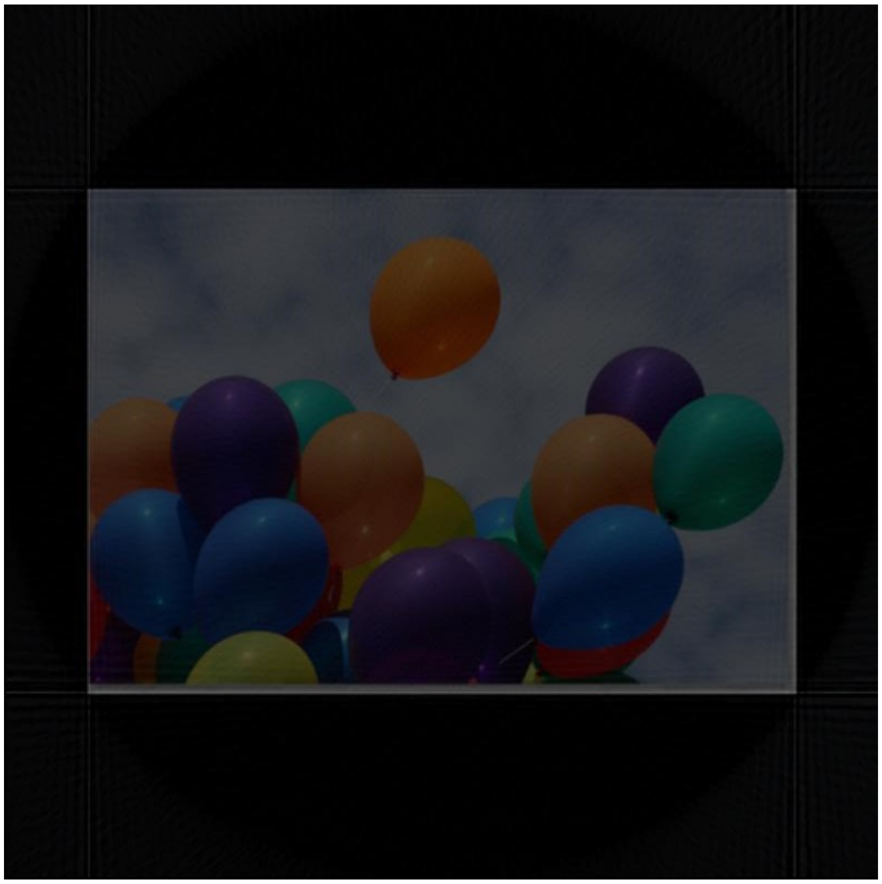

Figure 8.78: SL-Reconstructed Image Filtered with Adaptive Kuwahara Filter with IQR

Consider the image in figure 8.79. It was produced by filtered back projection with the Ram-Lak filter. Then, this reconstructed image was filtered with a 3x3 Kuwahara filter with window variance replaced with IQR, resulting in the image shown in figure 8.80

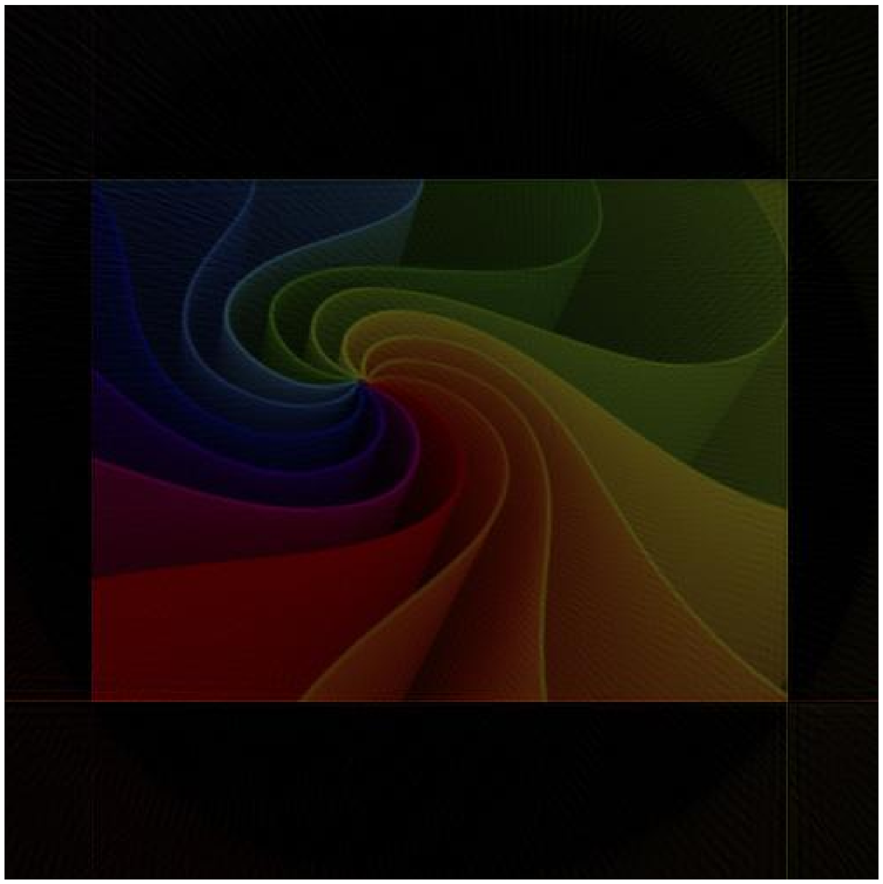

Figure 8.79: Reconstructed Image Produced by Filtered Back Projection with Ram-Lak Filter

Next, the image in figure 8.79 was filtered with an adaptive Kuwahara filter with IQR replacing the window variance, resulting in the image in figure 8.81

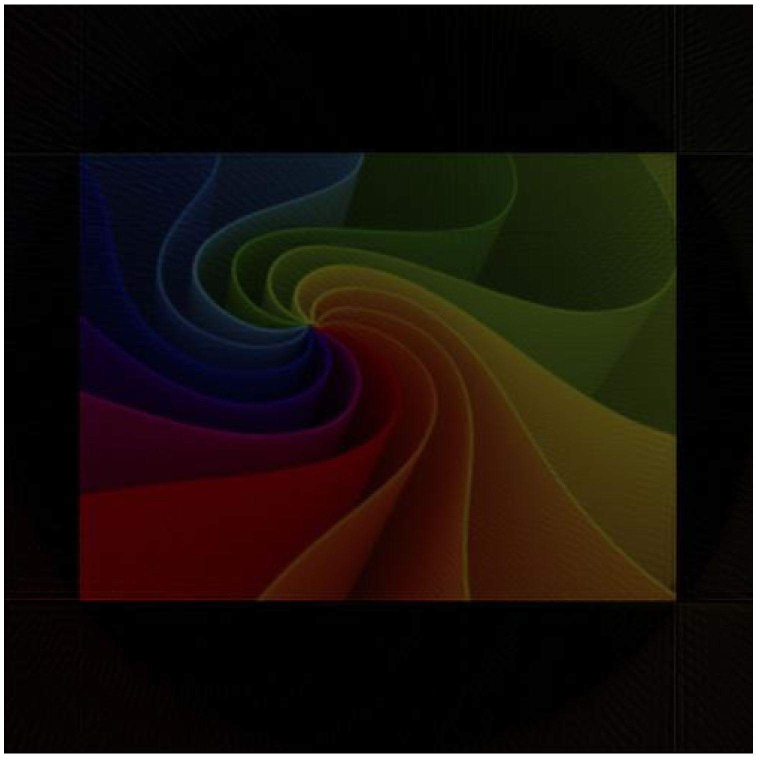

Figure 8.80: RL-Reconstructed Image with 3x3 Kuwahara Filter with IQR

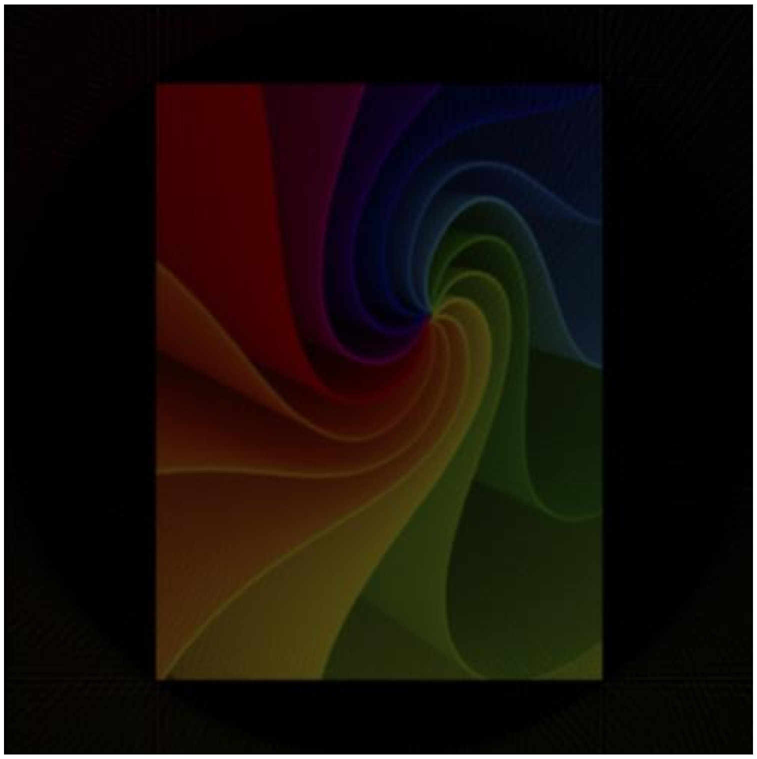

Figure 8.81: RL-Reconstructed Image Filtered with Adaptive Kuwahara Filter with IQR

Consider the image in figure which was produced by filtered back projection using the Shepp-Logan filter. Then, this image was filtered using a 3x3 Kuwahara filter with IQR replacing window variance, resulting in the image in figure 8.83

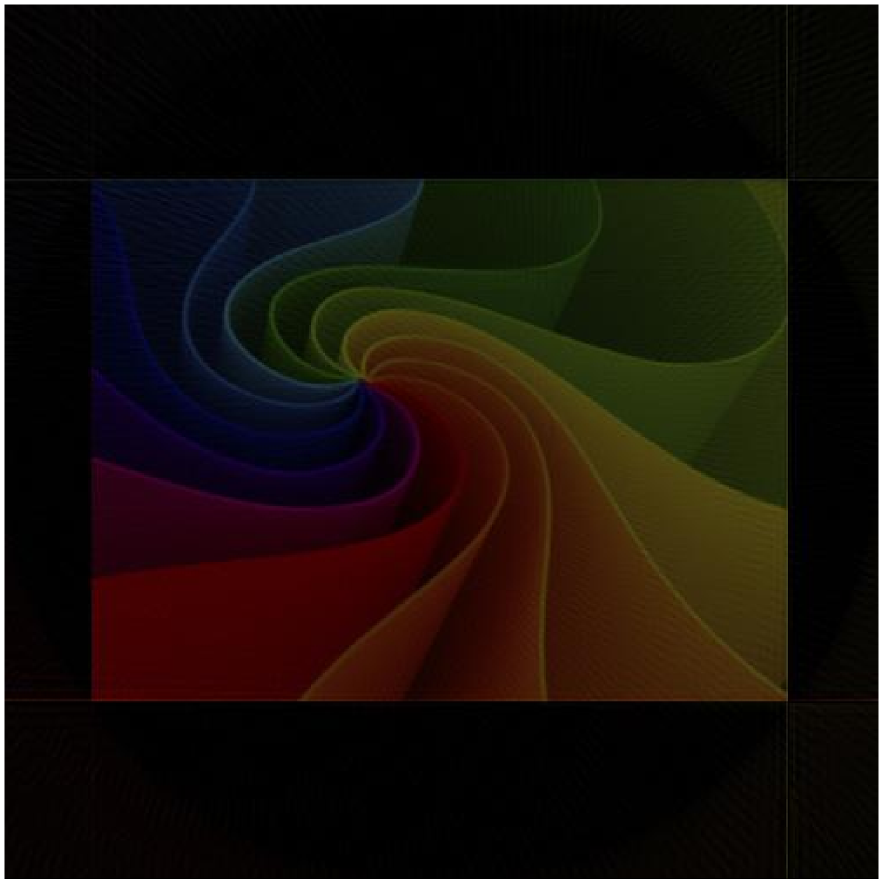

Figure 8.82: Reconstructed Image Produced by Filtered Back Projection with Shepp-Logan Filter

Finally, the image in figure 8.82 was filtered with an adaptive Kuwahara filter with IQR replacing the window variance, resulting in the image in figure 8.84

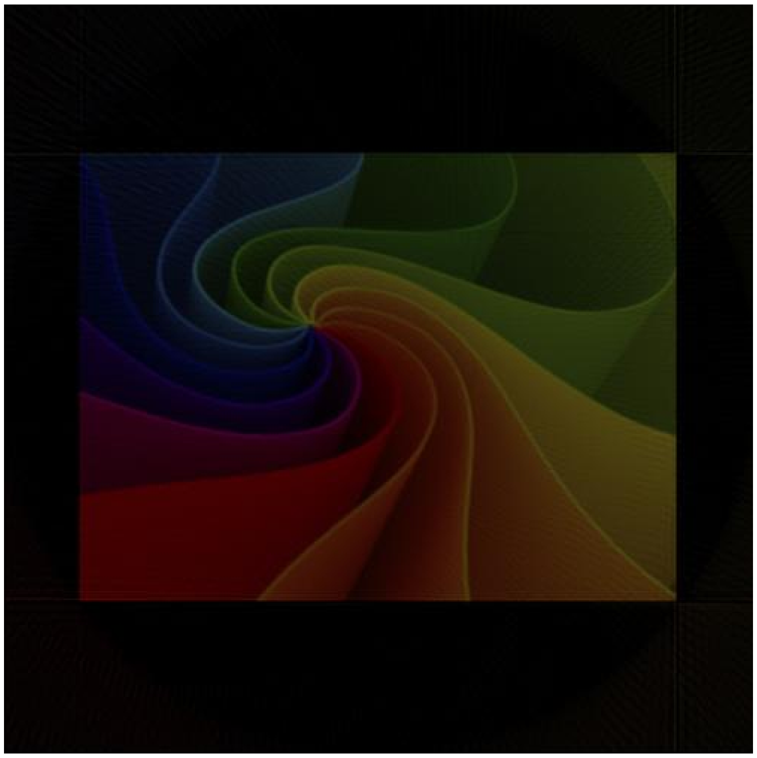

Figure 8.83: SL-Reconstructed Image Filtered with 3x3 Kuwahara Filter with IQR

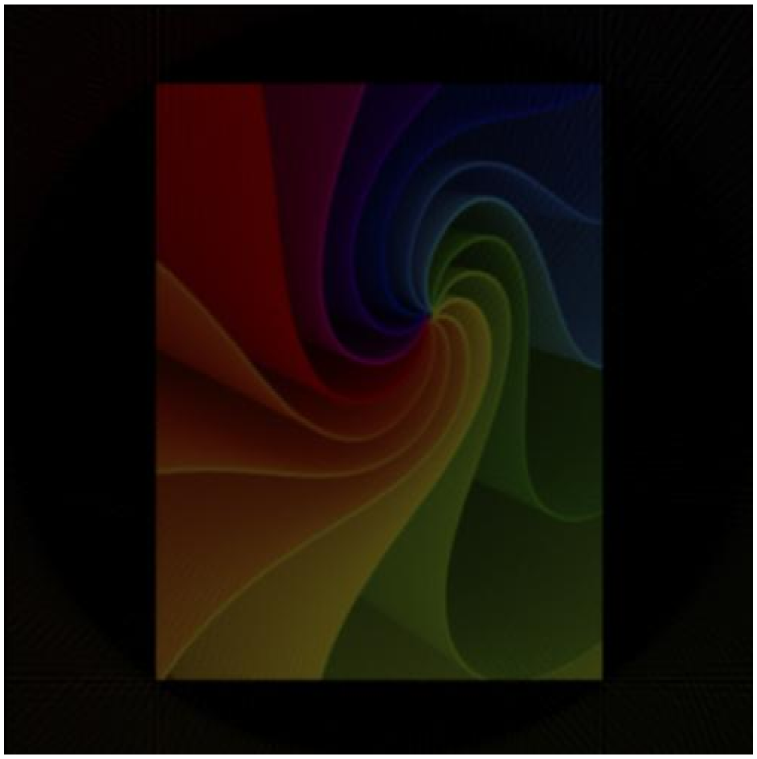

Figure 8.84: SL-Reconstructed Image Filtered with Adaptive Kuwahara Filter with IQR

Table 8.5: Total Variation Values for Color Images

Image	TV Red Norm	TV Green Norm	TV Blue Norm
RL-Reconstructed balloons image	2.3640	2.1695	2.4539
RL-Recon and 3x3 Kuwahara with IQR	1.8414	1.6854	1.9145
RL-Recon and 5x5 Kuwahara with IQR	1.4364	1.3105	1.4877
RL-Recon and Adaptive Kuwahara with IQR	1.40	1.2788	1.4508
SL-Reconstructed balloons image	2.1999	2.0149	2.2815
SL-Recon and 3x3 Kuwahara with IQR	1.7569	1.6065	1.8277
SL-Recon and 5x5 Kuwahara with IQR	1.3942	1.2733	1.4441
SL-Recon and Adaptive Kuwahara with IQR	1.3578	1.2405	1.4070
RL-Reconstructed color-psychology image	1.9092	1.6846	1.5265
RL-Recon and 3x3 Kuwahara with IQR	1.5271	1.3440	1.2062
RL-Recon and 5x5 Kuwahara with IQR	1.1893	1.0309	0.8938
RL-Recon and Adaptive Kuwahara with IQR	1.1507	0.9994	0.8550
SL-Reconstructed color-psychology image	1.7813	1.5696	1.4194
SL-Recon and 3x3 Kuwahara with IQR	1.4528	1.2762	1.1401
SL-Recon and 5x5 Kuwahara with IQR	1.1505	0.9953	0.8584
SL-Recon and Adaptive Kuwahara with IQR	1.116	0.9673	0.8228

The TV norm in each of the three color channels, red, green, and blue is a measure in each of those channels of the noise or extraneous detail in the image. Now, since the entries for the reconstructed and adaptive filter color images show values which are less in each of the three color channels than those values for the reconstructed filtered images (using a 3x3 Kuwahara filter), the adaptive Kuwahara filter improves the image quality over the 3x3 and 5x5 Kuwahara filters for both the

Ram-Lak and Shepp-Logan reconstructed images, and for both the balloons image and the color psychology image.

Chapter 9

Conclusions

A number of algorithms have been presented for accomplishing various purposes in images: crack endpoint detection, detection of crack intersection points, orientation detection of lines, determination of line thickness, and filtering of reconstructed images.

The crack endpoint detection algorithm worked by scanning across a binary crack image and determining the left endpoint of a crack and then the right endpoint of a crack. The cracks were so different from each other that a separate set of code was required for different cracks. Code was developed and tested on two cracks.

The detection of crack intersections worked by reading in a binary crack image and computing vertical differences between two parts of a crack to determine where they intersect. Horizontal differences were used when the two crack branches had more of a vertical characteristic.

The orientation detection algorithm computes the Radon transform (sinogram) of an image of lines at various angles. Then, the maximum value of the sinogram is computed at each angle and is plotted as a function of angle. This allows for the selection of a threshold that will separate the actual Radon transform-detected angles from the rest of the values in the sinogram, or the clutter level. The clutter in the image is caused by the number of lines at different angles being in close proximity to each other as they emerge from one point. The clutter level is evident when the histogram of the maximum values of the sinogram is plotted. The fact that the maximum sinogram value obtained at the angle of each line in the image is much greater than the level of the clutter allows for the selection of a threshold that will separate the actual Radon transform-detected angles from the clutter values in the sinogram. That threshold is chosen to be the mean of the maximum sinogram values.

Figure 8.18 above shows the input image with 35 lines at different angles (starting at 2 degrees, with 2-degree increments, up to 70 degrees). Figure 8.19 above shows a histogram of the maximum of the sinogram at each angle. A count of the histogram peaks reveals 31 angles. However, the sinogram of figure 8.18 shows 35 peaks at 35 angles corresponding to the 35 detected lines. The mean of the 14 detected angles was computed to be 37.5 degrees which is exactly the mean of the 14 input angles. Also, the standard deviation of those 14 detected angles is 20.1556 degrees. Similarly, the mean of the 35 detected angles was computed to be 36 degrees which is precisely the mean of the 35 input angles. Also, the standard deviation of those 35 detected angles is 20.199 degrees.

The line thickness algorithm consisted of four parts, a part to detect horizontal lines and determine their thickness, a part to detect vertical lines and determine their thickness, a part to detect single pixel thick lines at an angle, a part to detect multiple pixel thick lines with a positive slope and determine their thickness, and a part to detect multiple pixel thick lines with a negative slope and determine their thickness. One image to which this algorithm was applied has eight lines of varying thicknesses. The far-right vertical line has a thickness of 5 pixels. The vertical line in the middle of the image has a thickness of 2 pixels. The far-left vertical line has a thickness of 3 pixels. The top horizontal line has a thickness of 4 pixels. The horizontal line in the middle of the image has a thickness of 1 pixel. The bottom horizontal line has a thickness of 2 pixels. The diagonal line that is decreasing from left to right has a thickness of 3 pixels. The diagonal line that is increasing from left to right has a thickness of 4 pixels. The sum of all of the pixel widths is equal to 24. The number of lines is 8. Therefore, the average pixel thickness is 3 with a standard deviation of 1.2247. This is exactly what the MATLAB code determined.

The last problem considered was that of filtering images which had been Radon-transformed and then reconstructed from the sinograms using filtered backprojection with either the Ram-Lak or Shepp-Logan filter. Four gray scale images, a building, fruit, another building, and the moon were all Radon-transformed, producing sinograms. The projections forming the sinogram were then convolved with a high-pass filter (either Ram-Lak or Shepp-Logan) and then integrated (or summed in the computer approximation to integration) from 0 to 179 degrees in order to produce the inverse Radon transform (or an approximation to it on the computer). This filtered backprojection produced filtered laminograms or reconstructed images. Then, Kuwahara-like filters were utilized to remove the noise and blurring in the reconstructed images. However, instead of computing the mean and variance in a window around each pixel in the image, the mean and the interquartile range (IQR) were instead computed. Then, a 9x9 Tomita-Tsuji filter with a 7x7 subregion filter was used to remove the blurring. Next, an adaptive Kuwahara filter was employed. This filter started with a window size of 3x3 using 2x2 subregions whose size could grow if the IQR of the next greater window size was less than the IQR of the current window size. The Kuwahara filter was also used for a 5x5 filter with 3x3 subregions as well as a 3x3 filter with 2x2 subregions. These filters were applied to each Ram-Lak reconstructed image as well as to each Shepp-Logan reconstructed image. Next, the total variation (TV) norm was computed on the filtered images. The total variation measures the amount of blurring or noise in the image. The idea is that the lower the TV norm, the more a particular filter is reducing noise or blurring. The adaptive Kuwahara filter always performed better than the 3x3 filter and slightly worse than the 5x5 filter. Finally, the adaptive Kuwahara filter was used on two color images, an image containing balloons and a color psychology image. Here, the filter had to be applied to each color channel (red, green, and blue) separately. An adaptive Kuwahara filter was applied to each Ram-Lak reconstructed image as well as to each Shepp-Logan reconstructed image. Here, the adaptive Kuwahara filter with an inital 3x3 window with 2x2 subregions performed better than than the Kuwahara filter with a 3x3 window and 2x2 subregions and better than the Kuwahara filter with a 5x5 filter with 3x3 subregions.

Bibliography

[1] S. R. Deans, "Hough transform from the radon transform," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 185–188, 03 1981.

[2] M. Lyra and A. Ploussi, "Filtering in spect image reconstruction," *Journal of Biomedical Imaging*, 2011.

[3] O. Demirkaya, "Reduction of noise and image artifacts in computed tomography by nonlinear filtration of projection images," *Proceedings of SPIE*, vol. 4322, 7 2001.

[4] F. Yang, D. Zhang, K. Huang, and Y. Yang, "Image artifacts and noise reduction algorithm for cone-beam computed tomography with low-signal projections," *Journal of X-Ray Science and Technology*, vol. 26, pp. 1–14, 10 2017.

[5] C. Ehman, L. Yu, A. Manduca, A. Hara, M. Shiung, D. Jondal, D. Lake, R. Paden, D. Blezek, Z. Li, M. Bruesewitz, C. McCollough, D. Hough, and J. Fletcher, "Methods for clinical evaluation of noise reduction methods at computed tomography," in *Radiological Society of North America 2012 Scientific Assembly and Annual Meeting*, 01 2012.

[6] F. E. Boas, "Iterative reduction of artifacts in computed tomography images using forward projection and an edge-preserving blur filter," *U.S. Patent 8,233,586 B1*, 07 2012.

[7] Y. Hung, "Shearography: A new optical method for strain measurement and nondestructive testing," *Optical Engineering*, 06 1982.

[8] E. D. S. Loutridis and L. Hadjileontiadis, "Forced vibration behaviour and crack detection of cracked beams using instantaneous frequency," *NDT & E International*, 07 2005.

[9] D. M. Blunt and J. A. Keller, “Detection of a fatigue crack in a uh-60a planet gear carrier using vibration analysis,” *Mechanical Systems and Signal Processing*, 11 2006.

[10] D. G. Yongyong He and F. Chu, “Using genetic algorithms and finite element methods to detect shaft crack for rotor-bearing system,” *Mathematics and Computers in Simulation*, 08 2001.

[11] S. N. J.-i. T. Ichiro Komura, Taiji Hirasawa and K. Naruse, “Crack detection and sizing technique by ultrasonic and electromagnetic methods,” *Nuclear Engineering and Design*, 06 2001.

[12] A. Dutta and S. Talukdar, “Damage detection in bridges using accurate modal parameters,” *Finite Elements in Analysis and Design*, 01 2004.

[13] M. Rucka and K. Wilde, “Application of continuous wavelet transform in vibration based damage detection method for beams and plates,” *Journal of Sound and Vibration*, pp. 536–550, 11 2006.

[14] Q. Wang and X. Deng, “Damage detection with spatial wavelets,” *International Journal of Solids and Structures*, pp. 3443–3468, 08 1999.

[15] C.-C. Chang and L.-W. Chen, “Detection of the location and size of cracks in the multiple cracked beam by spatial wavelet based approach,” *Mechanical Systems and Signal Processing*, pp. 139–155, 01 2005.

[16] X. Zhu and S. Law, “Wavelet-based crack identification of bridge beam from operational deflection time history,” *International Journal of Solids and Structures*, pp. 2299–2317, 04 2006.

[17] L. Z. Ser-Tong Quek, Quan Wang and K.-K. Ang, “Sensitivity analysis of crack detection in beams by wavelet technique,” *International Journal of Mechanical Sciences*, pp. 2899–2910, 12 2001.

[18] X. C. Jiawei Xiang, Yongteng Zhong and Z. He, “Crack detection in a shaft by combination of wavelet-based elements and genetic algorithm,” *International Journal of Solids and Structures*, pp. 4782–4795, 08 2008.

[19] V. L. Peggy Subirats, Jean Dumoulin and D. Barba, “Automation of pavement surface crack detection using the continuous wavelet transform,” *2006 International Conference on Image Processing (ICIP)*, 10 2006.

[20] J. H. M. Krause and W. Krenkel, “(micro)-crack detection using radon transform,” *Material Science and Engineering:A*, pp. 7126–7131, 10 2010.

[21] Y. O. Ouma and M. Hahn, “Wavelet-morphology based detection of incipient linear cracks in asphalt pavements from rgb camera imagery and classification using circular radon transform,” *Advanced Engineering Informatics*, pp. 481–499, 08 2016.

[22] F. M. Nejad and H. Zakeri, “An optimum feature extraction method based on wavelet-radon transform and dynamic neural network for pavement distress classification,” *Expert Systems With Applications*, pp. 9442–9460, 08 2011.

[23] F. M. Nejad and H. Zakeri, “An expert system based on wavelet transform and radon neural network for pavement distress classification,” *Expert Systems with Applications*, pp. 7088–7101, 06 2011.

[24] F. M. Nejad and H. Zakeri, “A comparison of multi-resolution methods for detection and isolation of pavement distress,” *Expert Systems with Applications*, pp. 2857–2872, 03 2011.

[25] C. M. Yeum and S. J. Dyke, “Vision-based automated crack detection for bridge inspection,” *Computer-Aided Civil and Infrastructure*, 05 2015.

[26] L. Ying and E. Salari, “Beamlet transform-based technique for pavement crack detection and classification,” *Computer-Aided Civil and Infrastructure Engineering*, 06 2010.

[27] S. Zhibiao and G. Yanqing, “Algorithm on contourlet domain in detection of road cracks for pavement images,” *Journal of Algorithms and Computational Technology*, 03 2013.

[28] M. K. Lu Sun and Y. Zhang, "Weighted neighborhood pixels segmentation method for automated detection of cracks on pavement surface images," *Journal of Computing in Civil Engineering*, 03 2016.

[29] R.-T. W. Fu-Chen Chen, Mahammad R. Jahanshahi and C. Joffe, "A texture-based video processing methodology using bayesian data fusion for autonomous crack detection on metallic surfaces," *Computer-Aided Civil and Infrastructure Engineering*, 02 2017.

[30] N. G. Prateek Prasanna, Kristin Dana and B. Basily, "Computer-vision based crack detection and analysis," *Proc. SPIE 8345, Sensors and Smart Structures Technologies for Civil, Mechanical, and Aerospace Systems 2012*, 04 2012.

[31] F. D. Tien Sy Nguyen, Stéphane Begote and M. Avila, "Free-form anistropy: A new method for crack detection on pavement surface images," *2011 18th IEEE International Conference on Image Processing (ICIP)*, 09 2011.

[32] Y. L. Xuhang Tong, Jie Guo and Z. Yin, "A new image-based method for concrete bridge bottom crack detection," *2011 International Conference on Image Analysis and Signal Processing*, 10 2011.

[33] Y. Hu and C. xia Zhao, "A novel lbp based methods for pavement crack detection," *Journal of Pattern Recognition Research (JPRR)*, 2010.

[34] A. Mohan and S. Poobal, "Crack detection using image processing: A critical review and analysis," *Alexandria Engineering Journal*, pp. 787–798, 06 2018.

[35] T. Yamaguchi and S. Hashimoto, "Fast crack detection method for large-size concrete surface images using percolation-based image processing," *Machine Vision and Applications*, pp. 797–809, 08 2010.

[36] S. N. Tomoyuki Yamaguchi and S. Hashimoto, "An efficient crack detection method using percolation-based image processing," *2008 3rd IEEE Conference on Industrial Electronics and Applications*, 06 2008.

[37] T. Yamaguchi and S. Hashimoto, "Improved percolation-based method for crack detection in concrete surface images," *2008 19th International Conference on Pattern Recognition*, 12 2008.

[38] T. Yamaguchi and S. Hashimoto, "Image processing based on percolation method," *IEICE Transactions on Information and Systems*, pp. 2044–2052, 07 2006.

[39] K. Jafari-Khouzani and H. Soltanian-Zadeh, "Radon transform orientation estimation for rotation invariant texture analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1004–1008, 06 2005.

[40] N. Aggarwal and W. Karl, "Line detection in images through regularized hough transform," *IEEE Transactions on Image Processing*, pp. 582–591, 03 2006.

[41] T. S. Hitesh Rajput and S. Kar, "Using radon transform to recognize skewed images of vehicular license plates," *Computer*, pp. 59–65, 06 2006.

[42] K. S. Diwikar Tiwary and Anveshraj, "Fingerprint identification using radon transform," *International Refereed Journal of Engineering and Science (IRJES)*, pp. 106–110, 10 2014.

[43] A. K. S. Kim, "Mathematical concepts for image reconstruction in tomography," *Industrial Tomography*, 2015.

[44] W. Burger and M. Burge, *Principles of Digital Image Processing, Advanced Methods.* Springer, 2013.

[45] L. M. Murphy, "Linear feature detection and enhancement in noisy images via the radon transform," *Pattern Recognition Letters*, pp. 279–284, 1986.

[46] C. Zach, T. Pock, and H. Bischof, "A duality based approach for realtime tv-l^1 optical flow," *Pattern Recognition Proc DAGM 2007*, 2007.

[47] K. Bartyzel, "Adaptive kuwahara filter," *SIViP*, pp. 663–670, 2016.