Wide View And Line Filter For Enhanced Image Gradient Computation And Edge Determination

Luke Bagan
University of South Carolina

Follow this and additional works at: https://scholarcommons.sc.edu/etd

Part of the Mechanical Engineering Commons

Recommended Citation

This Open Access Thesis is brought to you by Scholar Commons. It has been accepted for inclusion in Theses and Dissertations by an authorized administrator of Scholar Commons. For more information, please contact digres@mailbox.sc.edu.
WIDE VIEW AND LINE FILTER FOR ENHANCED IMAGE GRADIENT COMPUTATION AND EDGE DETERMINATION

by

Luke Bagan

Bachelor of Science
University of South Carolina, 2021

Submitted in Partial Fulfillment of the Requirements

For the Degree of Master of Science in

Mechanical Engineering

College of Engineering and Computing

University of South Carolina

2023

Accepted by:

Yi Wang, Director of Thesis

Yan Tong, Reader

Ann Vail, Dean of the Graduate School
DEDICATION

To my mom and dad, thank you for supporting me and motivating me to keep pushing farther in my academic journey, to all my friends that kept me going when the path ahead seemed insurmountable; I would not have been able to do this without you.
ACKNOWLEDGEMENTS

Many people have helped me or affected my academic journey while working on towards my thesis over the past two years.

First and foremost, thank you to the Office of Naval Research (contract No. N00014-22-C-1011) and our program officer Dr. Geoff Main for making it possible for me to work in such an exciting industry. Geoff, your guidance, and direction on the project have allowed me to grow exponentially as a student and professional.

The most important person to my journey so far at U of SC has been Dr. Yi Wang. I never thought I would be capable of getting a masters in any engineering discipline nor work on a project this important. I want to thank you for believing in me and trusting my work ethic. My entire career path has been altered for the better thanks to your support and counsel. I also want to thank you for imparting some of your intelligence to me and showing me how I can become the best version of myself.

To the Integer Technologies team, specifically Tristan Kyzer, thank you for working alongside me throughout this process. Your advice during this process has been vital to my success. Additionally, your willingness to help during my time was extremely appreciated.

Finally, I’d like to thank Dr. Yan Tong. Before working on this project, I had no background in image processing and computer vision. Without your ideas, suggestions, and teachings, I would not have gotten to where I am today.
ABSTRACT

Edge determination is a challenging yet crucial step in the object detection process for images. It is the first in a multi-step process, serving as the foundation for all subsequent operations. Its accuracy directly affects the success of any future processing techniques and final detection. The challenge of edge detection is derived from a variety of factors, including noise, image sharpness, orientation, empirical parameters, and computational complexity. Many traditional kernel-based operators excel at tackling one of these problems, but trade off their ability to handle others. For example, the popular Sobel operator uses a horizontal and vertical constant high-pass kernel. The two kernels are used to tackle other operators’ challenges of directional propensity. Theoretically, the combined two kernels can estimate edge orientations with a certain degree of accuracy, however, the effectiveness of its two directional operators are not justified. This means it uses an arbitrary combination of the two kernels. Additionally, its high-pass nature amplifies noise, and the constant architecture of its kernel means it is unable to adapt to the varying light intensities in the photo. This presents large issues in localizing edges and accurately identifying them in the first place. Two new gradient detection kernels based on the two-dimensional high order Taylor Series expansion were proposed and constructed in MATLAB with the goal of tackling most of these problems. The key idea of the first kernel is to use a wide range of the pixels in view to suppress noise, hence improving the gradient intensities of edges. The second kernel builds on the first to leverage its noise suppression benefits while tackling an additional problem of degraded
and low contrast edge boundaries. In principle, it can detect smooth lines in the presence of discontinuities and poor quality. The filter architecture allows for precise gradient calculation, edge detection, and orientation determination to less than one degree of the true value when faced with signal to noise ratios that exceed 0.75.
# TABLE OF CONTENTS

Dedication .......................................................................................................................... iii

Acknowledgements ............................................................................................................ iv

Abstract ................................................................................................................................v

List of Tables .................................................................................................................... vii

List of Figures .................................................................................................................. viii

Chapter 1: Introduction ........................................................................................................1

Chapter 2: Development of the Wide View Filter ...............................................................7

Chapter 3: Development of the Line Filter ........................................................................16

Chapter 4: Wide View and Line Filter Performance Analysis ..........................................22

Chapter 5: Leveraging both Filters for Accurate Edge Angle Detection ...........................33

Chapter 6: Conclusion........................................................................................................49

References ..........................................................................................................................50
LIST OF TABLES

Table 2.1 2D Taylor Series Expansion up to the 5th Order...............................................11
Table 2.2 2D Taylor Series Expansion Shorthand Notation..............................................12
Table 2.3 2D Taylor Series Expansion System of Equations
   Expressed in Matrix Notation................................................................................13
Table 2.4 2D Taylor Expansion System of Equations Expressed as one Formula..........13
Table 2.5 Weighted 2D Taylor Series Expansion In Matrix Notation with Weights ........14
Table 2.6 Weighted 2D Taylor Series Expansion System of
   Equations Expressed as one Formula .................................................................15
Table 3.1 Neighboring Pixel Intensity Vector for the Line Filter......................................20
Table 3.2 Equation to Select Pixel Intensities corresponding to a
   Specific Wide View Filter on the Line Filter.........................................................20
Table 3.3 Simplified Derivative Solution Expression for each
   Wide View Filter on the Line Filter......................................................................21
Table 3.4 Simplified Derivative Solution Equation for the Entire
   Line Filter with Weights Considered....................................................................21
Table 5.1 Numerical comparison of calculated angles at 10-degree angular resolution ...40
Table 5.2 Numerical comparison of calculated angles at 10-degree angular resolution ...42
Table 5.3 Numerical comparison of calculated angles at 10-degree angular resolution ...45
Table 5.4 Numerical comparison of calculated angles at 10-degree angular resolution
   with a 29-pixel filter length and Np = 150............................................................48
LIST OF FIGURES

Figure 2.1 The schematic of the global and the local coordinate system. .........................10

Figure 3.1 Schematic of the local coordinate system and line filter layout with wide view filter layout.............................................................................................................18

Figure 4.1 Synthetic Line Photo Used for Initial Analysis of the Wide View and Line Filter ........................................................................................................................................23

Figure 4.2 Wide View and Line Filter Gradients and Edges Compared with Sobel and Prewitt ....................................................................................................................................24

Figure 4.3 Canny Edges from Synthetic Line Image Using Derivative of Gaussian Method ......................................................................................................................................26

Figure 4.4 New Synthetic Line Photo with Varying SNR Levels used for SNR Analysis with Line Filter ....................................................................................................................................................27

Figure 4.5 Line Filter Gradients and Edges for the SNR Analysis ....................................28

Figure 4.6 Line Filter Gradients and Edges for the SNR Analysis with a Larger Filter Size ...........................................................................................................................................................30

Figure 4.7 Washed out image of Lake Murray ..................................................................31

Figure 4.8 Wide View and Line Filter Results on the Washed-Out Lake Murray Image Compared with Sobel and Prewitt .................................................................................................32

Figure 5.1 Illustration of a cubic spline for one pixel contributing to the histogram containing all pixel angle values for the image ..................................................................................35

Figure 5.2 Postprocessing flow of the angle detection script .............................................37

Figure 5.3 The synthetic line image ....................................................................................39

Figure 5.4 Multi line photo with close lines angle detection results at a 10-degree angular resolution .................................................................................................................................40

Figure 5.5 Image of Dense Synthetic Line Photo ................................................................41
Figure 5.6 Dense line photo with close lines angle detection results at a 10-degree angular resolution

Figure 5.7 Image of Lake Murray Water

Figure 5.8 Lake Murray photo with close lines angle detection results at a 10-degree angular resolution

Figure 5.9 Zoomed in view of Lake Murray Wave Fields

Figure 5.10 Image of Lake Murray Water

Figure 5.11 Multi line photo with close lines and low SNR angle detection results at a 10-degree angular resolution (Np = 150 & Filter Size = 29px)
LIST OF SYMBOLS

\( N_h \)  Number of pixels in the horizontal direction of the image (IE: number of columns)

\( N_v \)  Number of pixels in the vertical direction of the image (IE: number of rows)

\( N_p \)  Number of pixels in the neighborhood of the wide view filter

\( f(x, y) \)  Pixel intensity at image coordinate \( x, y \)

\( f_{xyy}(x, y) \)  2\textsuperscript{nd} degree xy derivative of pixel at image coordinate \( x, y \)

\( A \)  Matrix of 2d Taylor Series \( x \) and \( y \) coefficients

\( z \)  Vector of pixel intensity derivatives

\( b \)  Vector of pixel intensities
CHAPTER 1
INTRODUCTION

1.1 IMAGE PROCESSING

With the rise of automation and artificial intelligence, both computer vision and image processing have become the center of debate on the best and most effective way to tackle the many challenges in the industry. Image processing is the process of taking an image and applying mathematical or computational algorithms to it to extract important features and useful information or alter the original image in a favorable way. Image processing is a broad term that covers a wide set of powerful topics, some of which include image restoration, image segmentation, image compression, image morphology, object recognition, and notably for the present research topic, gradient based edge detection (A.B. da Silva & Gelson 2005). These processes are applicable to a wide variety of industries including medical imaging, autonomous vehicles, manufacturing, astronomy, professional photography, and general computer vision (Jahne 2002).

Recently, the autonomous vehicles industry has been taken over by artificial intelligence and machine learning to detect objects in images as they have displayed a high degree of accuracy with a relatively low degree of necessary computational power. While this favors machine learning in the autonomous vehicle industry and many other fields, image processing based on low-level features is progressing to mitigate computational burden on fieldable systems when it comes to objects detection and ensure its reliability. Machine learning on the other hand has not been able to improve upon its
initial successes for some time and has somewhat stagnated in the field due to a variety of challenges and obstacles (Gupta et al., 2010). Machine learning requires a vast amount of training data in order to train neural networks that work reliably. This presents an inherent problem to machine learning as its performance is restricted to a dataset, and it cannot interpret unfamiliar situations which it was not specifically trained on (Guo et al., 2016). This issue becomes even exacerbated for object detection in the littoral environment being pursued by this research, because the dataset dedicated to littorals is extremely scarce, and the environment is highly cluttered and diversified.

Deep learning is meant to simulate the human vision system when it comes to autonomous vehicles, however it cannot replicate certain intangible human characteristics such as common sense, intuition, explanation, and ability to process foreign situations. This results in a set of instances which deep learning cannot address. These are often referred to as edge cases. On the other hand, image processing does not have this issue. Image processing manipulates the image at face value and defines the features known to be effective. Image processing techniques can ensure that the original image will be analyzed according to its physical representation and that nothing will be missed. Because of this, it can be argued that the autonomous vehicles industry should divert their efforts to an image processing-based pipeline that leverages image characteristics such as edges in order to detect objects.

1.2 GRADIENT CALCULATION AND EDGE DETECTION

Gradient detection is often used as the foundation of edge detection. This is because it addresses the physical nature of an edge in an image. Edges are defined as a
rapid change in pixel intensity and can therefore be located where significant gradients exist in an image (Bhardwaj, S., & Mittal, A. 2012). Once image gradients are calculated from an image, they can be thresholded to produce binary edge maps that contain only the largest magnitude derivatives.

Widely used gradient detection filters aim to address a variety of obstacles. In doing so, they also come with their own distinct disadvantages. The most common gradient-based edge detection methods use Sobel, Prewitt, and Roberts kernels which are typically 3x3 patterns convolved around an image to produce the intensity derivatives of each pixel (Kekre & Saylee 2010). Occasionally, alternate larger variations of these kernels in 5x5, 7x7, and further increasing size are used to improve gradient and edge detection performance. In theory, the large kernels suppress noise better than the 3x3 and can effectively improve the edge map result (Bogdan, Bonchis, & Orhei 2019). However, while easy to apply to an image, the accuracy of these methods depends greatly on the image quality and presence of edges. In addition, the constant and simplistic structure of these kernels limits the performance they can achieve.

The most common issue is the presence of noise in image signals. Most gradient detection methods and edge detection methods are highly susceptible to noise and often amplify undesirable image noise due to the image sharpening properties they possess (and the small range of view used in the filter) (Lin, W., & Wang, J. 2017)( N. Ofir, M. Galun, B. Nadler & R. Basri 2016). Image smoothing filters are common to reduce the effects of noise, however such filters decrease image quality, blur edges and can make low gradient edges undetectable. On top of this, they alter the real intensity of information provided from the original image.
Additional external factors like refraction, poor focus, object obstruction, low light, smooth intensity transitions and others can all affect the quality of an edge boundary. Considering all these influences, traditional edge detectors commonly result in false detection, missed detection, and inaccurate localization. Alternative methods to gradient-based edge detection that delve into second derivative zero crossings, deep learning based approaches, image morphology, and more have been proposed to address these issues (Sun et al., 2022). A systematic approach that can directly operate on the original images and raw pixel intensities and address most of the issues with minimal manual manipulation is still strongly needed.

1.3 ANGLE DETECTION

In gradient-based edge detectors, gradients are only part of the equation when it comes to determining if a pixel is an edge or not. Edge orientation information is typically a critical piece of data to use with gradient magnitudes to make an accurate determination as to whether an edge exists in a specific location. For example, when many edges align with a specific direction in the littoral environment, they may represent the wave crest. Gradient operators will produce a wide variety of derivative magnitudes across an image with thick gradient lines around edges in a photo. This poses a problem of edge pixel localization which can be addressed with extreme precision when using highly accurate gradient angles for corresponding pixels. Edges will always produce the largest derivatives in a local cluster of gradients, and knowing their angles means the maximum gradient and therefore edge location can be extracted from the immediate neighborhood by tracing along the orthogonal direction. Outside of the information it can
Conventional methods for determining angles are insufficient in terms of accuracy. The most accepted method, which is used in Canny’s edge detection process, for obtaining a pixel’s gradient direction is to use the arctangent of the $x$ and $y$ gradients (Bhardwaj, S., & Mittal, A. 2012). This method is unacceptable as it relies on accurate gradient information in two directions, assuming no errors will be produced in those directions to skew the calculation. For example, if an edge lies at a 45 degree angle, its angle will be estimated using the distant gradients obtained in the $x$ and $y$ direction. Any errors in the $x$ and $y$ direction will be translated into the angle calculation. This results in vague estimations of angles which are never true to the actual values. The errors in this method are visible when it comes to determining edges with them and explain why Canny suffers from broken edges and false positives. A better method would search for edges with a finer angular resolution to produce a set of gradients where one dominant gradient at a specific orientation prevails above the rest.

In addition to allowing us to localize edges with high fidelity, gradient angles serve as a low-level image feature that can provide many benefits in a variety of image processing techniques. Pixel orientation can provide insight into objects in the photo and characterize their immediate movement. In addition, it can be used as a piece of data to group similar edges in the photo together. One straightforward and practical use for angles is to group wave edges together from an aquatic scene so that the water can be
analyzed. Another way angles can be useful is to separate edges in a scene that may have been incorrectly linked during the edge determination process. Using highly accurate angles in low level features improves the accuracy of all the associated processes.

1.4 RESEARCH OBJECTIVE

The most popular autonomous driving systems on the market today are built with AI and deep learning at the core. These models have proven time and again to be unreliable and dangerous for a multitude of reasons. The main problem facing this industry is the need for vast amounts of training data required to train neural networks which are intended to replicate human visual and decision-making systems. The truth is, however, there will never be enough training data to address all the edge cases that an AI-powered autonomous navigation system may encounter. Physics and image processing-based methods will be necessary to ensure that edge cases will not pose an issue to these systems and to reach level five full driving automation.

The objective of this research is to build upon the foundational concepts necessary for a physics-based autonomous driving suite. This includes developing all the necessary components for a highly reliable edge detection process, which can be taken in the future and used as the basis for an object detection pipeline. The components can be boiled down to three principal ideas. The first is a highly accurate gradient calculation kernel, which is robust to low signal-to-noise ratio environments, can bridge small breaks/occlusions in the image, and can recognize gradients produced from exceedingly small objects. In addition, a similar kernel will be necessary which is more capable in detecting line edges rather than point edges. Both kernels should be adaptable to address a wide
variety of image scenarios. The final necessary component would be a gradient orientation detection system, which can determine the angle of an edge within one degree of its true value. Chapter 2 covers the development of the wide view filter, which is a derivative calculation kernel built using the ideas of the two-dimensional Taylor Series expansion and serves to address the issues of low SNR environments and small object recognition. Chapter 3 discusses the development of the line filter, which is based on the wide view filter and excels in the line detection and small occlusion bridging. Chapter 4 compares the results using these filters with conventional edge detection methods like Canny and Sobel. Finally, Chapter 5 outlines a novel angle detection process, which can meet the strict accuracy limitations set in place.
CHAPTER 2

DEVELOPMENT OF THE WIDE VIEW FILTER

The most critical step in building a reliable object detection system is establishing where true edges exist in an image. Doing so requires a highly robust derivative calculation kernel. The biggest issue traditional kernels face is signal noise, which interferes with the ability to obtain the true derivative value at a pixel. Typically, noise poses an extreme problem in obtaining spatial derivatives due to the form of gradient calculation kernels. Gradient kernels resemble image sharpening filters, which inherently increase the prominence of noise. The wide view filter is built to tackle this problem in this thesis research first by incorporating significantly more image information into the calculation (Alpert et al. 2010). This is consistent with the human vision system which is generally unaffected by noise in its wide view (N. Ofir, M. Galun, B. Nadler & R. Basri 2016). In addition to considering a new and larger filter framework, the wide view aims to attain industry-leading gradient accuracy by changing the method by which image derivatives are calculated and by providing a malleable architecture.

2.1 WIDE VIEW FILTER ARCHITECTURE

The wide view filter operates based on the principles of the two-dimensional Taylor Series expansion, and the mathematical foundation for calculating the high-order
derivates using a least-squares approach is provided in detail in section 2.2. Before that, it is important to understand how the wide view filter is constructed and how its architecture differs from common gradient operators such as the Sobel operator.

The most common Sobel operators are three-by-three kernels centered around the indexed pixel and calculate the horizontal and vertical gradients in an image that are combined to create one gradient magnitude matrix. These are very susceptible to noise due to their constant structure and small size. The proposed wide view filter, on the other hand, depends on the number of pixels, $N_p$, around an indexed pixel in the calculation. The wide view filter will automatically select the closest $N_p$ pixels to the index, which generally yields a circular structure. Importantly, $N_p$ will contain upwards of 100 neighboring pixels, meaning it is significantly more robust to signal noise when compared to the Sobel operator and the nine neighbors it uses.

The nearest neighbors of the indexed pixel define a local coordinate system denoted as $(x, y)$, which is contained in the global image coordinate system denoted as $(X, Y)$ where $X = 1, 2, \ldots, N_h$ and $Y = 1, 2, \ldots, N_v$, where $N_h$ and $N_v$ are, respectively, the number of pixels along the horizontal and the vertical direction. The local coordinates $(x, y)$ are, in turn, along the tangential and the normal direction of the local coordinate system used for computing the high-order derivatives. Figure 2.1 shows an example of how the wide view filter looks situated in an image at the pixel center $(X_0, Y_0)$. This pixel center represents a few significant locations for the gradient calculation. It is the location where the tangential and the normal derivatives need to be computed as well as the origin of the local coordinate system. The local coordinates containing $N_p$ neighbor pixels are captured in the light-yellow region. Their coordinate axes can be rotated to obtain
gradients in the normal and tangential directions in many intervals, allowing for highly accurate derivative calculations at many different angular positions. The tangential direction will have an angle offset $\theta$ between 0 and $2\pi$ with respect to the horizontal direction of the image ($X$).

![Figure 2.1 The schematic of the global and the local coordinate system.]

2.2 WIDE VIEW FILTER ALGORITHM

The gradients for the indexed pixel ($X_0$, $Y_0$), in the global coordinate system, will be calculated using a Taylor series expansion which is given in the following equation.
\[ f(x, y) = f(0, 0) + f_x(0, 0)x + f_y(0, 0)y + \frac{f_{xx}(0, 0)x^2}{2} + \frac{f_{yy}(0, 0)y^2}{2} \]
\[ + f_{xy}(0, 0)xy + \frac{f_{x^3}(0, 0)x^3}{6} + \frac{f_{y^3}(0, 0)y^3}{6} + \frac{f_{x^2y}(0, 0)x^2y}{2} \]
\[ + \frac{f_{xy^2}(0, 0)xy^2}{2} + \frac{f_{x^4}(0, 0)x^4}{24} + \frac{f_{y^4}(0, 0)y^4}{24} + \frac{f_{x^2y^2}(0, 0)x^2y^2}{6} \]
\[ + \frac{f_{xxy}(0, 0)xy^3}{6} + \frac{f_{xx^2y}(0, 0)x^2y^2}{4} + \frac{f_{xy^5}(0, 0)y^5}{120} + \frac{f_{x^4y}(0, 0)x^4y}{24} + \frac{f_{xy^4}(0, 0)xy^4}{24} \]
\[ + \frac{f_{x^2y^3}(0, 0)x^2y^3}{12} + \frac{f_{x^3y^2}(0, 0)x^3y^2}{12} \]

In this expression, \( f(x, y) \) corresponds to the pixel intensity at any of the pixels in the local coordinate system, while \( x \) and \( y \) correspond to the offset of that respective pixel from the local origin. For the sake of concise expression derivatives typically denoted in the following convention, \( f_{xxyy} \), are instead represented \( f_{x^2y^2} \). Assuming there are \( N_p \) pixels closest to \((X_0, Y_0)\) enclosed in the yellow circular region in Figure 2.1, which are denoted as \((X_i, Y_i)\) and \((x_i, y_i)\) and \( i = 1, 2, ..., N_p \), then the Taylor series expansion at the origin of the local coordinate system, i.e., \((X_0, Y_0)\) in the global coordinate, is written using the following format.
Table 2.2 2D Taylor Series Expansion Shorthand Notation

\[
f(x_i, y_i) = f^0 + f_x^0 x + f_y^0 y + \frac{f_{xx}^0 x^2}{2} + \frac{f_{yy}^0 y^2}{2} + f_{xy}^0 xy + \frac{f_{xx}^0 x^3}{6} + \frac{f_{yy}^0 y^3}{6} + \frac{f_{x^2y^2}^0 x^2 y^2}{2} + \frac{f_{x^2y^2}^0 x^2 y^2}{2} + \frac{f_{x^4}^0 x^4}{24} + \frac{f_{y^4}^0 y^4}{24} + \frac{f_{x^3y^3}^0 x^3 y^3}{6} + \frac{f_{x^3y^3}^0 x^3 y^3}{6} + \frac{f_{x^2y^3x^2y^2}^0 x^2 y^2}{4} + \frac{f_{x^2y^3x^2y^2}^0 x^2 y^2}{4} + \frac{f_{x^5}^0 x^5}{120} + \frac{f_{y^5}^0 y^5}{120} + \frac{f_{x^4y^4}^0 x^4 y^4}{24} + \frac{f_{x^4y^4}^0 x^4 y^4}{24} + \frac{f_{x^3y^3x^2y^3}^0 x^2 y^3}{12} + \frac{f_{x^3y^3x^2y^3}^0 x^2 y^3}{12}
\]  

(2)

In this case, \(f^0\) is the shorthand notation of \(f(0, 0)\). In Eq. (2), all terms of \(x_i\) and \(y_i\) are known since they only represent the relative positions between the neighboring pixels and the target indexed pixel. Considering all the known terms, all that is left to calculate are the intensity value \(f^0\) and the derivative coefficients. Although it may seem arbitrary to calculate the intensity value of the origin pixel, \(f^0\), since that should be known, the origin could be corrupted by noise. In a case where the origin is corrupted, it serves better to estimate the intensity, \(f^0\), of the origin pixel. Another variation of the wide view filter can easily be implemented which considers the intensity value of the origin as known. This will slightly alter the Eq. (3). Th origin intensity will always be multiplied by the column of ones in the \(A\) matrix which means it will always carry through as itself in the summation steps of matrix multiplication. It can therefore be removed from the \(z\) vector and subtraced from the \(b\) vector while the column of ones are removed from the \(A\) matrix if it is desired to operate the wide view filter in such a way.

Moving on from the origin, the derivative coefficients must be calculated. Specifically, when the 4\textsuperscript{th}-order derivative is the highest order, there are 15 unknowns that are the coefficients of all the derivatives. If the highest-order derivative is the 5\textsuperscript{th}, the
number of unknowns is 21. To obtain the 15 unknowns in the former system, we need at least 15 neighboring pixels, i.e., \((x_i, y_i)\) and \(i = 1, 2, ..., N_p\) and \(N_p = 15\), to generate a system of equations. In general, we will consider more than the minimum number of pixels necessary to minimize bias introduced by a few erroneous pixel values. In any case, this results in \(N_p\) equations, which can be cast into a compact matrix form represented by the following equation.

\[
\begin{bmatrix}
1 & x_1 & y_1 & \cdots & 1/4 x_1^2 y_1^2 & 1/6 x_1 y_1^3 \\
1 & x_2 & y_2 & \cdots & 1/4 x_2^2 y_2^2 & 1/6 x_2 y_2^3 \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
1 & x_{N_p} & y_{N_p} & \cdots & 1/4 x_{N_p}^2 y_{N_p}^2 & 1/6 x_{N_p} y_{N_p}^3
\end{bmatrix}
\begin{bmatrix}
f_0^0 \\
f_0^x \\
f_0^y \\
\vdots \\
f_{x^2 y^2}^0 \\
f_{x y^3}^0
\end{bmatrix} =
\begin{bmatrix}
f(x_1, y_1) \\
f(x_2, y_2) \\
\vdots \\
f(x_{N_p}, y_{N_p})
\end{bmatrix}
\]

Casting the system of equations into a matrix form allows all the derivatives to be computed in one shot using the format below.

\[
z = (A^T A)^{-1} A^T b = A^* b
\]

The superscript * represents the matrix pseudo-inverse. As shown by Eq. (4), in order to apply the precomputed filter coefficients to a new image, the pseudo-inverse matrix of A, i.e., \(A^*\) needs to be saved. In addition, since only the derivatives along the tangential and the normal directions are needed, i.e., the mixed derivatives in Eq. (2) are not needed, only the row entries corresponding to these desired derivatives are stored,
further reducing the storage. This also means that no cross-derivatives will be considered for the indexed pixel. Only normal and tangential derivatives will be computed, which will result in two times the highest order of derivative considered in the equation being calculated for the central pixel.

After the derivatives are calculated for the wide view filter in a specific orientation, the local coordinate axis can be rotated, and the process repeated. Doing so will change the A matrix in Eq. (3) as the relative positions from the local axes will change. This can be done for any arbitrary angle with a large number of $N_p$ and will result in the ability to determine a precise derivative magnitude for any edge, no matter its position or rotation in the image.

The neighboring pixels may have differential contributions to the calculation of the numerical derivatives at the origin of the local coordinate. Specifically, the pixels far away from the origin may have less important contributions, and the influences of all pixels should be weighted according to their difference from the origin. Mathematically, this is equivalent to multiplying a diagonal weight matrix on both sides of Eq. (3), i.e.,

$$
\begin{bmatrix}
\sqrt{w_1} & \sqrt{w_2} & \vdots & \sqrt{w_{N_p}} \\
\sqrt{w_1} & \sqrt{w_2} & \vdots & \sqrt{w_{N_p}} \\
\vdots & \vdots & \ddots & \vdots \\
\sqrt{w_1} & \sqrt{w_2} & \vdots & \sqrt{w_{N_p}} \\
\end{bmatrix}
\begin{bmatrix}
x_1 \\ x_2 \\ \vdots \\ x_{N_p} \\
y_1 \\ y_2 \\ \vdots \\ y_{N_p} \\
\end{bmatrix}
\begin{bmatrix}
\frac{x_1 y_1}{4} \\ \frac{x_2 y_2}{4} \\ \frac{x_3 y_3}{6} \\ \frac{x_4 y_4}{6} \\
\end{bmatrix}
= 
\begin{bmatrix}
f_0 \\ f_x \\ f_y \\ f_{x^2} \\ f_{y^2} \\ f_{x^2 y^2} \\ \vdots \\ \end{bmatrix}
$$

$$
\begin{bmatrix}
\sqrt{w_1} & \sqrt{w_2} & \vdots & \sqrt{w_{N_p}} \\
\sqrt{w_1} & \sqrt{w_2} & \vdots & \sqrt{w_{N_p}} \\
\vdots & \vdots & \ddots & \vdots \\
\sqrt{w_1} & \sqrt{w_2} & \vdots & \sqrt{w_{N_p}} \\
\end{bmatrix}
\begin{bmatrix}
f(x_1, y_1) \\ f(x_2, y_2) \\ \vdots \\ f(x_{N_p}, y_{N_p}) \\
\end{bmatrix}
= 
\begin{bmatrix}
f(x_1, y_1) \\ f(x_2, y_2) \\ \vdots \\ f(x_{N_p}, y_{N_p}) \\
\end{bmatrix}
$$

---

**Table 2.5 Weighted 2D Taylor Series Expansion In Matrix Notation with Weights**

<table>
<thead>
<tr>
<th>$\sqrt{w_1}$</th>
<th>$\sqrt{w_2}$</th>
<th>$\vdots$</th>
<th>$\sqrt{w_{N_p}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$1$</td>
<td>$x_1$</td>
<td>$y_1$</td>
<td>$\frac{1}{4} x_1 y_1^2$</td>
</tr>
<tr>
<td>$1$</td>
<td>$x_2$</td>
<td>$y_2$</td>
<td>$\frac{1}{4} x_2 y_2^2$</td>
</tr>
<tr>
<td>\vdots</td>
<td>\vdots</td>
<td>\vdots</td>
<td>\vdots</td>
</tr>
<tr>
<td>$1$</td>
<td>$x_{N_p}$</td>
<td>$y_{N_p}$</td>
<td>$\frac{1}{4} x_{N_p} y_{N_p}^2$</td>
</tr>
</tbody>
</table>

$\begin{bmatrix}
f_0 \\ f_x \\ f_y \\ f_{x^2} \\ f_{y^2} \\ f_{x^2 y^2} \\ \vdots \\ \end{bmatrix}$
Defining $\tilde{A} = \sqrt{W}A$ and $\tilde{b} = \sqrt{W}b$, we can use the same least-squares solution and pseudo-inverse in Eq. (4) to obtain the numerical derivatives, i.e.,

$$z = (\tilde{A}^T \tilde{A})^{-1} \tilde{A}^T \sqrt{W}b = \tilde{A}^* \sqrt{W}b$$  \hspace{1cm} (6)

$\tilde{A}^* \sqrt{W}$ will be the coefficients of the filter to be saved for future use. Hereafter, this one considering the weighted contributions from different neighboring pixels is named the weighted wide view filter.
CHAPTER 3

DEVELOPMENT OF THE LINE FILTER

While the wide view filter excels in gradient calculation in low SNR environments and offers a large range of benefits, it has one issue, i.e., the presence of low contrast boundaries along an edge. As expected, the contents and light intensities of an image vary across the image domain. This results in inconsistent broken boundaries and potentially weak areas along an edge that can result in missed detection.

Development started on the line filter with the idea of further improving the capabilities of the wide view filter, specifically the smooth detection of lines, to further improve the ability to detect gradients in the presence of extreme noise, and to produce a better orientation indication of any present edges. Because of this, the line filter is built on the same foundational principles as the wide view filter. The principal idea is to use many surrounding pixel values and a high-order (up to the 4\textsuperscript{th}-order) Taylor expansion to compute all the derivatives along both the normal and the tangential directions at a target pixel. The line filter can improve upon the wide view filter by determining numerical derivatives at the target pixel using the weighted derivative values of multiple pixels along the line filter oriented in a specific direction. This way, the line filter can fill any low-gradient areas along a line by using the high-gradient information along this line. In addition, by lengthening the filter, more image information is included in the neighborhood to achieve a higher fidelity gradient calculation under non-ideal noise conditions.
3.1 LINE FILTER ARCHITECTURE

The line filter can be considered a string of wide view filters. One wide view filter will be centered on the indexed pixel with a user-defined number of filters on both the left and right of the index. Figure 3.1 illustrates the concept of the line filter, while its important aspects are described as follows:

1. A user-defined number of pixels (in yellow) are indexed along a line centered on a target pixel (in red). Hereafter, the red target pixel is termed the origin pixel as it denotes the origin of the local coordinate system \((x, y)\). In most cases, the pixels on the line (in yellow) will not fall on real pixels and are therefore referred to as virtual pixels. The distance between virtual pixels on the line is spaced evenly and kept constant as “1” while the orientation of the line can be arbitrary. These virtual pixels pose no problem to computing derivatives because only their precise locations instead of their pixel intensity values are needed in the high-order Taylor expansion. The location of each virtual pixel along the line will be used to find its nearest neighbor pixels to run the wide view filter centered around this virtual pixel. The process outlined from the wide view filter can then be executed at each virtual pixel to yield the numerical derivatives along the line.

2. The wide view filter calculates one value of each derivative for every virtual pixel along the line, which means the line filter will contain multiple virtual pixels, all with multiple corresponding derivative values. The derivatives at the origin of the line filter are then calculated as the weighted average of those at the virtual line pixels. In this method, a set of weights can be applied to the derivatives at each virtual pixel. The origin pixel will hold the largest weight, while the pixels towards the end of each arm have very
small weights, resembling a normal distribution. Alternatively, the weights can be disregarded for all virtual pixels to improve the connectivity of a subtle edge.

3. The line filter described in (1) and (2) will be applied at many different orientations, 
\[ \theta = \frac{i \cdot 2\pi}{n_s} \], where \( n_s \) is a user-defined number of lines to resolve the circumferential direction and \( i = 1, \ldots, n_s \) is the index. It is expected that when the line filter aligns with the edges in the image, the numerical derivatives will show the strongest values along the tangential direction. Once again, this demonstrates the capability to extract the exact orientation of an edge in a photo. In this case, the line filter should be even more capable of predicting angular information going forward as it contains more derivatives of an ensemble of pixels over the wide view filter, which can capture the original edge rotation.

![Figure 3.1 Schematic of the local coordinate system and line filter layout with wide view filter layout](image-url)
3.2 LINE FILTER ALGORITHM

The line filter is implemented in MATLAB, and the numerical algorithms and procedures need to be delicately formulated to improve computing efficiency and to generate a single set of line filter coefficients (instead of iterative applications of multiple wide view filters), which is extremely important for practical utilization of the present line filter. It is expected that the line filter and wide view filters will be able to be parallelized to run efficiently on high-performance equipment. The numerical procedure is summarized as follows:

1. The half width of the line filter \( m \) is defined, and thus, the total length of the line is \( 2m + 1 \) (also the number of pixels along the line). This indicates how many wide view filters are necessary to construct the wide view filter.

2. For each virtual pixel and the origin pixel, its neighbor pixels used in the wide view filter will be determined using the relative position between them and the origin pixel. This is because the origin pixel is a “real” pixel, and the relative positions between it and the neighbors of all virtual pixels are integers. This yields a set of the neighbor pixels (used in the wide view filter) for each virtual pixel, i.e., \( \Delta_j = \{(\Delta x, \Delta y)_1, (\Delta x, \Delta y)_2, \ldots, (\Delta x, \Delta y)_{N_p}\} \), where \( j \) is the index of the virtual pixel, and \( N_p \) is the number of neighbor pixels for the \( j^{th} \) virtual pixel.

3. In order to create a single set of the line filter coefficients, we string together the neighbor pixels of all virtual and origin pixels in the form of a union, i.e., \( \omega = \{\Delta_1, \Delta_2, \ldots, \Delta_{2m+1}\} \). The total number of pixels in \( \omega \) is \( N \). This creates a single set that defines the range of the neighbor pixels for the entire line filter, whose intensity values will be used for computing derivates of the line filter.
4. Then, the intensity values of all pixels surrounding the line filter are given by

\[ F = \begin{bmatrix} f(X_1, X_1) \\ \vdots \\ f(X_i, Y_i) \\ \vdots \\ f(X_N, Y_N) \end{bmatrix} \]  \hspace{1cm} (1)

where \((X_i, Y_i) = (X_0, Y_0) + \omega(i), X_0\) and \(Y_0\) are the global coordinate of the origin pixel of the line filter, and \(X_i\) and \(Y_i\) are the global coordinate of the \(i^{th}\) neighbor pixel in \(\omega\).

Thus, the intensity values of neighbor pixels at the \(j^{th}\) virtual pixel can be easily retrieved by

\[ F_j = P_j F \]  \hspace{1cm} (2)

where \(F_j\) has a dimension of \(N_p\), the number of neighbor pixels around the \(j^{th}\) virtual pixel on the line filter; and \(P_j\) is a selection matrix with a dimension of \(N_p \times N_p\) and selects the neighbor pixels around the \(j^{th}\) virtual pixel out of all the neighbor pixels of the entire line filter. Note that \(P_j\) is a submatrix of the identity matrix with a dimension of \(N_p \times N_p\).

5. The original equation used to calculate derivatives from a wide view filter can be slightly altered using the principle of the selection matrix in Eq. (2). The numerical derivatives \(z_j\) computed by the wide view filter around the \(j^{th}\) virtual pixel is given by:
Table 3.3 Simplified Derivative Solution Expression for each Wide View Filter on the Line Filter

| \( z_j = A_j^*F_j = A_j^*P_jF \) |

where \( A_j^* \) is the coefficients of the wide view filter at the \( j^{th} \) virtual pixel. Note that \( A_j^* \) is not constant for all virtual pixels because the relative positions between each individual virtual pixel and its neighbor pixels vary along the line. Therefore, \( A_j^* \) depends on \( j \).

6. With all \( A_j^* \) available, the filter coefficient for the entire line filter can be written as

Table 3.4 Simplified Derivative Solution Equation for the Entire Line Filter with Weights Considered

| \( z = w_1z_1 + \cdots w_jz_j + \cdots w_{2m+1}z_{2m+1} = \sum_{j=1}^{2m+1} w_jA_j^*F_j \) |

\[ = \left( \sum_{j=1}^{2m+1} w_jA_j^* \right) F = A_lF \]

where \( A_l \) is the single set of coefficients for the entire line filter and varies with the orientation of the line filter and the number of the virtual pixels along the line.
CHAPTER 4
WIDE VIEW AND LINE FILTER PERFORMANCE ANALYSIS

It is hard to quantitatively evaluate the performance of gradient filters and edge
detection processes as there are no ground truth values to compare to when analyzing
complex images. Because of this, the performance of the Wide View Filter and Line
Filter were qualitatively analyzed by performing gradient and edge comparisons to
common methods. In these tests, the most important evaluation criteria are gradient
smoothness, true edge gradient contrast, angular invariance, noise suppression, edge
localization, and edge connectivity. All these metrics can be assessed with visual results.

To properly cover all the evaluation metrics, three tests were developed and
carried out. First, a synthetic image was generated with a constant signal-to-noise ratio
and various lines at different angles to provide a baseline of the Wide View Filter and
Line Filter’s capabilities. This photo gives us a good baseline to evaluate the gradient
calculation accuracies since all its parameters are known and adjustable. Because the
parameters are manipulable, the synthetic photo can provide a stress test to determine the
limits of the filter’s abilities in the second test. A real-world photo is a sensor
interpretation of the environment in front of it. Such a photo is affected by physical
properties like depth, perspective, movement, and others. The Wide View and Line
Filters have been tested on such photos in the third test to determine their efficacy in a
real-world situation.
4.1 SYNTHETIC IMAGE TESTING

The synthetic line photo consists of a multi-line enclosed shape on a white background with salt and pepper noise tuned to a signal-to-noise ratio of two. An SNR of two serves as a challenging yet realistic scenario that the gradient filters may need to overcome in field operation. In addition, the variably oriented lines in the figure serve as tests to judge the angular fidelity of the filters. It is well known that commonly used gradient filters like Sobel and Prewitt lack significant accuracy in determining diagonal line gradients because they only estimate the magnitude using the gradients calculated at the local $x$ and $y$ positions. Results processing the synthetic line photo with both the wide view and line filters, as well as Sobel and Prewitt kernels are provided in Figure 4.2 in a tabular format for easy comparison.

A few observations can be made from these results; however, it is important to understand what they mean before drawing any conclusions. Any lines and edges parallel to the $x$ axes of the arbitrarily oriented filter will be undetected, while the lines perpendicular to the axes, i.e., the normal direction of the lines, will have the strongest
detection. Second, lines in between the parallel and perpendicular directions of the axis will vary in intensity according to how close or far away its direction is from the normal direction. The closer the line is to the normal direction, the larger its derivative value will be. This specifically is why traditional gradient detection methods fall short. Using only $x$ and $y$ kernels can only guarantee accurate detection along those two directions. Anything in between them will be an assumption made from the local $x$ and $y$ gradients produced from the respective kernels. An edge in between the $x$ and $y$ gradients will produce smaller gradient values from those kernels and in turn be represented by an inaccurate smaller magnitude. This is one major reason the wide view and line filter can produce superior results. These filters can be oriented at any number of angles and produce gradients at a very fine angular resolution, meaning they leave nothing up to assumption.

<table>
<thead>
<tr>
<th></th>
<th>Wide View Filter</th>
<th>Line Filter</th>
<th>Sobel</th>
<th>Prewitt</th>
</tr>
</thead>
<tbody>
<tr>
<td>X Dir. Gradients</td>
<td><img src="image" alt="Wide View Filter X" /></td>
<td><img src="image" alt="Line Filter X" /></td>
<td><img src="image" alt="Sobel X" /></td>
<td><img src="image" alt="Prewitt X" /></td>
</tr>
<tr>
<td>Y Dir. Gradients</td>
<td><img src="image" alt="Wide View Filter Y" /></td>
<td><img src="image" alt="Line Filter Y" /></td>
<td><img src="image" alt="Sobel Y" /></td>
<td><img src="image" alt="Prewitt Y" /></td>
</tr>
<tr>
<td>Edges</td>
<td><img src="image" alt="Wide View Filter Edges" /></td>
<td><img src="image" alt="Line Filter Edges" /></td>
<td><img src="image" alt="Sobel Edges" /></td>
<td><img src="image" alt="Prewitt Edges" /></td>
</tr>
</tbody>
</table>

*Figure 4.2 Wide View and Line Filter Gradients and Edges Compared with Sobel and Prewitt*
Both the Prewitt and Sobel methods can detect gradients along the line in the photo. However, they are not very capable of separating those gradient values from the background noise. This conclusion can be made by comparing the relative intensities of those gradients to the salt and pepper-like noise surrounding them. This is especially true for the Prewitt method, which produces very muted values in general. Due to this phenomenon, the edge maps produced by their respective gradients maintain an intolerable level of false positives and broken edges. As is visible, all the true edges are very interrupted and lack any degrees of connectivity, which is paramount in any good edge map. On the other hand, the wide view and line filters are very capable. The gradients show strong contrast to the noise levels in the background, and because of that, their edge maps show vast improvement over the competition. The wide-view filter-produced edges show much better connectivity and noise suppression immediately, although it is noted that some false positives and breakages do persist. However, the line filter further improves the performance of the wide view filter. When properly aligned with an edge, the line filter can accurately detect a gradient even in the presence of strong noise. This means that no matter the orientation of edges in the original picture, the line filter will produce smooth and strong gradients, resulting in a noise-free/minimal and completely connected edge map.

The Canny edge determination process is also highly regarded in traditional edge detection processes and is therefore important to consider in comparison with our filters. Canny filter does not produce gradients in any novel way and instead includes a process for obtaining edges from a gradient map. Because of this, Canny filter is usually used
with the Derivative of the Gaussian method to define gradient intensities. The edge map produced from this method can be seen below in Figure 4.3 and shows clean and connected edges on the synthetic line object itself. Even though it does produce very good edges on the object, the result is completely degraded by noise.

4.2 SYNTHETIC IMAGE SIGNAL TO NOISE ANALYSIS

One of the most desirable features of the wide view and line filters is their ability to suppress noise. It is therefore important to gauge the limits of the filter’s capabilities regarding gradient detection in extreme noise. Another synthetic line photo was generated for this purpose so that the known SNR values could be changed and then used with filters. The new synthetic line photo is given in Figure 4.4 below. The set starts with the lowest SNR at two, which is the same as the synthetic line photo used in section 4.1. We know that the line filter is already capable of handling such a noise level, which is why it has been chosen as the starting point for this test. The set then progressively degrades all the way down to an SNR level of 0.5. This represents a worst-case scenario as the lines
are significantly degraded by noise, they are hard to see even with normal human vision. It also serves to evaluate how changing parameters of the filter architecture can alter the gradient results and adapt to harsh conditions.

Using a base configuration of the line filter, decent results can immediately be obtained. These can be viewed in Figure 4.5. The gradients and edges produced from the multi-line photo with an SNR of two are extremely accurate. The final edge map is extremely clean, containing no false positives and no line breakages. As the SNR
decreases, the quality of gradients and edge also does. Edges are still visible at an SNR of one, however they are broken and very messy due to the noise. This result is still promising, however as an SNR of one is hard for detection in the real world. It represents a situation, in which the image noise would be equal to the image signal from the sensor. This is highly unlikely, however not impossible, and should still be addressed. As expected, further decreasing the SNR results in very messy edges with almost undetectable true edges in the final edge map.

<table>
<thead>
<tr>
<th></th>
<th>X Dir. Gradients</th>
<th>Y Dir. Gradients</th>
<th>Edges</th>
</tr>
</thead>
<tbody>
<tr>
<td>SNR = 2</td>
<td><img src="image1" alt="X Dir. Gradients" /></td>
<td><img src="image2" alt="Y Dir. Gradients" /></td>
<td><img src="image3" alt="Edges" /></td>
</tr>
<tr>
<td>SNR = 1</td>
<td><img src="image1" alt="X Dir. Gradients" /></td>
<td><img src="image2" alt="Y Dir. Gradients" /></td>
<td><img src="image3" alt="Edges" /></td>
</tr>
<tr>
<td>SNR = 0.75</td>
<td><img src="image1" alt="X Dir. Gradients" /></td>
<td><img src="image2" alt="Y Dir. Gradients" /></td>
<td><img src="image3" alt="Edges" /></td>
</tr>
</tbody>
</table>
The gradient and edge fidelity from Figure 4.5 can be improved by increasing the filter size. Doubling the length of the line filter and the number of neighborhood pixels used previously allows us to obtain extremely robust results. This yields a line filter with a length of 29 pixels, and 250 neighborhood pixels per wide view filter which is very capable of handling an SNR level of one. It can even obtain smooth and accurate edges at an SNR of 0.75. As the SNR falls to 0.5, the edge map degrades significantly, though it can capture the general geometry of the multi-line object that it was not able to do before. These are clear in Figure 4.6. Doubling the line filter size does not significantly increase the computational cost, and more importantly, it can be parallelized, so it can be assumed that further increasing the filter size will allow us to further enhance edge detection at outrageous SNR levels. It is also extremely important to note that these results can be obtained without the need to use any smoothing filters. All gradients and edges are obtained using the original photo information to preserve the image integrity and use it at face value. Sobel, Prewitt, and Canny would all need to apply some sort of smoothing filter before they could handle images with an SNR level of 2. The line filter, on the other hand, can handle significantly worse scenarios without the smoothing filter.
4.3 REAL IMAGE TESTING

The Wide View and Line Filter prove to be very capable as evidenced from the synthetic photo results and SNR analysis. However, these filters are designed with the idea of real world deployability in mind, and therefore, must be tested on representative data. One of the most challenging environments facing the image processing industry to date is the aquatic environment for several reasons. First and foremost, water is an ever-changing environment, which impacts the stability of the scene and any objects present in it. In addition, physical phenomena in this environment can significantly degrade photos, such as specular reflection, high dynamic range, and weather-related events.
Processing a photo of an aquatic environment means the filters can be stress tested and properly assessed. For these reasons and more, it was desired to test the capabilities of both filters in a similar environment. A highly washed-out Lake Murray scene as shown in Figure 4.7 was chosen. This photo provides a few objects to detect while simultaneously presenting an extreme lighting scenario across the length of the photo. The lighting variation creates harsh reflection on the left-hand side of the photo and almost washes out the buoy in that region. The lighting transition moving to the right-hand side of the photo creates a range of gradient data. These characteristics would normally challenge gradient-based filters; however, the wide view filter and line filter prove to be quite capable.

The washed-out Lake Murray photo was processed by both the wide view filter, and line filter and compared with the Sobel and Prewitt gradient detection methods. The results are shown in Figure 4.7. From this, a few conclusions can be drawn. First and foremost, the wide view and line filters are both capable of small object detection. Even in the sub-optimal lighting conditions of the photo, both buoys can be easily detected. It can be asserted that the wide view filter is better than the line filter in this respect because the wide view filter is smaller, and therefore, introduces less pixel information unrelated to the object in the gradient calculation. When observing the x direction gradient results between the two, it can be seen that the line filter does not produce visible gradient information. That is due to the relatively small size of the buoy edge (i.e., the top of the buoy) in comparison to the filter size in the x direction. This, however, comes with a
drawback in that it is more susceptible to noise, which is visible when comparing the two filter’s gradient and edge results in Figure. Both the line filter and wide view filter however, show significant noise suppression in their results as compared to Sobel and Prewitt. This is evident that fewer broken edges are detected in the water body. Finally, both in-house filters show the ability to emphasize line edges and repair small breaks since they look over a larger area in the photo. Small interferences in long smooth lines like those of the horizon do not cause difficulty to either the wide view or line filter.

<table>
<thead>
<tr>
<th></th>
<th>Wide View Filter</th>
<th>Line Filter</th>
<th>Sobel</th>
<th>Prewitt</th>
</tr>
</thead>
<tbody>
<tr>
<td>X Dir. Gradients</td>
<td><img src="image1" alt="Image" /></td>
<td><img src="image2" alt="Image" /></td>
<td><img src="image3" alt="Image" /></td>
<td><img src="image4" alt="Image" /></td>
</tr>
<tr>
<td>Y Dir. Gradients</td>
<td><img src="image5" alt="Image" /></td>
<td><img src="image6" alt="Image" /></td>
<td><img src="image7" alt="Image" /></td>
<td><img src="image8" alt="Image" /></td>
</tr>
<tr>
<td>Edges</td>
<td><img src="image9" alt="Image" /></td>
<td><img src="image10" alt="Image" /></td>
<td><img src="image11" alt="Image" /></td>
<td><img src="image12" alt="Image" /></td>
</tr>
</tbody>
</table>

*Figure 4.8 Wide View and Line Filter Results on the Washed-Out Lake Murray Image Compared with Sobel and Prewitt*
CHAPTER 5
LEVERAGING BOTH FILTERS FOR ACCURATE EDGE ANGLE DETECTION

Traditional gradient calculation kernels only consider gradients in the $x$ and $y$ direction. While these will produce generally accurate gradients in the $x$ and $y$ directions, anything in between must be estimated. Typically, this is done using an arctan of the calculated $x$ and $y$ gradients, which, however, is extremely unreliable. If an image consists of one line at a 45-degree angle, taking the arctan of the $x$ and $y$ gradients will suffice. However, there are many factors that will produce a highly inaccurate angle estimation when this approach is used. For one, the photo may contain severe levels of noise, which corrupt the gradient calculations and in turn induce error in the angle calculation. In addition, the discrete nature of the image domain combined with the small size of commonly accepted gradient detection kernels introduces significant inaccuracies. Sampling from a discrete domain means the data used is limited to a specific set of values and corresponds to a certain resolution. This limits the set of angles that can be perfectly represented by an image and subsequently captured in a kernel. When considering this in addition to the 90-degree separation of the $x$ and $y$ gradients, it is nearly impossible to estimate an edge’s angle with any degree of acceptable error. The wide view filter and line filter aim to address these problems. Because the wide view filter and line filter can consider a significantly larger area in its gradient calculation, it is much less susceptible to
noise and able to represent a finer resolution in this discrete domain (without inter-pixel interpolation). On top of that, the wide view and line filters can calculate gradients in an infinite number of directions depending on their size. This means the filters are capable of producing gradients directly from an edge at any arbitrary angle rather than interpolating them. These properties allow the filters to estimate edge angle within one degree of accuracy.

5.1 USING CUBIC SPLINE TO CALCULATE ANGLE AT A SINGLE PIXEL

The general idea in accomplishing this is to use the line filter or wide view filter to calculate the derivative values of each pixel at various orientation angles. All the derivative values produced by the line filter can then be used as data points along with their respective angle values to form a smooth function describing the relationship between filter orientation and gradient value. These data points are used to fit a cubic periodic spline which establishes the relationship between the derivative values and the orientation angles. Cubic periodic splines are capable of fitting piecewise functions to bridge data points which are smooth about the first and second derivative. This results in a smooth flowing function describing an entire set of data. In this case, it can then be used to calculate where the maximum derivative occurs for a specific pixel. Finding the maximum value in the cubic spline, or the point in which its slope is zero will correspond to a specific angle along the function. This means the filter angle that produces the maximum gradient at a given pixel can be estimated.
5.2 ANGLE DETECTION PROCESS

Figure 5.1 Illustration of a cubic spline for one pixel contributing to the histogram containing all pixel angle values for the image

Figure 5.1 illustrates the first stage of the angle detection procedure.

1. The line filter is first run at a user specified number of angles, defining the angular resolution. The derivative maps produced by the line filter at this angular resolution are fed in as the input of the angle detection script.

2. The angle detection script parses through each pixel of the derivative maps to produce a histogram of all the angles calculated for each pixel in the scene. Each pixel will have multiple derivative values calculated at various orientations by the line filter. These derivative values along with the orientation of the line filter that produced them provide the data point to build the periodic cubic spline at the indexed pixel.

3. The spline provides a set of piecewise function continuous to the second derivative (Figure 5.1a). This can be leveraged to find the angle value, which produced the
maximum derivative value for the pixel. First, the maximum derivative that the line filter was able to produce is found and then indexed on the spline. Because each gradient produced by the filter becomes a midpoint between two piecewise functions on the spline, the two surrounding the maximum derivative value must be analyzed. The piecewise functions that precede this value and come after are are examined for zero slope values along their respective intervals. Zero slope locations correspond to local minimums and maximums on the spline. Only one zero derivative value should exist for the two functions when analyzed over their respective domains.

4. The angle location at which the derivative zero value occurs is defined as the angle of the edge as that is where the maximum derivative occurs. In cases where the indexed pixel lies on an edge, the derivative values will be large and easy to process. However, in cases where the pixel lies in noise or in the background, the derivative values will be much smaller and fluctuate dramatically with the orientation angle. By measuring the range of the derivative values produced on a pixel’s spline and comparing them to the overall range of derivatives found in the image, the indexed pixel can be disregarded if its derivative range is too small.

5. After the angle is calculated for the indexed pixel, the script moves to the next pixel in the image and repeats the process to calculate its angle. All the calculated angles are then stored in a two-dimensional matrix which will be used to create a histogram displaying the frequency of all the angles calculated in the image (Figure 5.1b).
After this histogram is created, the script performs a set of post processing operations to clean the data and eventually cluster all the angles to match the edges in the image. The post processing steps are illustrated in Figure 5.2.

1. The first step in postprocessing is to separate any edges found to be at 180 degrees from the other data in the 170-to-180-degree range. This is because 180 degrees is geometrically equivalent to 0 degrees and any edges at 180 degrees should only be classified as 0 degrees. To separate the data at 180 degrees, the data is analyzed at a small scale of only 10 degrees, which searches for any peak values in the data preceding the 180-degree bin. Any peak values in the data indicate a decrease in the frequency of pixels in the subsequent bin and mark a point where the data can be separated.
Sometimes no peaks can be found, and the data is instead clustered and examined for any clusters falling within a tolerance of $\frac{1}{2}$ the angular resolution to 180 degrees. If either a peak is found or a cluster is calculated within the tolerance to 180 degrees, then the pixels after the peak or the pixels in the cluster are shifted to 0 by subtracting $\pi$ from their angle values.

2. Once the histogram has been shifted, the data should be cleaned up to facilitate use of the clustering method. The mean of the histogram data is first calculated and any bins with occurrences smaller than the mean are disregarded (Figure 5.2a). This will remove the background pixels with random noises in the images as they do not exhibit dominant angle detections (Figure 5.2b).

3. After the noisy data is removed from the histogram, the data can be clustered into groups, which correspond to edges found in the image (Figure 5.2c). Any pixels within a cluster can then be remapped onto the image to visualize their final calculated angles (Figure 5.2d).

5.3 ANGLE DETECTION RESULTS AND ANALYSIS

To determine how accurate angle detection is, the process was first tested on a variety of synthetic images. The synthetic images provide a measurable ground truth which can be used to compare against the outputs of the program. They also allow testing on a variety of cases which can represent challenges that often arise in the real world like closely aligned but still distinct edges and extremely low signal to noise ratios. The worst possible case for angle detection would be a dimly lit scene with many objects in close proximity, however such a situation cannot be used for testing as no measurable ground
truth can be determined after the scene has been captured in a photo. Synthetic line tests allow for replication of these challenging conditions so that the accuracies in such situations can be predetermined, and confidence can be ensured when using the program.

The first image tested with the angle detection script is a multi-line image consisting of variably oriented lines with two closely aligned to each other. The purpose of using this image is to gauge the script’s performance in detecting the angle of two lines with similar angles and being able to separate them. This photo in Figure 5.3 has also been corrupted with a signal to noise ratio of two to further complexify the test.

![Figure 5.3 The synthetic line image](image)

The script performed very well on the synthetic line image and achieved desired results immediately. The results are presented in Table 5.1 and Figure 5.4. Table 5.1 contains the numerical results for each edge in two formats. The cluster center angle represents the average angle of all the pixels included in its group while the peak angle represents the most frequent angle appearing in a group. In most cases, the cluster center is more reliable.

These were obtained running the line filter once every 10 degrees from 0 all the way to 180. The angles for each line fall well within the tolerance range of plus or minus
one degree from the true value aside from line at 174 degrees. Even still, the estimated angle for this line is only slightly out of range at 1.1 degrees. On top of this, the lines at 174 degrees and 180 degrees can be reliably separated. Some pixels are falsely identified in the wrong angle group; however, this is a result of the noise skewing their true value and does not occur frequently enough to cause issue.

Figure 5.4 Multi line photo with close lines angle detection results at a 10-degree angular resolution

Table 5.1 Numerical comparison of calculated angles at 10-degree angular resolution

<table>
<thead>
<tr>
<th>True Angle</th>
<th>Cluster Center Angle</th>
<th>Cluster Peak Angle</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-0.733</td>
<td>-0.527</td>
</tr>
<tr>
<td>23</td>
<td>22.908</td>
<td>22.426</td>
</tr>
</tbody>
</table>
While the previous photo presented a case to test the separation capabilities of the angle detection process, we wanted to ensure the capabilities by using another compact synthetic line photo with an even more difficult case. This photo, shown in Figure 5.5, contains many more lines and decreases the overall separation of every line in the photo.

<table>
<thead>
<tr>
<th>Angle (Degree)</th>
<th>Measured Value</th>
<th>True Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>63.5</td>
<td>63.113</td>
<td>62.120</td>
</tr>
<tr>
<td>90</td>
<td>89.990</td>
<td>90.241</td>
</tr>
<tr>
<td>135</td>
<td>135.002</td>
<td>135.848</td>
</tr>
<tr>
<td>174</td>
<td>172.908</td>
<td>174.855</td>
</tr>
</tbody>
</table>

This serves as a test of overloading the histogram with peaks to examine the performance of the clustering method. The angle detection script again had no problem handling this image and produced accurate results within half a degree of the true angle for each of the lines in the photo. Its results can be seen in Figure 5.6 and Table 5.2. Somewhat surprisingly, in this case, the same two lines at 174 and 180 degrees from the previous test, were more separable. This could be caused by a number of factors but is most likely a result
of the clustering algorithm which is more attuned to create smaller groups of angles for this photo since it contains many lines of varying sizes as compared to the previous photo.

Figure 5.6 Dense line photo with close lines angle detection results at a 10-degree angular resolution

| Table 5.2 Numerical comparison of calculated angles at 10-degree angular resolution |
|-------------------------------|-------------------|-------------------|
| True Angle | Cluster Center Angle | Cluster Peak Angle |
| 0  | -0.435 | -0.573 |
| 23 | 23.063 | 22.598 |
| 45 | 45.159 | 44.118 |
| 63.5 | 63.237 | 62.504 |
| 90 | 89.808 | 90.126 |
| 105 | 105.011 | 104.479 |
Similarly, to the wide view and line filter results analysis, it is desirable to test the angle detection algorithm on a real-world photo to ensure it has the potential to be used in real applications. The next photo tested therefore was a photo of Lake Murray wave field, as shown in Figure 5.7.

![Figure 5.7 Image of Lake Murray Water](image)

This picture was chosen because it provides a few interesting challenges for applying the angle detection script. The angle detection script could be useful in scenarios, where additional environmental data is necessary. In the case of the Lake Murray image, highly accurate angular data can reveal trends and information regarding the wave field direction, possible subsurface obstacles, and direction of the underwater currents. This can prove to be challenging especially for photos like this because waves are a noisy group of objects to track and are constantly changing.

Interestingly, the angle detection script was able to extract features not readily visible to the naked eye. According to the photo, it appears there is one dominant direction
that the wave crests are orientated and traveling at. What is not apparent is that a secondary wave field exists, which the angle detection script can capture. When we zoom in on the photo, each wave field is visible. Figure 5.8 and Table 5.3 show the angle detection script at a 10-degree angular resolution. Because the image is not man made, there is no known ground truth to compare the estimated angles with. With the enlarged view, however, the calculated angles appear very close to what can be visually estimated and measured after capture.

As stated, the angle detection script can reveal a second wave field which cannot be visually extracted without prior indication. Figure 5.9 shows a better visual to observe this. One dominant wave field overpowers the smaller secondary in the large picture, while in the zoomed in version there are two dominant directions of the waves. Over time in the frame sequence, these two directions can be confirmed as the waves travel orthogonally to their dominant direction.
Figure 5.8 Lake Murray photo with close lines angle detection results at a 10-degree angular resolution

Figure 5.9 Zoomed in view of Lake Murray Wave Fields

<table>
<thead>
<tr>
<th>Cluster Center Angle</th>
<th>Cluster Peak Angle</th>
</tr>
</thead>
<tbody>
<tr>
<td>78.931</td>
<td>79.985</td>
</tr>
<tr>
<td>90.484</td>
<td>89.347</td>
</tr>
</tbody>
</table>

Table 5.3 Numerical comparison of calculated angles at 10-degree angular resolution

The last photo tested by the angle detection script is a lower SNR version of the second multi-line photo tested, used to gauge the capabilities of the angle detection script when presented with a wholly noisy photo.
Without changing the configurations of the line filter of angle detection script, the results came out quite inaccurate. This is somewhat expected as all previous photos had been somewhat standard scenarios, and analyses from the filters themselves have shown that increased filter sizes are necessary to suppress more noise. Because of this, the line filter size had to be increased to a length of 29 pixels and a neighborhood size of 150 pixels per wide view filter. The results using this configuration can be seen in Figure 5.11 and Table 5.4. An increased filter length both reduced the number of false detections among the noise and improved the accuracy of the detected angles. The final angle calculations all came within the desired one degree of tolerance, and many of them were significantly closer. Even though the filter had to be increased in size, the calculations were done with a similar computational efficiency as compared to the original size filter. This means the highly accurate angle calculations came with minimal additional costs.
Figure 5.11 Multi line photo with close lines and low SNR angle detection results at a 10-degree angular resolution (Np = 150 & Filter Size = 29px)
Table 5.4 Numerical comparison of calculated angles at 10-degree angular resolution with a 29-pixel filter length and Np = 150

<table>
<thead>
<tr>
<th>True Angle</th>
<th>Cluster Center Angle</th>
<th>Cluster Peak Angle</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.075</td>
<td>-0.057</td>
</tr>
<tr>
<td>23</td>
<td>22.303</td>
<td>22.632</td>
</tr>
<tr>
<td>63.5</td>
<td>63.139</td>
<td>61.564</td>
</tr>
<tr>
<td>90</td>
<td>90.003</td>
<td>89.742</td>
</tr>
<tr>
<td>135</td>
<td>134.443</td>
<td>133.912</td>
</tr>
<tr>
<td>174</td>
<td>174.074</td>
<td>174.815</td>
</tr>
</tbody>
</table>
CHAPTER 6

CONCLUSION

The wide view filter demonstrates great promise for edge detection in these case studies, and there are several important observations worth pointing out. The use of the wide view enables an effective means of suppressing noise in low SNR images; second, by examining the tangential and the normal direction of the filter (viz., the local axes orientation), necessary directional information for object edges can be extracted, making edge detection direction-selective; and last, using a wider view, i.e., more surrounding pixels enhances noise suppression and increases the contrast ratio between the signals and the noises and connectivity between scattered edges, while making the edge thicker.

The line filter builds upon the promise that the wide view filter displayed. The results presented show that the line filter exhibits strong edge detection when it aligns with the edges, and therefore, it can be used to accurately determine the direction of the lines with proper postprocessing. Additionally, it shows a large improvement of connecting broken edges over the wide view filter. When the line filter is placed at the correct orientation for an edge it should be able to increase the amount of information that may have otherwise been lost due to noise or obstructions. In addition, the SNR analysis performed shows that the line filter is proficient in handling large noise levels in a photo. Notably, the line filter can handle noise levels previously impossible to work with when using traditional gradient operators. The architecture of the line filter allows it to be adapted to many scenarios it may encounter and should be able to handle real-world
levels of noise when using the right configurations. Developments with the line filter can be made, including an angle detection algorithm to work in conjunction with the line filter, which will reveal the orientation of an edge with a high degree of accuracy.

The first stage of angle detection proves to be flexible, adaptable, and accurate. When combined with a sufficient line filter configuration, angles can be determined to within one degree of their true value every time even with very low quality of the image. This accuracy is unachievable without the use of the line filter and associated angle detection methods even with pristine images.

It should be noted that the current implementation of angle detection can likely be improved from a computational aspect. The script is run for every pixel in an image despite its edge status. This results in a lot of wasted computation for pixels which do not require angular determination. An easy and efficient fix for this would be to perform edge determination before and gather the indices of the edges. This set of indices would serve as the domain for angle detection to take place.
REFERENCES


Alpert, Sharon, & Galun, Meirav, & Nadler, Boaz, & Basri, Ronen (2010). Detecting Faint Curved Edges in Noisy Images. *European Conference on Computer Vision*, 750-763, [https://doi.org/10.1007/978-3-642-15561-1_54](https://doi.org/10.1007/978-3-642-15561-1_54)


Donoser, Michael, & Schmalsteig, Dieter (2014). Discrete-Continuous Gradient Orientation Estimation for Faster Image Segmentation. *IEEE Conference on Computer Vision and Pattern Recognition*, 3158-3165, [https://doi.org/10.1109/CVPR.2014.404](https://doi.org/10.1109/CVPR.2014.404)


