Fall 12-2019

A Longitudinal Study of Mammograms Utilizing the Automated Wavelet Transform Modulus Maxima Method

Brian C. Toner
University of Maine, brian.toner@maine.edu

A LONGITUDINAL STUDY OF MAMMOGRAMS UTILIZING THE AUTOMATED WAVELET TRANSFORM MODULUS MAXIMA METHOD

By

Brian Christopher Toner

B.A. Mathematics University of Maine, 2013

M.A. Mathematics University of Maine, 2015

A DISSERTATION

Submitted in Partial Fulfillment of the

Requirements for the Degree of

Doctor of Philosophy

(in Computer Science)

The Graduate School

The University of Maine

December 2019

A LONGITUDINAL STUDY OF MAMMOGRAMS UTILIZING THE AUTOMATED WAVELET TRANSFORM MODULUS MAXIMA METHOD

By Brian Christopher Toner

Dissertation Co-Advisors: James Fastook, Andre Khalil

An Abstract of the Dissertation Presented
in Partial Fulfillment of the Requirements for the
Degree of Doctor of Philosophy
(in Computer Science)
December 2019

breast cancer, image analysis, WTMM, WTMMM, computer science

Breast cancer is a disease which predominatly affects women. About 1 in 8 women are diagnosed with breast cancer during their lifetime. Early detection is key to increasing the survival rate of breast cancer patients since the longer the tumor goes undetected, the more deadly it can become. The modern approach for diagnosing breast cancer relies on a combination of self-breast exams and mammography to detect the formation of tumors. However, this approach only accounts for tumors which are either detectable by touch or are large enough to be observed during a screening mammogram. For some individuals, by the time a tumor is detected, it has already progressed to a deadly stage.

Unlike previous research, this paper focuses on the predetection of tumorous tissue. This novel approach sets out to examine changes in the breast microenvironment instead of locating and identifying tumors. The purpose of this paper is to explore whether it is possible to discover changes in the breast tissue microenvironment which later develop into breast cancer.

We hypothesized that changes in the breast tissue would be detected by analyzing mammograms from the years prior to the discovery of tumorous tissue by a radiologist. We analyzed a set of time-series digital mammograms corresponding to 26 longitudinal cancer cases, obtained through a collaboration with Eastern Maine Medical Center (EMMC) in Bangor, Maine. We automated the Wavelet Transform Modulus Maxima (WTMM) method, a mathematical formalism that we used to perform a multifractal analysis. In particular, this automated WTMM (AWTMM) was used to calculate the Hurst exponent, a metric that is correlated with breast tissue density. The AWTMM allowed us to see with greater detail the changes in mammogram tissue, specifically concerning breast density. The results suggest that signs of malignancy can be observed as early as two years before standard radiological procedures. In this research, we identify a set of variables that show significance when classifying precancerous tissue.

PREFACE

This work represents the culmination of years of research. While there are many pages of material, there are many more which never made it to this final draft. The author's ambition was to make this paper accessible to a broad audience. However, to gain more in-depth insight into the paper, it is recommended that one also read the companion papers to this research: "Comparative multifractal analysis of dynamic infrared thermograms and X-ray mammograms enlightens changes in the environment of malignant tumors," "Mammographic evidence of microenvironment changes in tumorous breasts," and "Computational growth model of breast microcalcification clusters in simulated mammographic environments."

This paper serves as an overview of the current state of mammogram research in the CompuMAINE laboratory. The results laid out in this paper are a testament to the hard work and effort of many folks, and a sign of the many technical obstacles overcome. The author hope is that these results will be used to further our understanding of the breast microenvironment and help to advance our knowledge of breast cancer.

DEDICATION

$\sim$ To Amber

ACKNOWLEDGEMENTS

I cannot express enough gratitude to all of those people in my life who helped me get to this point. It has been a long journey. This venture and the many faces I have met along the way will stay with me.

My thanks go out firstly to Amber Hathaway. We have had our ups and downs, but she has always been at my side, ready to aid me when the time arises. This paper would have you believe that I am fluent in English, yet it would not be, but for her helpful phrasings. She has helped to shape this document into something readable and understandable.

I thank my advisor Andre Khalil, who has been patient, kind, and understanding during this ordeal. I cannot count the number of times that he has been my advocate; indeed, I am sure it is on the order of $|\mathbb{R}|$. I would not be where I am without the many opportunities and support he has provided me with over the years.

I would also like to thank my committee, Dr. Phillip Dickens, Dr. James Fastook, Dr. Amy Harrow, and Dr. Roy Turner. Without their guidance and support, this paper would not have been possible. Their willingness to help and support me throughout the years has been invaluable.

I would like to thank the Computer Science department for the graduate assistanceships they provided over the years, as well as the Department of Mathematics and Statistics for their support. Finally, I would like to thank Dr. Kody Varahramyan for supplying the research assistanceship I had over the last year.

I thank my family and friends for their support. I hope to make up the missed time once I have finished this program. In particular, I want to thank my mother and grandmother, for the many Saturday mornings, lunches, and conversations we shared over the years. I want to thank my Aunt Mary and Uncle Jim, whose frequent visits were delightful. I will not forget their kindness and generosity. I also thank my Uncle Tom and Aunt Terri, whose kindness and warmth was always appreciated. I thank my father for helping to foster my curiosity for computers. I thank my siblings, who all have never been more than a phone call away. Finally, I thank Mark Royer and Zac Hutchinson, who both have shared in the many trials we have faced through the years. In particular, I am grateful for the many lab meetings we have shared.

Thank you to the efforts of the members of the CompuMAINE laboratory. I am especially indebted to Basel White, who helped to label much of the data used in this analysis. His efforts help me greatly here.

I also want to thank the many SCIS and Mathematics staff and faculty at The University of Maine. They have provided me with an abundance of resources that have aided me through the years. In particular, I thank Karen Kidder, Wendy Robertson, Cathy Brown, and Tamara Hembree for their willingness to help. Further, I thank Dr. Harlan Onsrud, who was always willing to listen and made time to find out how our semesters were progressing. Additionally, I want to thank Dr. Bob Franzosa, who was always there to lend an ear and support me throughout the years, and who always gave me enough rope to perform the proper reidemeister move.

Finally, I want to acknowledge and thank Dr. Alain Arneodo who succumbed to brain cancer earlier this year. Without his contributions to this field, none of this research would be possible.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

AWTMM	Automated WTMM
CPU	Central Processing Unit
DCIS	Ductal Carcinoma In Situ
GPU	Graphical Processing Unit
H	Hurst Exponent
HA	Human Agent
HER2	Human Epidermal growth factor Receptor 2
HR	Hormone Receptor
IDC	Invasive Ductal Carcinoma
OpenCL	Open Computing Language
OpenMPI	Open Message Passing Interface
WTMM	Wavelet-Transform Modulus Maxima
WTMMM	Wavelet Transform Modulus Maxima Maxima

CHAPTER 1
INTRODUCTION

1.1 Introduction

Around 1 in 8 women will be diagnosed with breast cancer in her lifetime [1]. In 2017, there were an estimated 252,710 new cases of invasive breast cancer reported in American women and 2,470 cases diagnosed in American men [1].[1] Breast cancer is primarily a disease that affects older women, with only 3% of cases occurring in women under the age of 40 [2]. Many factors affect the severity and rate of occurrence of breast cancer. One such factor is breast density.[2] Breast density is determined by the ratio of glandular and connective breast tissue to total breast tissue [2]. Women with 26%-50% breast density have a 1.6 times greater and women with higher than 50% breast density have a 2.3 times greater chance of developing breast cancer than women with 11%-25% breast density [3]. Further, breast cancer is more difficult to detect in mammograms with greater ratios of dense to fatty breast tissue [4].

Mammographic breast cancer screenings can detect signs of breast cancer up to three years before a lump can be felt during a self-breast exam [2]. Breast mammography acts as a sentinel, alerting doctors to the growth of suspicious lesions. Early detection has been credited with the declining mortality rate of breast cancer in western countries [2]. However, there is evidence to suggest that screening may not help mortality rates for advanced stages of breast cancer. This failure to detect some breast cancer until it has reached an advanced stage may be because mammographic screenings are designed to look for tumors in the breast. However, some forms of breast cancer metastasize before they can grow to a size detectable in a mammogram [5].

[1] The incidents of breast cancer in transgender and gender nonconforming individuals are unknown.

[2] While young women have denser breast tissue than older women, breast density is not necessarily a risk factor in women under 40.

The objective of this research was to prove that the automated 2D wavelet-transform modulus maxima (AWTMM) method not only can be used to detect fluctuations in tissue microenvironment for longitudinal data, but also to demonstrate that these fluctuations are useful for detecting the signs of malignant tissue before an official radiological diagnosis. In this research, we present evidence that suggests that there are detectable changes in mammograms. We identified a set of variables that show significance in classifying precancerous tissue through comparing aggregated time groups of longitudinal mammograms. A summary of these results can be found in Chapter 6.

1.2 Background

In this study, we analyzed mammograms from patients that fall into two categories of cancerous breast tissue, the first being ductal carcinoma in situ (DCIS). DCIS is a condition in which epithelial cells mutate and begin to form abnormal cell clusters. Generally, DCIS is benign, with only 17% of cases developing into invasive cancer [2]. In many cases, DCIS grows slowly enough that it will have no impact on the patient's health [2]. However, between 20%-53% of cases of cancer are misclassified as DCIS [6–10]. The second group we examined was a general cancer category labeled as invasive ductal carcinoma (IDC). Invasive breast cancers are the most common type of breast cancer, making up 80% of cancer cases [11–13]. There are up to 21 different types of breast cancer, which generally fall into four different molecular subgroups,[3] each with unique properties, including detection and treatment [2]. Since we did not have access to the specific type of cancer, we grouped these invasive cancers into a single category. Ideally, given enough data, the mammograms should be filtered into more granular categories. However, the most common type of breast cancer in non-Hispanic white women is HR+/HER2-, accounting for around 82% of breast cancer cases in this population [2]. Since these data

[3]Three of the four categories rely on classifying the hormone receptor (HR) and human epidermal growth factor receptor 2 (HER2), with the four groups being HR+/HER2-, HR+/HER2+, HR-/HER2+, and Triple negative.

were acquired from a Maine based hospital and given that 94.64% [14] of the population of Maine falls into this demographic, most of the cases will likely be HR+/HER2-.

The major disadvantage of mammography is that it relies on tumor detection. Up to this point, little research has been conducted on detecting changes in breast tissue that lead to breast cancer. Most computer-aided diagnostics (CAD), like radiologists, focus on tumor detection. However, not all breast cancers can be detected this way [15]. Further, previous studies have shown that radiologists can overlook lesions that are obscured by dense breast tissue [15]. Many times, these cancers are able to be visually detected, in previous mammograms, after the eventual diagnosis [15]. Much in the way that mammograms have revolutionized early detection of tumors, this research hopes to go further and utilize these mammograms to find visually undetectable changes in mammograms that can be attributed to the development of breast cancer.

We also demonstrated that changes in the microenvironment of breast tissue were not only detectable but also useful in distinguishing between breasts containing malignant tissue and those without malignant tissue [16, 17]. Using the 2D wavelet-transform modulus maxima (WTMM) method, we were able to determine the relationship between breast tissue type and the Hurst exponent (H) [16], which is a metric used to quantify the global roughness of the images density fluctuations. Fluctuations in mammographic breast tissue fall into three categories: monofractal anti-correlated ($H < 0.45$) for fatty tissue, monofractal long-range correlated ($H > 0.55$) for dense tissue, and uncorrelated ($0.45 \leq H \leq 0.55$) for disrupted tissue [16]. A rough outline of these three categories was originally demonstrated in [18]. From this work, these regions were formally established and named in [16], with the disrupted tissue being coined in order to characterize tissue which exhibited high entropy (e.g., $0.45 \leq H \leq 0.55$) [16]. We discovered that disrupted tissue regions were found more frequently in tumorous than in healthy breast tissue. More specifically, tissue regions with $0.45 \leq H \leq 0.55$, as well as left versus right breast asymmetries, were found preferably in tumorous breasts when compared to normal breasts

($p < 0.0006$), as quantified using with the combined metric (Equation 12 in Marin et. al.) [16]. The leading hypothesis is that the changing microenvironment of the breast tissue causes the loss of tissue homeostasis, promoting the spread of disrupted-tissue regions [19–22].

Concurrently, while working on our research in Marin et. al., [16] we were working on simulating the growth of microcalcifications. In Plourde et. al. [23] we hypothesized that microcalcifications would be more likely to grow in breast tissue with a high level of entropy. In this experiment we simulated breast tissue with fractional Brownian noise. We discovered that the H value of the simulated breast tissue impacted the the growth rate of the simulated microcalcifications [23]. This further reinforced our hypothesis that the tissue microenvironment may play an important role in the development of breast cancer.

These insights suggest that our research is nearing a point where we may soon be able to predict the formation of malignant tissue accurately. More specifically, we hypothesized that the AWTMM would be able to be used to detect such changes in a time series analysis of mammograms. In this paper, we extend the scope of previous research by analyzing the time series data.

1.3 History

The 2D wavelet-transform modulus maxima (2D WTMM) method is a signal processing technique which is used to compute the multifractal properties of an image. Up until 2015, parameter settings and multiple power-law fittings for the 2D WTMM were selected manually. The experts, or Human Agents (HA), performing this task would visually inspect these sets of graphs, manually selecting a range in these graphs. Selecting this range was tedious, time consuming, and required a lot of training. A well trained HA could fit these curves at a rate of around one mammogram subregion per minute. Since each mammogram was gridded off into 256x256 pixel subregions, it could take several hours for a single HA to finish off one mammogram. For example, a single 3000x3900 pixel mammogram would be divided into around 11 columns and 15 rows. If 60% of the subregions contained breast tissue, then this mammogram would have around 100 subregions to be analyzed, which equates to around 100 minutes spent analyzing the mammogram. Of course, if an image was larger, or contained more breast tissue, it would take even longer for the HA to classify the subregions. This does not take into account the time spent running the 2D WTMM method, which, if not precomputed, the HA would have to wait for the 2D WTMM method to generate the curves before they could classify the subregion. For a single subregion, the 2D WTMM method could take up to half a minute from start to finish. To continue the example, this means that the HA could wait up to fifty minutes for the WTMM to be computed for each of the 100 regions.

There were a couple of issues that arose from processing the images this way. The first was that it was time consuming, both to train HA and to classify the subregions. Further, the fitting procedure was subjective, meaning that two expert HAs may not always agree on the best fit. In an internal audit, we found the average agreement between two expert HAs to be around 85%. This meant that while there is a high level of agreement between HAs, the classification of the regions would not be entirely consistent. For this reason, HAs had to regularly meet to discuss their classification of regions and to appraise one another's assessments to maintain a high level of consistency.

For these reasons, we set out to find a way to automate this procedure. Early builds of the automated WTMM (AWTMM) were able to classify the regions with an average agreement of around 85% with the expert HAs, meaning they had a high level of agreement with an expert HA. Further, the AWTMM was able to classify the output of the WTMM of a single region in less than a second. This meant that all the subregions of a mammogram could be classified in minutes instead of hours. The same 100 region mammogram discussed previously could now be analyzed in approximately 16 minutes. The AWTMM also freed up time for HAs, who were often senior scientists with other responsibilities.

From this point, we changed our focus from the traditional griding method outlined above to a more complex system (see Chapter 5). We hypothesized that additional information on how the tissue changed across the mammogram would provide us with new insights into the development of breast cancer. Now instead of hundreds of subregions, a single mammogram could be divided into thousands of subregions that needed to be classified. The number of subregions approximately increased our resolution sixty-four fold from the original griding protocol. That is, $n2^2 2^2 2^2 = n2^6$, where n is the number of subregions to classify. From our example above, if a mammogram had 100 subregions to classify originally, under the new schema there would now be 6400 regions to classify.

Early implementations of this system were able to classify a single mammogram with the new finer griding system in around 17 hours. Most of that 17 hours was dedicated to the time needed to run the WTMM on each of the 6400 regions. For a HA, this would have taken around 4.4 days, excluding the WTMM time. The major disadvantage of this new system was that while it was faster and produced much larger quantities of more refined data, analyzing data in bulk now required a huge computational investment. Though not ideal, one of the ways to combat this was to run multiple sessions of the WTMM/AWTMM in tandem across multiple machines. While this was still a step up from running a single instance across a single machine, this arrangement added additional complexity to the pipeline. It also meant that multiple machines needed to be monitored, and that the load across machines was not consolidated, adding complication to our method. Eventually, some of these tasks were handled by utilizing bash shell scripts, but the system was still inefficient.

To remedy these problems, we decided that a ground up rework of the AWTMM and mammogram processing pipeline was necessary. First, the AWTMM algorithm was streamlined and converted to OpenCL [24] code. This meant that we could now take advantage of the GPUs to compute the AWTMM. Next, the software was altered to utilize OpenMPI [25]. Since each subregion is calculated independently of the rest of the image, this meant that we could implement an embarrassingly parallelizable schema. In other words, we could have one supervisor thread manage a collection of worker nodes. These worker nodes would request a region from the supervisor. Once they completed the task of computing the WTMM/AWTMM for their designated region, they would report back the results and request the next region. This continued until no regions were left in the queue, which resulted in a directly linear increase in speed based on the total number of worker nodes. Thus, if we split our 6400 regions over 80 CPUs, then the mammogram would now only take around 13 minutes to compute. Further, the scalability of this solution allowed us to add and remove nodes as needed, depending on workload and urgency.

As of the writing of this paper, this is the current structure and function of the mammogram analysis pipeline. In addition to the stated hypothesis, this paper will also serve to outline a formal mathematical definition of the AWTMM, assess its capabilities, and discuss future improvements in the mammogram analysis pipeline. Additionally, we will outline some of the other mathematical and algorithmic tools utilized in our image analysis pipeline.

CHAPTER 2
THE WTMM METHOD

The wavelet transform (WT), wavelet transform modulus maxima (WTMM), and the wavelet transform modulus maxima maxima (WTMMM) are the underlying mathematical machinery used in this paper. The WTMM utilizes a special class of functions, commonly called wavelets, which have been compared to a 'mathematical microscope' because of their ability to resolve image features [26–28]. The WTMM method can be used as a way to characterize the fluctuations in density across an image [16]. While the WTMM method has its roots in thermodynamics [26, 29, 30], it has been utilized in a wide range of fields due to its ability remove noise and to identify features within images [16–18, 26, 27, 29–53]. With respect to mammography, we are particularly interested in the ability of the WTMM method to classify breast tissue density [16, 17, 39, 45, 46]. For an in depth explanation of the deeper workings of this tool, please refer to one of the many references included here [16, 17, 26–30, 39, 40, 45–47, 50–52, 54–56].

2.1 Wavelet Transform

As mentioned above, the end goal of utilizing the WTMM method in this research is to compute a number which can be correlated to density fluctuations in an image, the H-value. This first step of the WTMM method is to identify where those fluctuations exist in the image. For that reason, the wavelet transform is utilized to help compute the direction and magnitude of the shifts in density at different size scales. This section will provide an overview of the calculations needed to perform the wavelet transform. We define the vectorized form of the wavelet transform as

$$T_{\psi}[f](\mathbf{b},a) = \begin{pmatrix} T_{\psi_1}[f](\mathbf{b},a) = \frac{1}{a^2}\int\int f(\mathbf{x})\psi_1\left(\frac{\mathbf{x}-\mathbf{b}}{a}\right)d^2\mathbf{x} \\ T_{\psi_2}[f](\mathbf{b},a) = \frac{1}{a^2}\int\int f(\mathbf{x})\psi_2\left(\frac{\mathbf{x}-\mathbf{b}}{a}\right)d^2\mathbf{x} \end{pmatrix}, \tag{2.1}$$

where f is a single-valued, self-affine function, that is, the function must have some self similarity.[1] Further, we will let ψ_1 and ψ_2 be equations A.3 and A.4, respectively. Traditionally in the WTMM method, ψ is the first or third order partial derivative of the 2D Gaussian function, $\phi(x, y)$, taken in both the x (Figure A.1a) and y (Figure A.1b) directions at multiple size scales, a. Further, a is the width and height of the Gaussian kernel. In the case of the wavelet transform, the Gaussian smooths the image, while the derivative gives the gradient. The size scales smooth the image at increasing values of a, rendering more prominent features visible while disregarding more localized features.

Notice that the equations in 2.1 and A.1 share a lot of similar features. Observe that if we solve the integral in 2.1 through integration by parts, we get

$$T_\psi[f](\mathbf{b}, a) = \nabla \{T_\phi[f](\mathbf{b}, a)\} = \nabla \{\phi_{\mathbf{b},a} * f\}. \tag{2.2}$$

From here, we can see how the wavelet transform ties in to the idea of the convolution. However, the major difference when computing the wavelet transform is that our kernel must sum to 0. This is a foundational assumption made when working with wavelets, known as the admissibility condition [57]. Two such equations that meet this requirement are Equations A.3 and A.4.

Next, we can express the wavelet transform in terms of the modulus

$$\mathcal{M}_\psi[f](\mathbf{b}, a) = |\mathcal{T}_\psi[f](\mathbf{b}, a)| = \sqrt{T_{\psi_1}[f](\mathbf{b}, a)^2 + T_{\psi_2}[f](\mathbf{b}, a)^2} \tag{2.3}$$

and the argument

$$\mathcal{A}_\psi[f](\mathbf{b}, a) = \operatorname{Arg}(T_{\psi_1}[f](\mathbf{b}, a) + (i)(T_{\psi_2}[f](\mathbf{b}, a))) \tag{2.4}$$

For the WTMM method, the modulus is the magnitude and the argument is the gradient direction of the wavelet transform. As the name suggests, the WTMM (of the wavelet transform modulus maxima method) are defined as the (x, y) coordinates for which the

[1]Equation 2.1 is implemented with a continuous convolution. For a basic overview of a mathematical convolution and how it applies to the WTMM method, see Appendix A.

modulus is maximal in the direction of the argument. More specifically, our goal is to locate the maxima lines in $\mathcal{M}$ and then to locate the maxima lines with the locally maximal modulus. This will allow us to construct something analogous to a topological map of the different image features. This process will be discussed in Section 2.2.

2.2 WTMMM

As mentioned in Section 2.1, the goal of the WTMM method is to locate all the WTMM at a given scale. Once these WTMM have been located over the set of all scales, we can form connected chains which are called maxima chains. The WTMMM are defined as the coordinates along the maxima chains which are locally maximum. These linked chains across all scales ($a > 0$) are known as maxima lines. Let $\mathcal{L}(a)$ be the set of all maxima chains that exist at any scale a. Further, let the partition functions be defined as

$$\mathcal{Z}(q,a) = \sum_{\mathcal{L}\in\mathcal{L}(a)} \left(\sup_{(a')\in\mathcal{L},a'\leq a} \mathcal{M}_{\psi}[f](\mathbf{b},a') \right)^{q}, \tag{2.5}$$

where $q \in \mathbb{R}$ correspond to statistical order moments. These statistical order moments provide insight into the fractal geometry of the image. Utilizing the power-law, we can see how the partition function changes proportionally with respect to a and $\tau(q)$ [26, 29, 30, 55, 56]. We can define the function scaling exponents $\tau(q)$ as

$$\mathcal{Z}(q,a) \sim a^{\tau(q)}, \tag{2.6}$$

where $a \to 0^{+}$. Here q and $\tau(q)$ relate to the fractal properties of the signal, with a nonlinear $\tau(q)$ being related to a multifractal signal. Next let us consider the corresponding singularity spectrum $D(h)$. The singularity spectrum of f can be determined from the Legendre transform of $\tau(q)$ with

$$D(h) = \min_{q}(qh - \tau(q)). \tag{2.7}$$

Because utilizing the Legendre transform can lead to computational instability, we can alternatively use h and $D(h)$ with respect to their Boltzmann weights,

$$\mathcal{W}_\psi[f](q,\mathcal{L},a) = \frac{1}{\mathcal{Z}(q,a)} \left| \sup_{(a')\in\mathcal{L},a'\leq a} \mathcal{M}_\psi[f](\mathbf{x},a') \right|^q, \tag{2.8}$$

utilizing the WTMMM chaining data, to help alleviate these computational problems.

These weights allow us to compute the expectation values

$$h(q,a) = \sum_{\mathcal{L}\in\mathcal{L}(a)} \ln \left| \sup_{(a')\in\mathcal{L},a'\leq a} \mathcal{M}_\psi[f](\mathbf{b},a) \right| \mathcal{W}_\psi[f](q,\mathcal{L},a) \tag{2.9}$$

and

$$D(q,a) = \sum_{\mathcal{L}\in\mathcal{L}(a)} \mathcal{W}_\psi[f](q,\mathcal{L},a)\ln(\mathcal{W}_\psi[f](q,\mathcal{L},a)). \tag{2.10}$$

If we take the limit of $h(q,a)$, then we come up with the equation

$$h(q) = \lim_{a\to 0^+} \frac{h(q,a)}{\ln(a)}. \tag{2.11}$$

Likewise, taking the limit of $D(q,a)$ yields

$$D(q) = \lim_{a\to 0^+} \frac{D(q,a)}{\ln(a)}, \tag{2.12}$$

which allows us to calculate $D(h(q))$. We can now calculate the fractal dimension, $D(h)$, of all points in the image with the Holder exponent h, where the Holder exponent represents the strength of the singularities of the image corresponding to $D(q=0)$ [27,45,55,56]. As long as the $D(h)$ singularity spectrum is monofractal, $D(h)$ can be used to compute a global roughness, which is quantified by the Hurst exponent H, with the corresponding $\tau(q)$ spectrum through the relationship

$$\tau(q) = qH - 2. \tag{2.13}$$

In order to discriminate between a monofractal and multifractal signal, we need to consider a range of q values which is as large as possible. However, image size limits the number of q-values one can utilize, with smaller images being able to utilize fewer q-values.

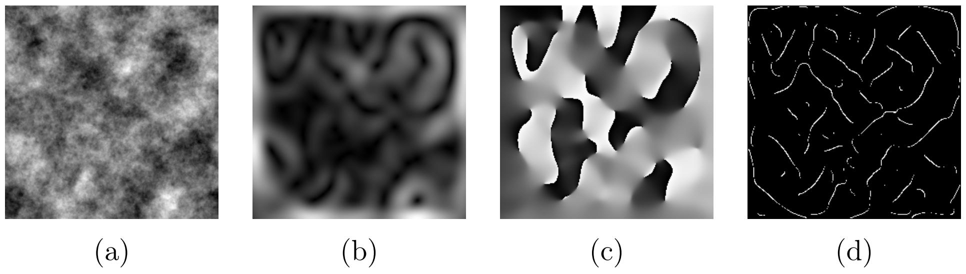


Figure 2.1: An example of the modulus, argument, and maxima chains from a single size scale ($a = 73$).
Figure 2.1b depicts the modulus of 2.1a taken at size scale $a = 73$. Figure 2.1c depicts the argument of 2.1a taken at size scale $a = 73$. Figure 2.1d depicts the maxima chains of 2.1a taken at size scale $a = 73$.

Since the images we utilized were 360×360 pixels, with only the inner 256 pixel region being kept for the WTMM analysis, we only utilize q-values from the range $q \sim -2$ to $q \sim 3$. Further, in this range of q-values, the q-values closer to 0 were given higher weights when performing the AWTMM, as discussed in Chapter 3.

2.3 Visualizing the WTMM

Sections 2.1 and 2.2 provided the mathematical framework for utilizing the WTMM. To help ground those equations, we will now turn to some graphical examples to help illustrate some of their meaning. If one recalls, we described the modulus and argument, equations 2.3 and 2.4, respectively, as the magnitude and direction of the wavelet transform. An example of the modulus, argument, and maxima chains from a single size scale ($a = 73$) can be seen in Figures 2.1b, 2.1c, and 2.1d respectively. Figure 2.2 illustrates the filters, modulus, arguments, and maxima chains at multiple scales.

2.4 Visualizing Space-scale Skeletons

Section 2.3 provides illustrations for the argument, modulus, and maxima lines produced from the WTMM method. In this section we illustrate the space-scale skeletons discussed in Section 2.2. Figure 2.3, taken from Marin et. al., serves as a graphical representation of how the maxima lines can be linked together to form the space-scale

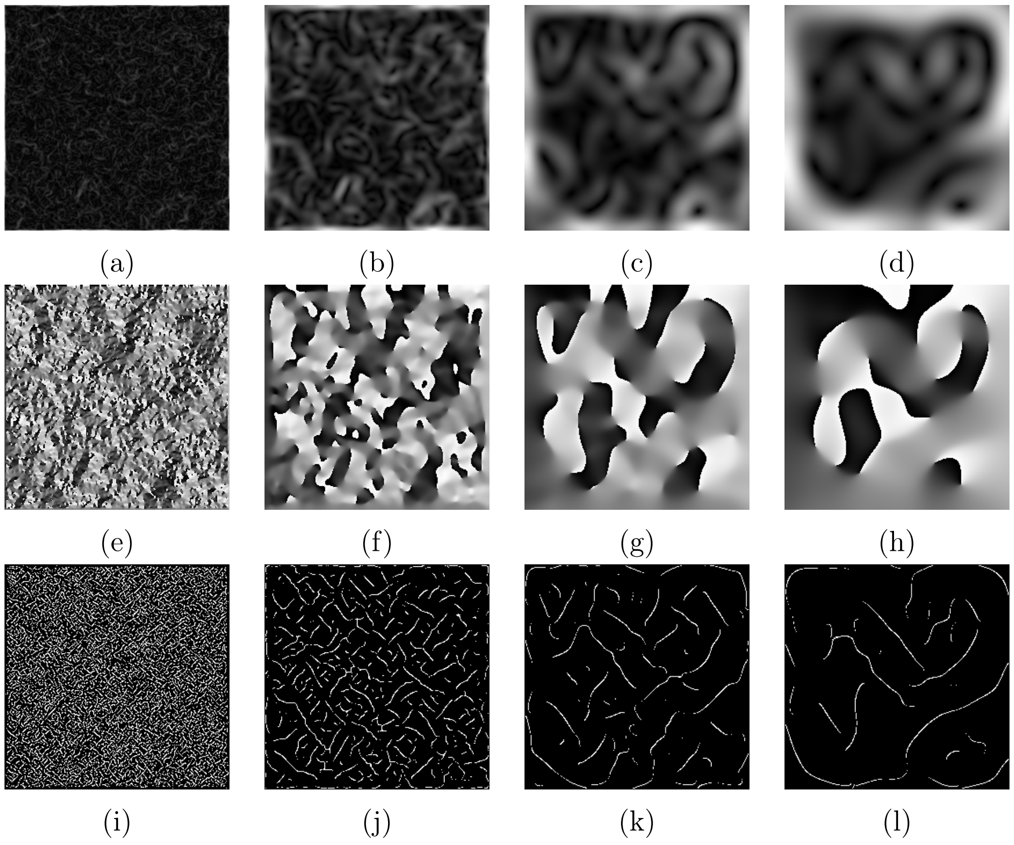


Figure 2.2: The wavelet transform applied to 2.1a at the scales (column-wise) $a = \{7, 29, 73, 127\}$.
Figures 2.2a–2.2d are the modulus at multiple scales. Figures 2.2e–2.2h are the arguments at multiple scales. Figures 2.2i–2.2l are the maxima chains at multiple scales.

skeletons. The first row (Figures 2.3a through 2.3c) depicts three unique regions taken from a mammogram, representing the three types of breast tissue, with the three columns representing fatty, disrupted, and dense breast tissue, respectively. The middle three rows (Figures 2.3d through 2.3l) represent the maxima chains at three different scales. The last row (Figures 2.3m through 2.3o) represents the space-scale skeletons that are constructed from all of the maxima chains.

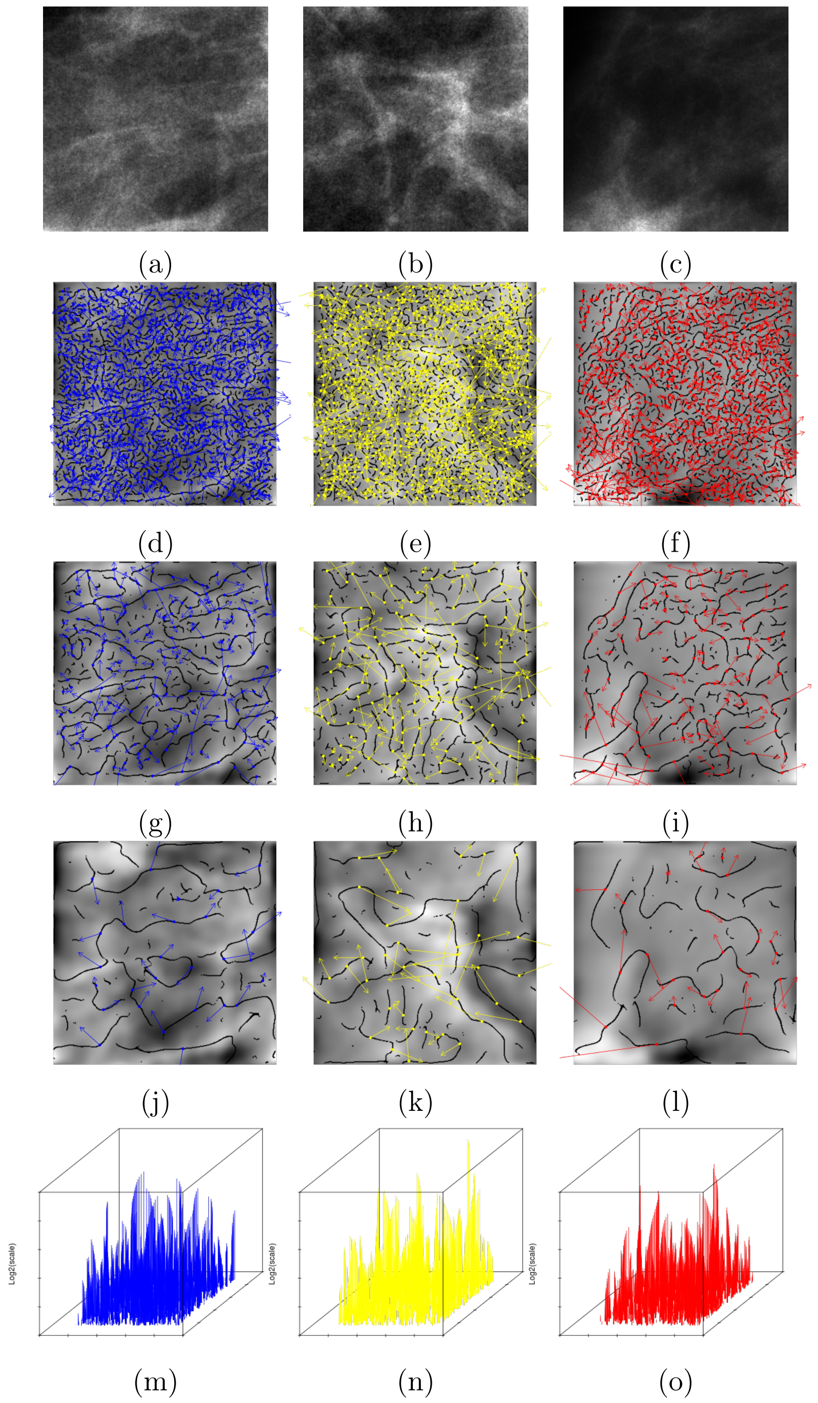


Figure 2.3: Illustration of space-scale skeletons.
This illustration, taken from Marin et. al. [16], depicts the construction of a 'space-scale skeleton' at different scales. The first row, 2.3a through 2.3c, are subsections of mammograms with different H values. The columns represent tissue with $H \leq 0.45$, $0.45 < H < 0.55$, and $H \geq .55$, respectively.

2.5 Visualizing $h(q, a)$ and $D(q, a)$

These $h(q, a)$ and $D(q, a)$ are similar to curves which we will utilize during our calculations of the Automated WTMM, discussed in Chapter 3. An example of the $h(q, a)$ and $D(q, a)$ curves can be seen in Figures 2.4 and 2.5, respectively. Note that in Chapter 3.2, we will look at these curves as sets of sets, $\mathcal{H}$ and $\mathcal{D}$. The notation in this chapter was utilized to reflect the canonical notation of the WTMM method.

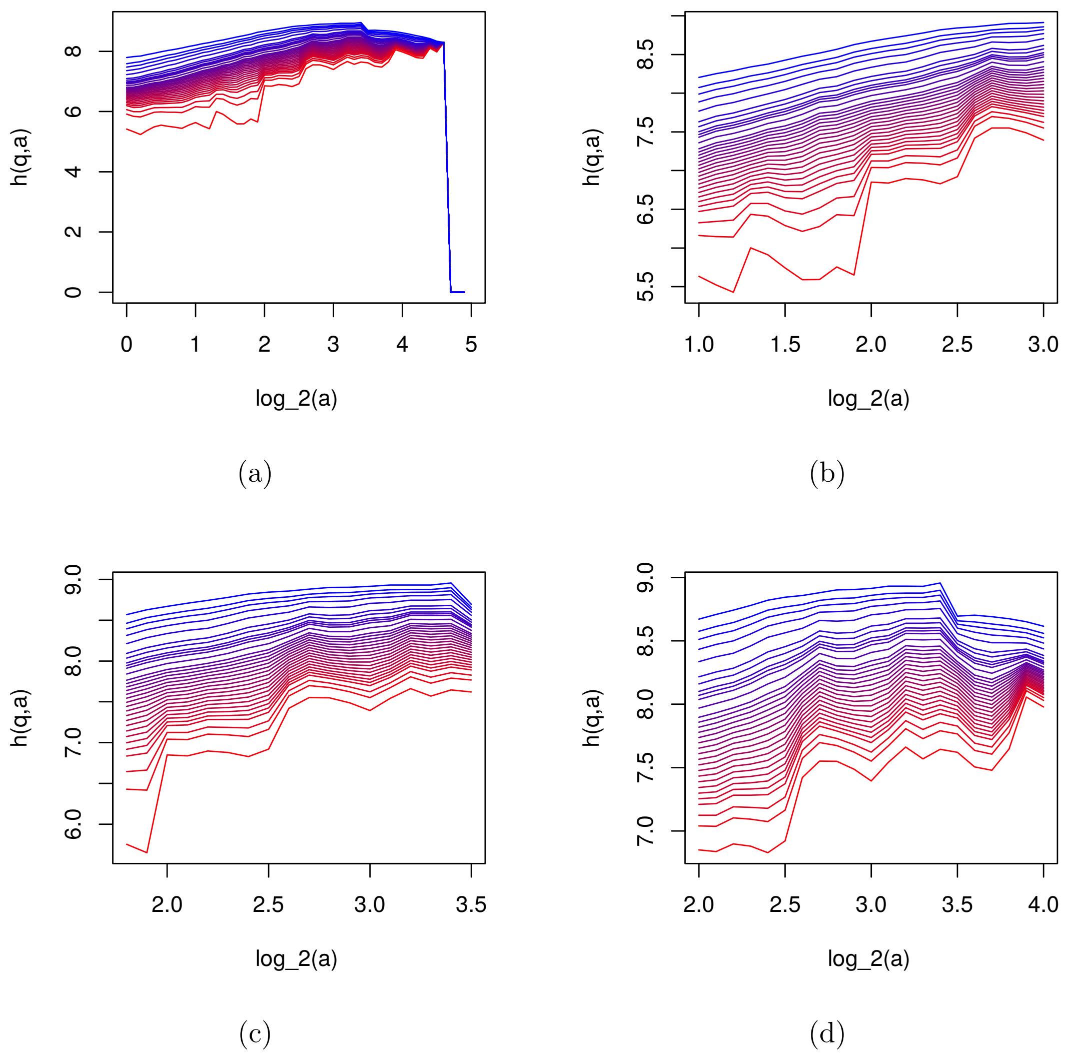
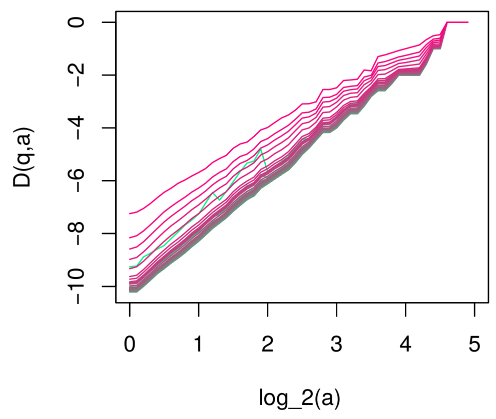
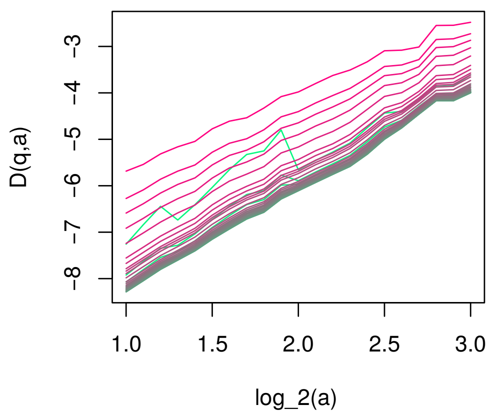


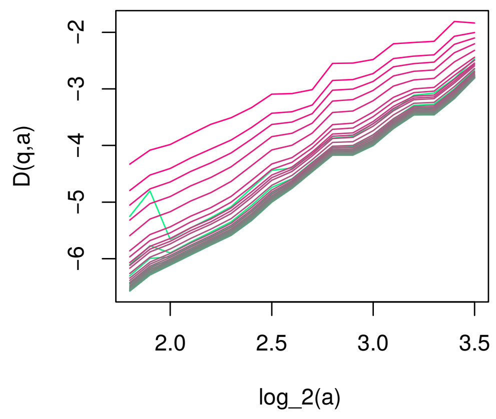
Figure 2.4: An example of a $h(q, a)$ curve.
This figure contains an example of an $h(q, a)$ curve, with Figure 2.4a providing an example of the full $h(q, a)$ graph. Figure 2.4b is a subset where $h(q, 1 \leq a \leq 3)$. The slope of $h(0, 1 \leq a \leq 3) = 0.6089$, which corresponds to dense (density fluctuations are spatially positively correlated) tissue in a mammogram. Figure 2.4c is a subset where $h(q, 1.8 \leq a \leq 3.5)$. The slope of $h(0, 1.8 \leq a \leq 3.5) = 0.5064$, which corresponds to disrupted tissue (high entropy, i.e. the density fluctuations are uncorrelated) in a mammogram. Figure 2.4d is a subset where $h(q, 2.0 \leq a \leq 4.0)$. The slope of $h(0, 2.0 \leq a \leq 4.0) = 0.3064$, which corresponds to fatty tissue (density fluctuations are spatially anti-correlated) in a mammogram. This is a graphical representation of a typical set of $h(q, a)$ curves from Equation 2.9. The q-values from blue to red are: $q =$ {-2, -1.5, -1, -0.8, -0.6, -0.5, -0.4, -0.3, -0.2, -0.1, 0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1, 1.2, 1.4, 1.6, 1.8, 2, 2.5, 3, 3.5, 4, 5}



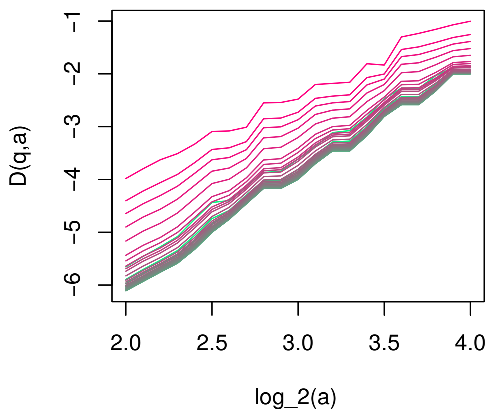
(a)



(b)



(c)



(d)

Figure 2.5: An example of a $D(q,a)$ curve.
This figure contains an example of a $D(q,a)$ curve, with Figure 2.5a providing an example of the full $D(q,a)$ graph. Figure 2.5b is a subset where $D(q, 1 \leq a \leq 3)$. The slope of $D(0, 1 \leq a \leq 3) = 2.171$, which corresponds to a 2D space. Figure 2.5c is a subset where $D(q, 1.8 \leq a \leq 3.5)$. The slope of $D(0, 1.8 \leq a \leq 3.5) = 2.123$, which again corresponds to 2D space. Figure 2.5d is a subset where $D(q, 2.0 \leq a \leq 4.0)$. The slope of $D(0, 2.0 \leq a \leq 4.0) = 2.171$. This is a graphical representation of a typical $D(q,a)$ set of curves from Equation 2.10. The q-values from red to turquoise are: $q =$ {-2, -1.5, -1, -0.8, -0.6, -0.5, -0.4, -0.3, -0.2, -0.1, 0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1, 1.2, 1.4, 1.6, 1.8, 2, 2.5, 3, 3.5, 4, 5}

2.6 Design Decisions for utilizing the WTMM Method

The WTMM method is accessed through a software package, written in C and TCL, called xsmurf. The xsmurf software is a signal and image analysis software package, written and designed for computers of the 1990s. These computers were far more limited on RAM and processing power when compared to modern machines. For that reason, xsmurf was not designed with portability or future-proofing in mind; instead, many of the design decisions revolved around RAM limitations and performance. The xsmurf software was also designed to run as a single-threaded, standalone application. More specifically, much of the code had circular references, global variables, and other organizational issues that made it difficult to modernize and maintain.

The xsmurf software package contains over a hundred thousand lines of C code and thousands of more lines of code written in TCL. Up until recently, there was no official version control software used to manage the development cycle of xsmurf. One of the first steps we took when designing the AWTMM method was to convert the xsmurf software into a library, libxsmurf. This overhaul is still in the works with our final goal of rewriting the software into OpenCL and eliminating the TCL dependencies. Much of the code has been reorganized, with the circular references being removed. However, because global variables are such an integral part of the software, they have been left in place until they can be addressed in future upgrades.

Recall from Chapter 2 note that the WTMM method chains the WTMMM together. This chaining process is the most involved process with complexity at worst $O(n^4)$. The computational complexity arises from searching the space above and below a particular WTMMM to locate the next WTMMM in the chain. One of the improvements we would like to make in the future would be to implement a more efficient search algorithm and to parallelize this code across multiple CPUs. We believe that this may be possible since each search could theoretically be run on a different thread.

Rewriting and optimizing the WTMM method for OpenCL would allow the code to be executed on a GPU. Processing the WTMM method on multiple GPU cores has the potential to increase our throughput. Even if GPU cores turn out not to be the best fit for the algorithm, we would still benefit from a streamlined version of the code which would run across multiple CPU threads or even field-programmable gate arrays (FPGAs). If, for instance, we could run the code on an FPGA, then the results would be orders of magnitude faster than our current implementation.

Another approach we could take toward improving the speed of the WTMM method would be to train a CNN (convolutional artificial neural network) on the image subregions. In theory, this CNN would be given an image subregion and would produce an H value based on the region. However, doing this would be, in some ways, a step backward. There is no guarantee that a CNN would produce a suitable result across multiple datasets. Even if we were to train a CNN on the mammograms successfully, there is no guarantee that this CNN would match the accuracy of the mathematical definition. Our choice, for at least this experiment, was to use something reliable and relatively slow, rather than to take the risk of using something that might be faster but might come at the expense of accuracy.

Alternatively, we could have used an ANN (artificial neural network) to classify only the h and D curves, the task that the AWTMM currently performs. At face value, this is not a bad idea. However, since the AWTMM algorithm performs nearly as accurately as a human, we decided that we would have little to gain from utilizing an ANN. We again made a similar design decision, opting to use the known system, since we had little to gain at the potential cost of time and accuracy.

CHAPTER 3
THE AUTOMATED WTMM METHOD

As discussed in Chapter 2, the 2D WTMM method requires curves to be fit to power-law exponent curves. The Automated WTMM (AWTMM) method was developed as a way to fit these curves. The invention of this algorithm is the author's core contribution to the Computer Science field. In this section, we will discuss this automated fitting algorithm and provide OpenCL/C code which can be used to run this operation. Additionally, a full listing of the code, with additional comments, can be found in Appendix B.

3.1 Linear Regression

Before a thorough explanation of the curve fitting procedure can be given, the supporting equations for this procedure need to be defined. This section relies heavily on basic statistical concepts, which can be found in any standard statistical textbook such as [58]. In particular, this section introduces the ideas and code for a basic linear model.

Since linear regression is a cornerstone of the AWTMM method, we will begin by discussing linear regression. For a set X, the mean, $\mu(X)$, is given as

$$\mu(X) = \sum_{i=1}^{k} \frac{x_i}{k}, \tag{3.1}$$

where $X = \{x_1, x_2, x_3, ..., x_k | k \in \mathbb{N}\}$. Listing 3.1 provides the code for this function.

```
double mean(__global double* aData,
            unsigned long aDataSize){
  double lRet = 0;

  for(unsigned long i = 0; i < aDataSize; i++){
    lRet += aData[i];
  }

  return lRet/aDataSize;
}
```

Listing 3.1: Code for Mean of Data.

The standard deviation ($\sigma(X)$), or the square-root of the variance, is defined as

$$\sigma(X) = \sqrt{\sum_{i=1}^{k} \frac{(x_i - \mu(X))^2}{k-1}}, \tag{3.2}$$

where $k > 1$. The code for the standard deviation is given in Listing 3.2.

```
double stdev(__global double* aData,
                unsigned long aDataSize){

  double lAverage = mean(aData,aDataSize);
  double lRet = 0;

  for(unsigned long i=0; i<aDataSize; i++){
    lRet += (aData[i]-lAverage)*(aData[i]-lAverage);
  }

  lRet = lRet/(aDataSize-1);
  return sqrt(lRet);

}
```

Listing 3.2: Code for Standard Deviation.

In addition to the mean and standard deviation, there is also a weighted version of both of these equations. These weights adjust the means and standard deviations by making some elements count less toward the final calculation, as opposed to the mean and standard deviation, which weigh each element equally. While these two statistics are not used in the linear regression model, they have the purpose of emulating human behavior with respect to the curve fitting model presented later in the chapter. The weighted mean is defined as,

$$\mu_w(X, W) = \frac{\sum_{i=1}^{k}(x_i w_i)}{\sum_{i=1}^{k}(w_i)}, \tag{3.3}$$

where w_i is the i^{th} element in the set of weights W, and $W \subset \mathbb{R}^k$. The code for the weighted mean is given in Listing 3.3.

```
double weighted_mean(__global double* aData,
                     __global double* aWeights,
                     unsigned long aArraySize){

  double lRet = 0;
  double lWeightSum = sum_vector(aWeights, aArraySize);

  for(unsigned long i = 0; i < aArraySize; i++){
    lRet += (aData[i]*aWeights[i])/lWeightSum;
  }

  return lRet;
}
```
Listing 3.3: Code for Weighted Mean.

The weighted standard deviation is calculated as

$$\sigma_w(X,W) = \sqrt{\frac{\sum_{i=1}^{k}(x_i - \mu_w(X,W))^2 w_i}{\sum_{i=1}^{k}(x_i w_i) - 1}}, \tag{3.4}$$

where w_i is the i^{th} element in the set of weights W, and $W \subset \mathbb{R}^k$. The code for the weighted standard deviation is given in Listing 3.4.

```
double weighted_stdev(__global double* aData,
                      __global double* aWeights,
                      unsigned long aSize){

  double lRet = 0;
  double lWeightedSum = sum_vector(aWeights,aSize);
  double lWeightedMean = weighted_mean(aData, aWeights, aSize);

  for(unsigned long i = 0; i < aSize; i++){
    lRet += ((aData[i]-lWeightedMean)*(aData[i]-lWeightedMean))
             *aWeights[i]/(lWeightedSum-1);
  }

  return sqrt(lRet);

}
```
Listing 3.4: Code for Weighted Standard Deviation.

Least squares regression is a mathematical tool used to model the linear system (defined in Listing 3.5) that best fits a set of ordered pairs (where an ordered pair is defined in Listing 3.6). Let A be a set of ordered pairs such that $A = \{\{x_1, y_1\}, \{x_2, y_2\}, ..., \{x_k, y_k\}\}$, where $k \in \mathbb{N}$. Additionally, $x_i \geq x_j \ \forall \ x_j \in A_x$, where A_x is the set of x components of A, namely, $A_x = \{x_1, x_2, ..., x_k\}$, and A_y is the set of all y components of A, namely, $A_y = \{y_1, y_2, ..., y_k\}$. We will denote the mean of A_x as $\bar{x}$ and the mean of A_y as $\bar{y}$. In linear regression, the coordinate $(\bar{x}, \bar{y})$ in the 2D plane will serve as the center of mass for the set of ordered pairs through which the line will be drawn, computed in Listing 3.7, lines 25 and 26.

First we compute the variance of A_x, which is the difference between sum of squares of A_x and $k\bar{x}^2$,

$$S_{xx}(A) = \sum_{i=1}^{k}(x_i^2) - k\bar{x}^2. \tag{3.5}$$

The code for S_{xx} is given in Listing 3.7, lines 43 and 44.

Next we compute the covariance of A_x and A_y.

$$S_{xy}(A) = \sum_{i=1}^{k}(x_i y_i) - k\bar{x}\bar{y}. \tag{3.6}$$

This calculation gives us information on how x and y are correlated.

Utilizing these two equations, we can then compute the slope of the regression line, $\hat{\beta}$, as the ratio between S_{xy} and S_{xx},

$$\hat{\beta}(A) = \frac{S_{xy}(A)}{S_{xx}(A)}. \tag{3.7}$$

In Listing 3.7, $\hat{\beta}$ is computed on lines 37-40.

We can then compute the y-intercept of the regression line, $\hat{\alpha}$, as

$$\hat{\alpha}(A) = \bar{y} - \hat{\beta}(A)\bar{x}. \tag{3.8}$$

$\hat{\alpha}$ is computed in lines 41 and 42 in Listing 3.7.

Next, we compute the variance of A_y,

$$S_{yy}(A) = \sum_{i=1}^{k} y_i^2 - k\bar{y}^2, \tag{3.9}$$

computed in Listing 3.7, lines 45 and 46. From here we can compare the variance of A_x to the variance of A_y to give us R,

$$R(A) = \hat{\beta}(A)\sqrt{\frac{S_{xx}}{S_{yy}}}. \tag{3.10}$$

This is computed in line 45 of Listing 3.7.

Finally we can use R to compute the R^2 value. The R^2 value provides an estimate of how well the linear model fits the data. The R^2 value is given as

$$R^2(A) = \left(\hat{\beta}(A)\sqrt{\frac{S_{xx}}{S_{yy}}}\right)^2, \tag{3.11}$$

where $R^2(A)$ provides us with an unsigned quantity which expresses how well the linear model fits the data set. The R^2 equation is computed on line 48 of Listing 3.7.

```c
struct LinearModel{

  double vBetaHat;          //Slope of the regression line.
  double vAlphaHat;         //Y-Intercept
  double vSxx;              //Standard deviation x
  double vSyy;              //Standard deviation y
  double vRSquared;         //R^2 value
  double vRVal;             //R value

  struct PointF vMeanXY;    //The mean of the x,y data.
  double vSumXY;            //Sum xy
  double vSumXSquared;      //Sum x^2
  double vSumYSquared;      //Sum y^2

  double vMinX;             //Min x range
  double vMaxX;             //Max x range

};
```

Listing 3.5: Structure for a Linear Model.

```c
struct PointF{
  double x; //The x coordinate of the ordered pair.
  double y; //The y coordinate of the ordered pair.
};
```

Listing 3.6: Structure for an (x,y) ordered pair.

```c
1  __kernel void calculate_least_squares_regression(
2         __global struct LinearModel* aDst,
3         __global struct PointF* aData,
4         unsigned long aDataSize,
5         unsigned long aFirst,
6         unsigned long aLast){
7
8    //Initialize the return argument
9    aDst->vSumXY = 0;
10   aDst->vSumXSquared = 0;
```

```
  aDst->vSumYSquared = 0;
  aDst->vBetaHat = 0;
  aDst->vAlphaHat = 0;
  aDst->vSxx = 0;
  aDst->vSyy = 0;
  aDst->vRSquared = 0;
  aDst->vRVal = 0;
  aDst->vMeanXY = {0,0};

  //Performing the linear model calculations

  unsigned long lSize = aLast-aFirst+1;

  for(unsigned long i = aFirst; i < aLast+1; i++){
    aDst->vMeanXY.x += aData[i].x/(double)lSize;
    aDst->vMeanXY.y += aData[i].y/(double)lSize;
    aDst->vSumXY = aDst->vSumXY
                 + (aData[i].x*aData[i].y);
    aDst->vSumXSquared = aDst->vSumXSquared
                 + (aData[i].x*aData[i].x);
    aDst->vSumYSquared = aDst->vSumYSquared
                 + (aData[i].y*aData[i].y);

  }


  aDst->vBetaHat = (aDst->vSumXY
                 - (lSize*aDst->vMeanXY.x*aDst->vMeanXY.y))
                   /(aDst->vSumXSquared
                   - (lSize*aDst->vMeanXY.x*aDst->vMeanXY.x));
  aDst->vAlphaHat = aDst->vMeanXY.y
                  - (aDst->vBetaHat*aDst->vMeanXY.x);
  aDst->vSxx = aDst->vSumXSquared
             - (lSize*aDst->vMeanXY.x*aDst->vMeanXY.x);
  aDst->vSyy = aDst->vSumYSquared
             - (lSize*aDst->vMeanXY.y*aDst->vMeanXY.y);
  aDst->vRVal = aDst->vBetaHat*sqrt(aDst->vSxx/aDst->vSyy);
  aDst->vRSquared = aDst->vRVal*aDst->vRVal;


  aDst->vMinX = aData[aFirst].x;
  aDst->vMaxX = aData[aLast].x;

}
```
Listing 3.7: Function for computing the linear model of a set of ordered pairs.

3.2 Defining $\mathcal{H}$ and $\mathcal{D}$

In this section, we will be discussing the procedure for processing the power-law fittings for h and D. Here we will consider the power-law fittings generated by the WTMM method from some image M (Chapter 4.1, Definition 1), where M is a real valued m by n matrix. The output of the power-law fittings of the 2D WTMM will be two sets of sets, which are called h and D.

One can consider h and D as sets of sets of xy coordinates. Each of these sets of sets are indexed by their Q-value, which represents the statistical order moment of the particular set of of ordered pairs. More specifically, $Q = \{q_1, q_2, ..., q_k\}$ is a finite subset of the real numbers. For this analysis, the set of q-values used is

$$Q = \{-2, -1.5, -1, -0.8, -0.6, -0.5, -0.4, -0.3, -0.2, -0.1, 0, 0.1, 0.2, 0.3, \\ 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1, 1.2, 1.4, 1.5, 1.6, 1.8, 2, 2.5, 3, 3.5, 4, 5\}. \quad (3.12)$$

We will call the function that generates the set of h curves $h : M, Q \to \{\mathbb{R}^2, \mathbb{R}^2, ..., \mathbb{R}^2\}$, where $|h(M, q = r)| = k$, for $k \in \mathbb{N}$, $r \in Q$. In other words, $h(M, q = r) = \{\{x_1, y_1\}, \{x_2, y_2\}, ..., \{x_k, y_k\}\}$. Further, we will call the function D that generates the set of curves $D : M, Q \to \{\mathbb{R}^2, \mathbb{R}^2, ..., \mathbb{R}^2\}$. In other words, $D(M, q = r) = \{\{x_1, y_1\}, \{x_2, y_2\}, ..., \{x_k, y_k\}\}$ where $|D(M, q = r)| = k$. In the context of this paper, one could consider a set of order pairs as defined by Listing 3.8.

```
struct PointFList{
  struct PointF vData[50];  //The array of ordered pairs
  unsigned long vDataSize;  //Number of ordered pairs <= 50
};
```

Listing 3.8: Structure for a set of ordered pairs.

To classify the power-law fittings, we must find the linear models for all possible subsets of the h and D charts. In code, we accomplished this by creating a list of subsets (an array of Listing 3.9), and computing the regression lines between the first and last elements identified by the SizeTPair from Listing 3.9. Generally for the WTMM method, these pairs are referred to as a_{min} and a_{max}. However, in this chapter will call them γ and ρ, respectively. This is done to prevent the mathematical definitions listed in this chapter from becoming too cumbersome.

```c
struct SizeTPair{
  unsigned long first;   //Index of the first element, gamma.
  unsigned long second;  //Index of the second element, rho.
};
```

Listing 3.9: A structure for holding pairs of indices of elements.

Next we can compute the linear model for either h or D utilizing the code in Listing 3.10. This code works by passing the set of points for some index $q \in Q$. Once the linear model is computed, the results are stored in a set of linear model data (Listing 3.11) at the index associated with q.

```c
__kernel void calc_curvedata_rl(struct RegressionList* aRL,
                                struct LinearModel* aLM,
                                struct SizeTPair* aBounds,
                                unsigned long* aPairSize,
                                struct PointFList* aData,
                                unsigned long* aDataSize){

  //The current row and column of the regression list
  unsigned long i = get_global_id(0);
  unsigned long j = i/(*aDataSize);
  unsigned long k = i%(*aDataSize);

  //Compute the regression of the subset range
  calculate_least_squares_regression(aLM+i,
                                     aData[k].vData,
                                     aData[k].vDataSize,
                                     aBounds[j].first,
                                     aBounds[j].second);

  //Set the values in the return variable
  aRL[j].vSlope[k] = aLM[i].vBetaHat;
  aRL[j].vIntercept[k] = aLM[i].vAlphaHat;
  aRL[j].vRSquared[k] = aLM[i].vRSquared;

  if(k==0){
    aRL[j].vAMin = aData[k].vData[aBounds[j].first].x;
    aRL[j].vAMax = aData[k].vData[aBounds[j].second].x;
  }


}
```

Listing 3.10: Code to compute the linear model of a subregion of points.

```
struct RegressionList{

  double vAMin;
  double vAMax;
  double vSlope[32];
  double vRSquared[32];
  double vIntercept[32];
  double vDeltaSlope[32];
  double vDeltaRSquared[32];
  double vDeltaIntercept[32];

};
```

Listing 3.11: Structure for an (x,y) ordered pair.

To define this system mathematically, we will have to discuss some of the finer details, such as indexing D and h by $q \in Q$. We can define the collection of all h values indexed by Q as $\mathcal{H}(M) = \{h(M, q = -2), h(M, q = -1.5), ..., h(M, q = 0), h(M, q = .1), ..., h(M, q = 4), h(M, q = 5)\}$.[1] Further, let $\mathcal{H}_{q,\gamma \leq x \leq \rho}(M)$ denote the set of subsets where each element of h contains only the x values which fall between γ and ρ. For example, suppose we have the set

$$\begin{aligned}\mathcal{H}_{Q,\gamma \leq x \leq \rho}(M) = &\{\{\{0, y_{q=-2,0}\}, \{.01, y_{q=-2,1}\}, ..., \{1, y_{q=-2,100}\}\},\\ &\{\{0, y_{q=-1.5,0}\}, \{.01, y_{q=-1.5,1}\}, ..., \{1, y_{q=-1.5,100}\}\},\\ &...,\\ &\{\{0, y_{q=5,0}\}, \{.01, y_{q=5,1}\}, ..., \{1, y_{q=5,100}\}\}\}.\end{aligned}$$

[1]While h and D have similar mathematical representations here, they represent entirely different physical phenomena.

Then

$$\begin{aligned}\mathcal{H}_{Q,.5\leq x\leq .75}(M) =&\{\{\{.50, y_{q=-2,50}\}, \{.51, y_{q=-2,51}\}, ..., \{.75, y_{q=-2,75}\}\},\\ &\{\{.50, y_{q=-1.5,50}\}, \{.51, y_{q=-1.5,51}\}, ..., \{.75, y_{q=-1.5,75}\}\},\\ &...,\\ &\{\{.50, y_{q=5,50}\}, \{.51, y_{q=5,51}\}, ..., \{.75, y_{q=5,75}\}\}\},\end{aligned}$$

and

$$\mathcal{H}_{(q=-2),.5\leq x\leq .75}(M) =\{\{.50, y_{q=-2,50}\}, \{.51, y_{q=-2,51}\}, ..., \{.75, y_{q=-2,75}\}\}.$$

Note that in this context, the second indexing number tells us what the element id is for the particular member. For example, $y_{q=5,50}$ informs us that this is the 50^{th} element of subset corresponding to $q = 5$.

Similar to the collection of h values, we can define the collection of all D indexed by the set Q as $\mathcal{D} = \{D(M, q = -2), D(M, q = -1.5), ..., D(M, q = 0), D(M, q = .1), ..., D(M, q = 4), D(M, q = 5)\}$. Additionally, let $\mathcal{D}_{q,\gamma\leq x\leq\rho}$ denote the set of subsets where each element of $\mathcal{D}$ contains only the x values which fall between γ and ρ. These definitions provide us with the mathematical framework needed to build up the mathematical model used in the Automated WTMM (AWTMM) method. In the following sections, we will discuss the mathematical foundation used to construct this model.

3.3 Fitness Parameters

The goal of any fitness function is to select the optimal solution from a set of potential solutions. In the case of this analysis, we desire to find the value of H, $h(M, q = 0)$, which best describes the rougness of M. This procedure, when performed by a human agent (HA), involves the selection of a region based on how the h and D curves behave. We chose the parameters listed later in this section as they best relate to the scientific interpretation of those characteristics the HA looked for while manually fitting the curves.

The first step the HA takes in isolating a region is finding a subregion where h_0 is relatively linear. Then the HA checks to make sure that the slope of h_0 (Hurst exponent) and D_0 (fractal dimension) are reasonable values, e.g., $\sim 0 < \hat{\beta}(\mathcal{H}_{q=0,\gamma\leq x\leq\rho}(M)) <\sim 1$ and $\sim 1.7 < \hat{\beta}(\mathcal{D}_{q=0,\gamma\leq x\leq\rho}(M)) \sim 2.3$, respectively. Next, the HA verifies that all of $\mathcal{H}_{q=0,\gamma\leq x\leq\rho}(M)$ are roughly linear. We will represent this set as $\mathcal{R}_{Q,\gamma\leq x\leq\rho}(M)$, where

$$\begin{aligned}\mathcal{R}_{Q,\gamma\leq x\leq\rho}(M) = \{&R^2(\mathcal{H}_{(q=-2),\gamma\leq x\leq\rho}(M)), R^2(\mathcal{H}_{(q=-1.5),\gamma\leq x\leq\rho}(M)),\\ &...,\\ &R^2(\mathcal{H}_{(q=0),\gamma\leq x\leq\rho}(M)), R^2(\mathcal{H}_{(q=.1,\gamma\leq x\leq\rho)}(M)),\\ &...,\\ &R^2(\mathcal{H}_{(q=4),\gamma\leq x\leq\rho}(M)), R^2(\mathcal{H}_{(q=5),\gamma\leq x\leq\rho}(M))\},\end{aligned} \tag{3.13}$$

where R^2 is from Equation 3.11. The HA inspects these slopes, paying the most attention to values closer to $q = 0$. In other words, the HA checks to ensure that $\mu_w(\mathcal{R}_{q=0,\gamma\leq x\leq\rho}(M), W)$ (Equation 3.3) is high. To simulate this, we assigned weights (W) to each curve. Let the set of weights (W), which are indexed by Q, be defined as

$$W = \{w_{q=-2}, w_{q=-1.5}, ..., w_{q=0}, w_{q=0.1}, ..., w_{q=4}, w_{q=5}\}. \tag{3.14}$$

An example set of weights could be

$$\begin{aligned}W = \{&0.1, 0.5, 1, 1.8, 2.6, 3, 4, 5, 7, 9, 10, 9, 8, 7, 6, 5, 4.6, 4.2, 3.8, 3.4, 3, 2.5, 2,\\ &1.83, 1.66, 1.33, 1, 0.5, 0.2, 0.2, 0.2, 0.2\}.\end{aligned}$$

The final thing the HA checks is that all of the slopes of $\mathcal{R}_{Q,\gamma\leq x\leq\rho}(M)$ are roughly the same, again lending more weight to the q-values closer to q_0. This was done to reduce the chance of misclassifying a multi-fractal signal. We will call this set of slopes

$\mathcal{M}_{Q,\gamma\leq x\leq\rho}(M)$,[2] where

$$\begin{aligned}\mathcal{M}_{Q,\gamma\leq x\leq\rho}(M) = \{&\hat{\beta}(\mathcal{H}_{(q=-2),\gamma\leq x\leq\rho}(M)), \hat{\beta}(\mathcal{H}_{(q=-1.5),\gamma\leq x\leq\rho}(M)),\\ &...,\\ &\hat{\beta}(\mathcal{H}_{(q=0),\gamma\leq x\leq\rho}(M)), \hat{\beta}(\mathcal{H}_{(q=.1,\gamma\leq x\leq\rho)}(M)),\\ &...,\\ &\hat{\beta}(\mathcal{H}_{(q=4),\gamma\leq x\leq\rho}(M)), \hat{\beta}(\mathcal{H}_{(q=5),\gamma\leq x\leq\rho}(M))\},\end{aligned} \tag{3.15}$$

where $\hat{\beta}$ is from Equation 3.7. In other words, we want to ensure that the weighted standard deviation of this set of slopes, $\sigma_w(\mathcal{M}_{Q,\gamma\leq x\leq\rho}(M), W)$ (Equation 3.4), is small.

3.4 Fitness Function

Now that we've established where the fitness parameters originated from, we can combine these variables into a single equation, one that will yield the optimal region where γ and ρ best match the HA's selection. This fitness score should be high when it is closer to the HA selection. Further, this selection procedure operates under the assumption that there are multiple sufficient answers, all of which will yield similar values, but that there is one best answer.

This fitness function should maximize $\mathcal{R}_{q=0,\gamma\leq x\leq\rho}(M)$ (Equation 3.13) and $\mu_w(\mathcal{R}_{Q,\gamma\leq x\leq\rho}(M), W)$ (Equation 3.3), giving equal priority to both of these values. Therefore, let R_e be the scaled Euclidean distance between these two values, where

$$R_e = \frac{1}{\sqrt{2}}\sqrt{(\mathcal{R}_{q=0,\gamma\leq x\leq\rho}(M))^2 + (\mu_w(\mathcal{R}_{Q,\gamma\leq x\leq\rho}(M), W))^2}. \tag{3.16}$$

Next we scale $\hat{\beta}(\mathcal{H}_{q=0,\gamma\leq x\leq\rho}(M))$ (Equation 3.7), $\hat{\beta}(\mathcal{D}_{q=0,\gamma\leq x\leq\rho}(M))$ and $\sigma_w(\mathcal{M}_{Q,\gamma\leq x\leq\rho}(M), W)$ (Equation 3.4) between 0 and 1 using the scaling function,

$$scale(x, x_{min}, x_{max}) = \frac{x - x_{min}}{x_{max} - x_{min}}. \tag{3.17}$$

The scale function is defined in code in Listing 3.12.

[2] Note that this is not the same $\mathcal{M}$ as used in Chapter 2.

```
double scale_value(double aVal, double aMin, double aMax){
  return (aVal-aMin)/(aMax-aMin);
}
```

Listing 3.12: The scale function.

We compute these scaled values as

$$H_s = scale(\hat{\beta}(\mathcal{H}_{q=0,\gamma\leq x\leq\rho}(M)), h_{min}, h_{max}), \tag{3.18}$$

$$D_s = scale(\hat{\beta}(\mathcal{D}_{q=0,\gamma\leq x\leq\rho}(M)), d_{min}, d_{max}), \tag{3.19}$$

and

$$S_s = scale(\sigma_w(\mathcal{M}_{Q,\gamma\leq x\leq\rho}(M), W)), s_{min}, s_{max}), \tag{3.20}$$

where h_{min} and h_{max} are the minimum and maximum Hurst exponent, d_{min} and d_{max} are the minimum and maximum fractal dimension, and s_{min} and s_{max} are the minimum and maximum weighted standard deviations for the slopes of $\mathcal{H}_{q=0,\gamma\leq x\leq\rho}$. These three scaled equations are defined in lines 28-36 of Listing 3.15.

It is important to observe that for H_s, D_s, and S_s, any value which falls between 0 and 1 is just as valid as any other value which falls between 0 and 1. For example, $H_s = 0.5$ is just as valid as $H_s = 0.25$. It is only when we arrive at values outside of the range $[0, 1]$ that we would like to penalize the fitness. Therefore, we will define a function that fixes this fitness so that any valid value will be 1, and penalize the fitness otherwise. Let $fix(x)$ be defined as

$$fix(x) = \begin{cases} x: & x < 0 \\ 1: & 0 \leq x \leq 1 \\ 1-x: & x > 1 \end{cases}. \tag{3.21}$$

This function is defined as 'calc_fit' in Listing 3.13.

```
double calc_fit(double aVal, double aMin, double aMax){

  double lVal = scale_value(aVal,aMin,aMax);
  double lRet = 0;

  if(lVal > 1){
    lRet = 1-lVal;
  } else if (lVal < 0){
    lRet = lVal;
  } else {
    lRet = 1;
  }

  return lRet;
}
```
Listing 3.13: Function for computing the fixed fitness.

Finally, let us combine and scale these 5 variables together to form the scaled fitness function. This scaled fitness function is defined in Listing 3.15, line 38 and mathematically as,

$$fit_{\gamma \leq x \leq \rho}(M, G) = \frac{R_e + H_s + D_s + S_s}{4}, \tag{3.22}$$

where G is the set of weights and scaling parameters:

$$G = \{W, \{h_{min}, h_{max}\}, \{d_{min}, d_{max}\}, \{s_{min}, s_{max}\}\}. \tag{3.23}$$

Now that we can compute the fitness of a particular γ and ρ, we can extend this computation across the set of all valid permutations of γ and ρ to find the optimal subset. Let the set of all fitness functions for all permutations of γ and ρ be defined as

$$\mathcal{V} = \{fit_{\gamma \leq x \leq \rho}(M, G) | \{\gamma, \rho\} \in \mathcal{X} \times \mathcal{X} | \rho - \gamma \geq 1\}, \tag{3.24}$$

where $\mathcal{X}$ is set generated by the function
$g(x_0, k, dx) = \{x_0 + (dx)0, x_0 + (dx)1, x_0 + (dx)2, ..., x_0 + (dx)(k-1), x_0 + (dx)k\}$ and $\mathcal{X} = g(0, 100, .1)$. Further, let the elements of $\mathcal{V}$ be indexed such that $v_n \in \mathcal{V}$ and n is the index of the permutation for $\{\gamma', \rho'\}_n \in \mathcal{X} \times \mathcal{X}$.

To find the optimal element of $\mathcal{V}$, we choose the i^{th} element from $\mathcal{V}$ such that $v_i \geq v_j \ \forall \ v_j \in \mathcal{V}$ and let $v_{max}(M, G) = (fit_{\gamma \leq x \leq \rho}(M, G))_i$ represent the element with the greatest fitness. This operation would be performed by sorting a set of classified curve data (Listing 3.16) by the fitness. Further, the Hurst exponent and fractal dimension for M are

$\hat{\beta}(\mathcal{H}_{q=0,\gamma'\leq x\leq\rho'}(M))$ and $\hat{\beta}(\mathcal{D}_{q=0,\gamma'\leq x\leq\rho'}(M))$ respectively. To classify the group that the Hurst exponent falls into, we use the classification function

$$hgroup(x,f) = \begin{cases} B & x < 0.45, f \geq \tau \\ Y & .45 \leq x \leq .55, f \geq \tau \\ R & x > .55, f \geq \tau \\ N & f < \tau \end{cases}, \tag{3.25}$$

where f is the fitness of the classifier and τ is global minimum valid fitness threshold. [3] For this research, B, Y, R, and N refer to the classification groups $H < 0.45$, $0.45 \leq H \leq 0.55$, $H > 0.55$, and no-scaling, respectively. Historically, these groups were color coded with the colors: Blue, Yellow, Red, and Gray. However, for this research, the color green is often used to represent the Y group. This is done because images contain a green channel instead of a yellow channel. To represent the green pixels as yellow requires additional image processing. For further details on these groups, see Chapter 2. For example, $hgroup(\hat{\beta}(\mathcal{H}_{q=0,\gamma'\leq x\leq\rho'}(M)) = .35, v_{max} \geq \tau)$ is classified as group B. Likewise, $hgroup(\hat{\beta}(\mathcal{H}_{q=0,\gamma'\leq x\leq\rho'}(M)) = .35, v_{max} < \tau)$ is classified as group N. Note that this is a slight abuse of the notation that we laid out above, but this serves to illustrate the classification system. The code definition for hgroup is given as a combination of Listing 3.14 and Listing 3.15, lines 77-83.

```c
char classify_h_group(double aHValue){

  char lRet = 0x00;

  if(aHValue > .55){
    lRet = 'R';
  } else if (aHValue < .45){
    lRet = 'B';
  } else {
    lRet = 'Y';
  }
```

[3]For this paper we chose $\tau = .75$, which represents the cutoff where any one of the fitness parameters resulted in a value such that $fit(x) = 0$.

```

  return lRet;

}
```

Listing 3.14: Function for classifying hgroup.

```
void classify_curvedata(__global struct CurveData* aDst,
            double aMinWeightedR2,
            double aMinHValue,
            double aMaxHValue,
            double aMinDValue,
            double aMaxDValue,
            double aThreshWeightedH){

  //Initialize our return values
  bool lSmallDelta = false;
  bool lBadH = false;
  bool lBadHStdev = false;
  bool lBadD = false;
  bool lNoScaling = false;
  bool lBadR2 = false;
  bool lBadWR2 = false;
  aDst->vSuitable = true;


  for(int i = 0; i < 5; i++){
    aDst->vComment[i] = '-';
  }
  aDst->vComment[5] = 0;

  //Compute the fitness of the curve
  double lFitCalc = sqrt((aDst->vR2Value*aDst->vR2Value)
            +(aDst->vWeightedR2*aDst->vWeightedR2))/1.414213562;
  double lScaleH = calc_fit(aDst->vHValue,
                aMinHValue,
                aMaxHValue);
  double lScaleD = calc_fit(aDst->vDValue,
                aMinDValue,
                aMaxDValue);
  double lScaleWSDH = calc_fit(aDst->vStdevWeightedH,
                  0,
                  aThreshWeightedH);

  aDst->vFitness = (lScaleH+lFitCalc+lScaleD+lScaleWSDH)/(4.0);


  //Is the h-value suitable?
  if(aDst->vHValue < aMinHValue || aDst->vHValue > aMaxHValue){

    aDst->vSuitable = false;
    aDst->vComment[2]='H';

  }

  //Is the weigthed standarad deviation of h suitable?
  if(aDst->vStdevWeightedH > aThreshWeightedH){
    aDst->vSuitable = false;
    aDst->vComment[3]='S';
  }

  //Is our D-value suitable?
  if(aDst->vDValue < aMinDValue || aDst->vDValue > aMaxDValue){
    aDst->vSuitable = false;
    aDst->vComment[4]='D';
  }

  //Is our R^2 value suitalbe?
  if(aDst->vR2Value < aMinWeightedR2){
    aDst->vSuitable = false;
    aDst->vComment[0]='R';
  }

  //Is our weighted R^2 value suitable?
  if(aDst->vWeightedR2 < aMinWeightedR2){

    aDst->vSuitable = false;
    aDst->vComment[1]='W';
  }

  //Were any of the suitablity conditions not met?
  //If so, classify as No scaling (N)
  if(!aDst->vSuitable){
    aDst->vGroup = 'N';

  //Otherwise classify the h-value
  } else {
    aDst->vGroup = classify_h_group(aDst->vHValue);
  }

}
```

Listing 3.15: Function for classifying power-law fittings

```
struct CurveData{
  double vHValue;            //h(0,a)
  double vDValue;               //D(0,a)
  double vR2Value;              //How well the linear model fit

  double vAMin;                 //gamma for h(q,a), D(q,a) curves
  double vAMax;                 //rho for h(q,a), D(q,a) curves

  double vFitness;              //The fitness of the curve

  char vGroup;                  //R,Y,B,N
  char vComment[100];           //No scaling reason

  double vWeightedDelta;        //The weigthed delta
  double vWeightedH;            //Weighted h(0,a)
  double vWeightedD;            //Weighted D(0,a)
  double vWeightedR2;           //Fit of weighted linear model
  double vAvgDelta;             //Average data spread
  double vAvgH;                 //What is the average h-value
  double vAvgD;                 //What is the average D-value
  double vAvgR2;                //What is the average R^2

  double vStdevWeightedDelta; //stdev of the weighted delta
  double vStdevWeightedH;     //stdev of the weighted h(0,a)
  double vStdevWeightedD;     //stdev of the weighted D(0,a)
  double vStdevWeightedR2;    //stdev of the weighted D(0,a)
  double vStdevDelta;         //stdev of the weighted linear fit
  double vStdevH;             //stdev of the average h-value
  double vStdevD;             //stdev of the average D-value
  double vStdevR2;            //stdev of average R^2

  bool vSuitable;               //Flag for vaild curvedata

};
```
Listing 3.16: Structure for contaning classified power-law curve.

3.5 AWTMM Method Complexity and Design Choices

The complexity of the AWTMM method algorithm is at worst $O(n^2)$, based on the number of subsets used to compute the regression lines. However, given the embarrassingly parallelizable nature of the data, each regression line is independent of the others and can be computed on a different thread. Given enough GPU threads, all of the regression lines could be computed simultaneously for a given image subregion. One of the improvements to the algorithm we implemented early on was to generate a list of all valid subsets. This list was then referenced in each run of the AWTMM algorithm, allowing us to save computation time while computing the subsets of the regression lines. Precomputing the subsets instead of running the subsets in a double for-loop, allowed us to reduce the coding complexity of the algorithm. The most operationally complex process of the AWTMM algorithm is computing the regression line for each of the subsets of the h and D curves. Each of these algorithms (See Chapter 2) used in computing the regression lines have a complexity of $O(n)$. The final operation in the AWTMM, sorting, has a complexity of $O(n \log(n))$.

Before we implemented the fitness function, we used a binary decision tree to determine if the particular subregion was valid or not. Inside this loop, we also performed a check to see if the particular graph subregion had the greatest weighted R^2 value. The graph subregion with the greatest weighted R^2 which was still valid was used to classify the image subregion. If no such valid subregion existed, then the image subregion was classified as N (no-scaling). The disadvantage of this classification method is that it only provided information on the passing graph subregion. One could not ascertain how 'close' a graph subregion was to passing, only that it failed for one or more reasons.

While binary tree classification is fast and efficient, it lacked the overall flexibility we needed to classify the data. Additionally, this new classification schema could allow us to use a more robust set of parameters to classify the graph subregion. For instance, we could include variables like the standard deviation of the slopes of the h graphs. Using parameters that a human might not consider could potentially give us a more robust way of classifying subregions. This change took the number of operations needed to classify the calculated graph subregion (see Listing 3.15) from around 15 operations to around 30 operations.[4] Since the code was ported to OpenCL and distributed across multiple GPU cores, the overall run time of the algorithm decreased instead of increased. As it stands, the AWTMM is a lean algorithm. There are few places left for optimization. The

[4]These are C operations and not assembly operations.

AWTMM algorithm has reached a high level of efficiency, as it was redesigned from the ground up with performance and parallelization as the highest priorities. However, since the AWTMM was written in OpenCL, we could flash this algorithm to an FPGA, which would allow it to run at very near the speed of the logic gates.

We chose to utilize OpenCL so that we could take advantage of both GPUs and CPUs of different architectures. For instance, we were able to successfully compile and run the AWTMM method on an ARM processor and utilize both its GPU and CPU cores. This flexibility to move between architectures and platforms allows us to be agile with regard to our changing hardware needs.

The Automated Sliding Window AWTMM (ASWAWTMM) is written in C++ and makes calls to the OpenCL code, passing the data to the CPU. In addition to OpenCL, ASWAWTMM utilizes OpenMPI to start a supervisor thread that manages a collection of worker nodes. We chose to utilize OpenMPI for a few reasons. OpenMPI is a mature, easy to implement, fast, open-source software package, and comes highly recommended by experts in the field of high-performance computing. The implementation of the sliding window is discussed in Chapter 5.

After the supervisor thread launches, the worker nodes request subregions of the image and use the WTMM method to compute the h and D curves. The thread then passes the results of these computations back to the supervisor thread. After all of the regions are computed, the data for these curves are then passed to the AWTMM algorithm via OpenCL calls, where they are classified into one of the four groups (B, Y, R, or N). Since OpenMPI handles the message passing and the AWTMM algorithm computations are performed using OpenCL, there are only a handful of areas left to optimize. Steps could be taken to slim down the codebase and the dependencies ASWAWTMM uses. One such step would be the removal of vestigial code. Since the ASWAWTMM was developed as a single thread application, there still exist subroutines that were designed for this environment.

Eliminating these old pieces of code would simplify the code base and make it easier to maintain and upgrade.

In the original ASWAWTMM, we did not implement any form of multi-threading. This meant that each mammogram was computed utilizing only a single thread. That is, each image subregion was computed serially. At the time, the WTMM method was being utilized as a library, known as libxsmurf, to facilitate shared memory between the AWTMM software package and the WTMM. However, libxsmurf made heavy use of global variables, meaning that it would be challenging to implement a thread-safe version of the library.[5]

To work around the thread-safety issue, we decided to detach libxsmurf and opted instead to pipe commands to instances of xsmurf controlled by worker nodes. It is important to note that xsmurf is an older piece of software, designed to run on computers from the 1990s. The xsmurf software makes heavy use of disk caching to prevent overflowing system memory. To get around the bottlenecks associated with reading and writing data to disk drives, we decided to make a RAM disk to write all temporary files that xsmurf created during its runs. Further, we stored the h and D curves the WTMM method generated to the RAM disk. In this way, we create a simple and relatively fast IPC (inter-process communication) system. This system utilized code that already existed in both the xsmurf and AWTMM software packages, making it a nearly drop-in place solution.

The speedup curve of the xsmurf+AWTMM software package looks linear. That is, the tasks can be evenly divided between multiple threads, all of which will process tasks at around the same rate that a single CPU would process a task. Some efficiency is lost due to message passing, reading data, and data storage, meaning that a perfect linear speedup curve is not possible at this time. That is, the speedup is highly dependent on supercomputer cluster architecture. It is conceivable that if the WTMM method can be

[5]The xsmurf software package was originally designed as a standalone signal processing software package. We converted it into a library specifically to make function calls directly from the library instead of piping command to xsmurf.

rewritten to be more efficient, then the whole process may be able to be computed in the CPU cache. If this were the case, then we may observe a super-linear speedup curve.

CHAPTER 4
MASK ANALYSIS

4.1 Mask Properties

In Chapters 2 and 3, we informally defined the idea of a digital image. In this chapter, we will formally define this concept as well as some of the operations that can be performed on digital images.[1] A digital image (image for short) can be represented as a matrix whose elements are in $[0..1]$. More formally,

Definition 1 *A digital image B is an m by n matrix whose elements $B_{i,j} \in [0..1]$ for all $1 \leq i \leq m$ and $1 \leq j \leq n$. These $B_{i,j}$ values correspond to pixel intensities.*

The processing capabilities of the AWTMM allowed us to create a fine grid of both H-values and their associated H-groups. This schema allowed us to not only investigate the counts of the H-groups, but also to investigate the geometry of clusters of H-groups. In particular, we examined the groups of tissue classified as dense and disrupted, that is $H > 0.55$ and $0.45 \leq H \leq 0.55$, respectively. In other words, we segmented the image by H-group clusters, where segmentation in this context is defined as:

Definition 2 *Segmentation is the process of partitioning a digital image into multiple disjoint sets of pixels.*

An effective way of segmenting an image is achieved by creating what is known as an image mask. An image mask defines which pixels in the image are considered the foreground. Mathematically,

Definition 3 *A mask M is an m by n matrix whose elements $M_{i,j} \in \{0, 1\}$ for all $1 \leq i \leq m$ and $1 \leq j \leq n$.*

[1]Many concepts and definitions laid out in this chapter were taken from [59]. Many other concepts here were derived independently, though sources to prior work are listed for the convenience of the reader.

A mask can be created from an existing image through the use of a thresholding function. A thresholding function is any function T where $T : [0..1] \to \{0, 1\}$. Suppose we have a thresholding function

$$T_{min,max}(x) = \begin{cases} 1 & min \leq x \leq max \\ 0 & \text{Otherwise} \end{cases}, \tag{4.1}$$

where min and max are the upper and lower bounds of the thresholding function.[2] If we apply $T_{0.0,0.45}$, $T_{0.45,0.55}$, and $T_{0.55,1.0}$ to the matrix

$$B = \begin{bmatrix} 0.1 & 0.3 & 0.5 & 0.3 & 0.1 \\ 0.1 & 0.5 & 0.8 & 0.5 & 0.1 \\ 0.1 & 0.5 & 0.8 & 0.5 & 0.1 \\ 0.1 & 0.5 & 0.8 & 0.5 & 0.1 \\ 0.1 & 0.3 & 0.5 & 0.3 & 0.1 \end{bmatrix},$$

then the results will be

$$T_{0.0,0.45}(B) = \begin{bmatrix} 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 \end{bmatrix}, \qquad T_{0.45,0.55}(B) = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}, \text{and}$$

$$T_{0.55,1.0}(B) = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

For this paper, we built our masks by categorizing the data generated by the AWTMM. In particular, the thresholding function we used is given by Equation 3.25, where the

[2]Other examples of thresholding in digital images can be found in [60–62]

different groups (B, Y, R, and N) represent the different masks. An example of this masking process can be seen in Figure 4.1.

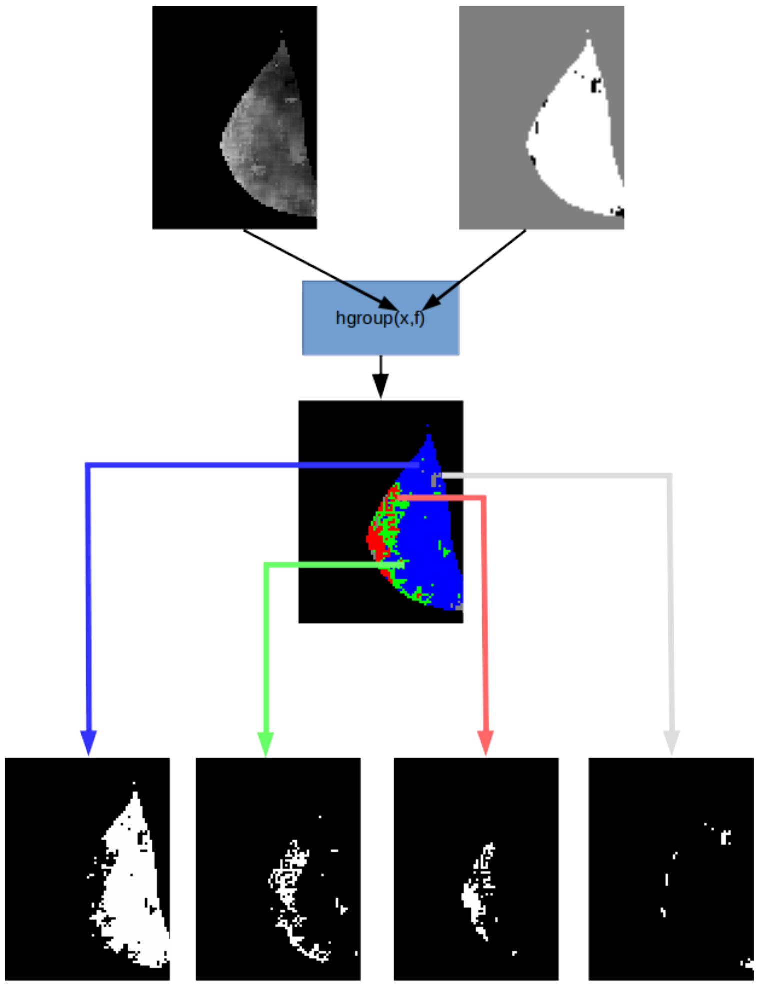


Figure 4.1: An example of generating masks from a set of H-values.
The top left image depicts the H-values (as grayscale pixels) computed for a mammogram. The top right image depicts the regions with a valid fitness. The white area denotes regions with valid fitness while the black area denotes regions with invalid fitness. The gray area represents regions which were not analyzed and are not being fed into the hgroup function (Equation 3.25). The color mammogram is a graphical representation of the classified groups, with blue, green, red, and gray being classified as B,Y,R, and N, respectively. The bottom row of images represent the masks generated for each of the groups B, Y, R, and N, from left to right.

4.2 Defining Clusters

Now that we have discussed how we obtain masks from an image, we can compute useful information from the elements contained in the mask. In the case of this analysis, we define a cluster as a set of 4-connected pixels which are disjoint from all other sets of 4-connected pixels. Here 4-connectivity refers to the number of adjacent, or neighboring mask elements a particular mask element has. Consider matrix Z,

$$Z = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

We say that $z_{2,2}$ has 2 neighbors and that $z_{3,3}$ has 4 neighbors. Mathematically, we define a cluster as the union of its boundary and interior. The interior of a cluster is defined as

Definition 4 *Let $k_{i,j} \in K$ where K is a mask. If $k_{i-1,j} + k_{i,j-1} + k_{i,j+1} + k_{i+1,j} < 4$, then (i,j) is in the boundary of the mask. For the cases where $k_{i-1,j}, k_{i,j-1}, k_{i,j+1},$ or $k_{i+1,j}$ fall outside of the mask, then their values are considered to be 0. If (i,j) is not in the boundary, then it is in the interior.*

Further, let the boundary be defined as

Definition 5 *Denote the boundary of K as $\partial(K)$, with $|\partial(K)| = \beta$ elements. The boundary of K consists of the points $\alpha_1, \alpha_2, \alpha_3, ..., \alpha_\beta$ with fwer than 4 neighbors, arranged in clockwise order as the boundary is traversed.*

The next property, perimeter, can be defined using the topological ideas of the digital plane and boundary.[3] For this research, we obtained the boundary of the object through a modified digital plane topology. For this modification, we linearly scaled the image by a

[3]Note that using the topological definition to find the interior and boundary of digital objects is not new and can be found in such references as [63, 64].

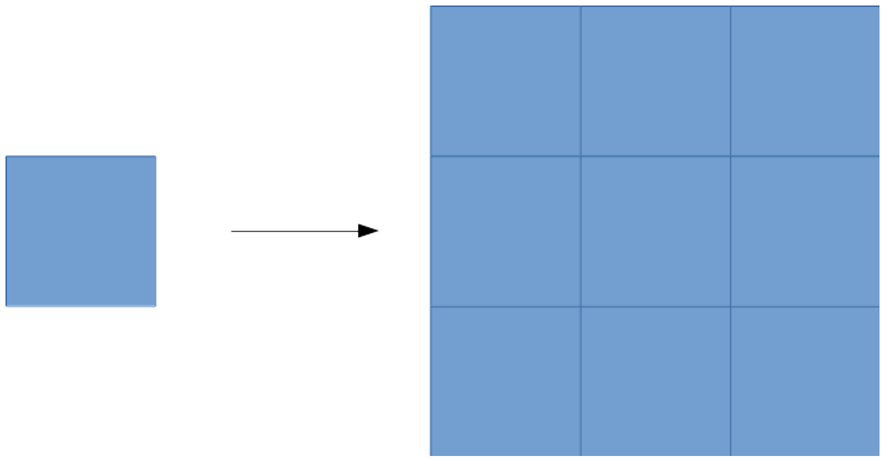


Figure 4.2: Scaling a pixel up by a factor of 3 in both the x and y directions.

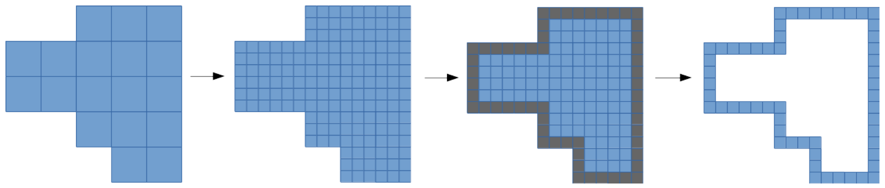


Figure 4.3: Extracting the boundary from a image mask.

factor of 3 in both the x and y directions. This transformed single pixels into blocks of 9 pixels, as shown in Figure 4.2. From here, we identified all pixels which had fewer than 4 neighbors. These pixels were considered our boundary points. In this way, we were able to create a list of boundary points for each shape in the mask. Internal boundaries were identified and removed by computing whether any of the boundary points of a polygon were contained in another polygon. A visualization of this procedure can be seen in Figure 4.3. The remaining set of boundary points were scaled back down by a factor of 3 in both the x and y directions, where they were utilized for the geometric equations. Further, the geometric properties (i.e., area, perimeter, diameter, etc.) of the scaled boundary were also assessed.

Up to this point, masks have been considered for images with only a single object in the foreground and objects which do not contain any internal boundaries. However, sometimes it is necessary to create a mask of an image which does not adhere to the above specifications.

Definition 6 *Consider the polygon $\mathcal{Q}$, where all the vertices of $\mathcal{Q}$ lie inside a polygon $\mathcal{P}$ and the boundary of $\mathcal{Q}$ does not intersect the boundary of $\mathcal{P}$. Then the polygon $\mathcal{Q}$ is said to be an internal boundary of $\mathcal{P}$.*

Figure 4.4a depicts an object with an internal boundary (in orange). While internal boundaries can provide important information, we only considered the geometric properties of external boundaries for this research. Henceforth all clusters considered will be non-internal boundaries, and all masks considered will be mathematically equivalent to a filled in disk. This conversion process is depicted in Figure 4.4 with Figure 4.4b representing the filled in mask of Figure 4.4a [59].

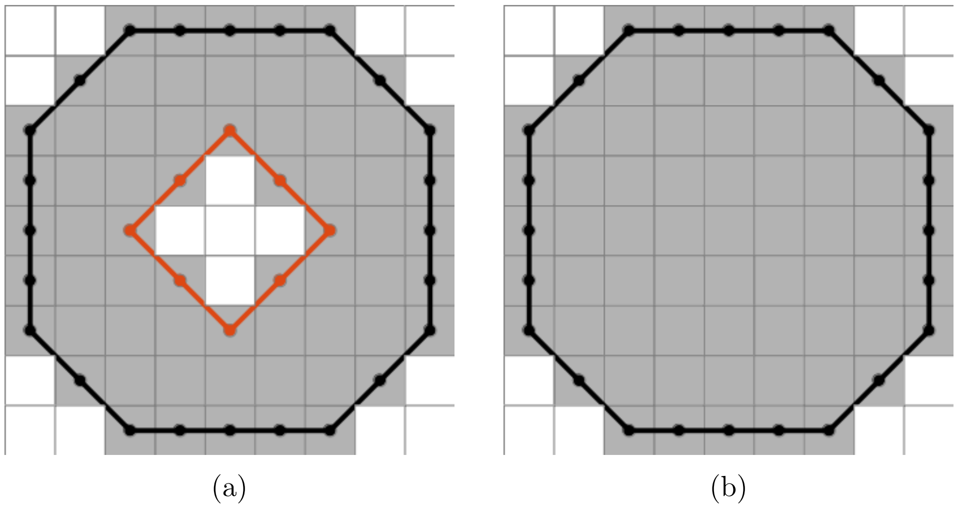


Figure 4.4: Example of an internal boundary.
An illustration of a mask with an internal boundary, alongside the same mask drawn topologically as a disk. (a) A circle with an internal boundary (orange) and a boundary (black). (b) The shape from (a) with the internal boundary filled in.

Suppose the mask K of an image B contains multiple disjoint boundaries $\mathcal{K} = \{\partial(K)_1, \partial(K)_2, ..., \partial(K)_r\}$. Each of these disjoint boundaries represent a sub-mask of the mask K. Each sub-mask can be treated as its own mask in regards to any of the computations laid out in this chapter (A, P, etc ...).

4.3 Calculating Shape Properties

In the context of this paper, a mask provides a simplified means of computing geometric properties of the object contained in the original image. That is, we have the ability to compute dimensional properties which the objects contained in the mask exhibit. Most of the properties we can extract from these clusters are derived from basic geometric properties such as area (A), perimeter (P), and diameter (D).

4.3.1 Area

Since pixel area is the most straightforward of these computations, we will begin with this feature. Suppose K is a mask with m rows and n columns. The pixel area function $A_{px} : K \to \mathbb{R}$ is defined as

$$A_{px}(K) = \sum_{i=1}^{m}\sum_{j=1}^{n}(k_{i,j}), \text{ where } k_{i,j} \in K \text{ and } k_{i,j} = 1 \text{ or } 0. \tag{4.2}$$

In other words, A_{px} is the sum of all elements of the matrix K.[4]

In addition to the pixel area (A_{px}), we can also compute the area of the boundary polygon. The area of the polygon is the absolute value of the sum of the cross products divided by two of the boundary points [65], or

$$A = \frac{\left|\sum_{i=1}^{\beta-1}(\alpha_i \times \alpha_{i+1}) + (\alpha_\beta \times \alpha_1)\right|}{2}, \tag{4.3}$$

where $\alpha_i \in \partial(K)$ and $i = 1, 2, ..., \beta$.

The convex area (A_{cvx}), is calculated in a similar way to polygon area, but instead utilizes the vertices associated with the convex hull, i.e.

$$A_{cvx} = \frac{\left|\sum_{i=1}^{\beta-1}(\alpha_i \times \alpha_{i+1}) + (\alpha_\beta \times \alpha_1)\right|}{2}, \tag{4.4}$$

where $\alpha_i \in \partial_{\text{cvx}}(K)$ and $i = 1, 2, ..., \beta$.

[4]While area of a digital object is a trivial measure, more detail on this particular metric can be found in [60–62]

The last area measurement we can compute is the area of the minimal bounding rectangle, A_{mbr}. The minimal bounding rectangle for a cluster in matrix K, $mbr(K)$, is defined as the smallest possible rectangle that contains all of the boundary points of the cluster in K. Since the minimal bounding rectangle is a polygon, we compute A_{mbr} as

$$A_{mbr} = A(mbr(K)). \tag{4.5}$$

4.3.2 Perimeter

To compute the perimeter of the cluster, we must consider the 'length' of the boundary. However, simply counting up the number of elements in the boundary would not necessarily yield an accurate perimeter. This is because if two boundary points are diagonal from each other, the distance between their centroids is $\sqrt{2} \approx 1.41$, whereas the distance between centroids of elements which share a an edge is 1. Take for example Figure 4.5 [59]. The number of pixels which make up the boundary is 14. However, the sum of the distance between all of the centroids is around 17.314. For that reason, we will define the perimeter as the sum of the distance between centroids instead of the raw count of boundary elements.[5] More precisely, let the boundary elements of a mask K correspond to vertices of the mask. Let $\alpha_i \in \partial(K)$, where $i = 1, 2, ..., \beta$. The perimeter of K, where $P : K \to \mathbb{R}$, can be computed as

$$P(K) = \sum_{i=1}^{\beta-1} d(\alpha_i, \alpha_{(i+1)}) + d(\alpha_\beta, \alpha_1), \tag{4.6}$$

where d is the standard Euclidean distance function $d : \mathbb{R}^2 \times \mathbb{R}^2 \to \mathbb{R}$,

$$d((x_1, x_2), (y_1, y_2)) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}. \tag{4.7}$$

In addition to the perimeter, we can also examine the convex perimeter of the cluster (P_{cvx}). The convex boundary can give us a sense of irregularity in boundary. That is, if the boundary is substantially longer then the convex perimeter, this means that the perimeter may be irregular. To define the convex perimeter, an understanding of convexity is needed.

[5] More detail on this particular metric can be found in [60–62]

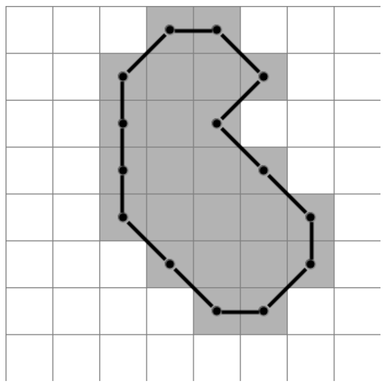


Figure 4.5: Example of polygon perimeter.
An illustration of the boundary of a mask in a matrix. The elements of the mask K are highlighted in gray and the boundary of K is marked with the black dots and connected by the heavy black lines. The perimeter is measured as 17.314 pixels, with 14 elements in the boundary [59].

Definition 7 *A boundary is said to be convex if every straight line segment between two vertices is on the boundary or in the interior.*

The convex hull is the smallest convex boundary in the set of all convex boundaries. Mathematically,

Definition 8 *A boundary is said to be the convex hull (∂_{cvx}) of the mask K if it is the "smallest" (i.e. has the fewest number of elements) convex boundary $\mathcal{P}$ such that $K contained by \mathcal{P}$.*

The convex hull can be found using a variety of methods; such algorithms are outlined in [66–69]. In this paper, we utilized an algorithm similar to the Quickhull method outlined in Barber et. al. [68].

The convex perimeter is calculated in a similar way to the perimeter (Equation 4.6), but instead utilizes the vertices associated with the convex hull, i.e.

$$P_{\text{cvx}} = \sum_{i=1}^{\beta-1} d(\alpha_i, \alpha_{i+1}) + d(\alpha_\beta, \alpha_1), \tag{4.8}$$

where $\alpha_i \in \partial_{\text{cvx}}(K)$ and $i = 1, 2, ..., \beta$.

4.3.3 Diameter

The next geometric property we can examine is diameter. Since these clusters are often irregular in shape, it is difficult to choose a single line that accurately reflects the shape's diameter. In the context of this paper, we will say that the diameter, D, is the distance between the two boundary points which are furthest from each other.

In addition to the diameter, we can also examine the distance between two points on boundary polygon with respect to the centroid. This gives us a sense of how far any one edge is away from the centroid. We will call the longest distance that passes through the centroid D_{max} and the shortest D_{min}. The line which passes between two distinct points on the boundary polygon intersects the centroid if

$$0 = \varsigma(a,b,\varphi) = \begin{cases} \frac{b_2-a_2}{b_1-a_1}(\varphi_1 - a_1) - \varphi_2 + a_2, & \text{if } b_1 \neq a_1 \\ -\varphi_1 + a_1, & \text{if } b_1 = a_1 \text{ and } b_2 \neq a_2 \end{cases}, \tag{4.9}$$

where $a = (a_1, a_2)$, $b = (b_1, b_2)$, $\varphi = (\varphi_1, \varphi_2)$, $a, b \in \partial(K)$, and φ is the centroid. Here ς is the slope-intercept formula. The condition will only be satisfied when φ rests on the line connecting a to b. This equation is used under the assumption that the mask has at least 1 element in the interior. The centroid can be computed on a mask K with m rows and n columns. The centroid is given by the function $\varphi : K \to \mathbb{R} \times \mathbb{R}$ defined as

$$\varphi(K) = \left(\frac{\sum_{i=1}^{m} \sum_{j=1}^{n} j \cdot k_{i,j}}{A_{px}(K)}, \frac{\sum_{i=1}^{m} \sum_{j=1}^{n} i \cdot k_{i,j}}{A_{px}(K)} \right), \text{ where } k_{i,j} \in K. \tag{4.10}$$

Finally, we can define

$$D_{\max}(K) = \max\{d(a,b) : a,b \in \partial(K) \ \text{ and } \varsigma(a,b,\varphi) = 0 : \varphi \text{ is the centroid of } K\} \tag{4.11}$$

and

$$D_{\min}(K) = \min\{d(a,b) : a,b \in \partial(K) \ \text{ and } \varsigma(a,b,\varphi) = 0 : \varphi \text{ is the centroid of } K\}. \tag{4.12}$$

An illustration, taken from [59], of the above detailed geometric properties can be see in Figure 4.6.

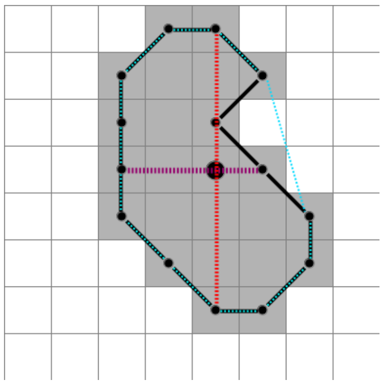


Figure 4.6: Example of geometrical features.
An illustration which demonstrates several of the object's geometrical features. The area of 24 pixels is highlighted by the gray region. $D = D_{\max} = 6$ pixels and $D_{\min} = 3$ pixels are denoted with the red and purple dashed lines, respectively. These both pass through the centroid, marked by the large black dot. The perimeter, measured as 17.314 pixels, is formed by the solid black lines connected by the smaller black dots. Finally the convex perimeter, measured as 15.307 pixels, is illustrated as the dashed teal lines connected by the smaller black dots.

4.3.4 Corners

One last feature that we can measure is the number of corners in a boundary, C. A corner is any element in the boundary whose neighbors are not either both horizontal, or both vertical, in 4-connected clusters. Counting the number of corners in a cluster is useful for understanding how flat the edges of the cluster are.

4.4 Calculating Shape Factors

While the shape properties defined in Section 4.3 can provide us valuable insight into cluster size, they do not necessarily provide information on the overall shape of the clusters. That is, they do not provide information on whether the cluster is spread out, if the boundary is smooth or rough, or if the shape is more elongated or round. To gain data on the shape of clusters, we can combine the shape properties in a variety of equations that yield information relating to the geometric characteristics of a cluster.

The first property, circularity ratio [70], is defined as

$$R_{cir} = \frac{4\pi A_{px}}{P^2}, \tag{4.13}$$

where A is area (Equation 4.2) and P is perimeter (Equation 4.6). A circularity value of close to one suggests that the object is circular, whether values less than one suggest that the object is ellipsoidal in nature. Its formula is derived from the perimeter and area calculations of a circle. The one major flaw with this equation is that it may not be able to distinguish between an elongated object with a smooth edge and a circular object with a rough edge. An object could be approximately circular, but if it has a rough or jagged edge, then even though the object appears circular, R_{cir} will have a value which is much greater than one.

Another property, filament ratio [70], which is defined as

$$R_{fil} = \frac{4A_{px}}{PD}, \tag{4.14}$$

where A_{px} is area (Equation 4.2), P is perimeter (Equation 4.6) and D is diameter (Equation 4.9). Filament ratio quantifies how filamentary (long and thin) an object is. Like R_{cir}, a value of close to one suggests that the object is circular. Values less than one suggest that an object is filamentary or elongated in nature. Unlike the circularity ratio, filament index is not as severely affected by jagged or rough edges.

Aspect ratio [71] is given by the equation

$$R_{asp} = \frac{D_{\min}}{D_{\max}}, \tag{4.15}$$

where $D_{\max}$ and $D_{\min}$ are as defined in equations 4.11 and 4.12, respectively. Aspect ratio is approximately the ratio of the shape's width and height along its primary axes. Since aspect ratio is the ratio between the primary axes, it provides information related to how elongated the object is. In this respect, an ellipse and rectangle with the same bounding region would be indistinguishable from each other. An aspect ratio of 1 would signify that the object is less elongated, and an aspect ratio less than one indicates some form of elongation.

The next shape factor we will look at is the ratio of polygon area to the convex area (R_{cvx}). Ratio of area to convex area is defined as

$$R_{cvx} = \frac{A}{A_{cvx}}, \tag{4.16}$$

were A is polygon area (Equation 4.3) and A_{cvx} is convex area (Equation 4.4). This calculation provides information on how close to convex the cluster is. This is useful for determining how spread out a cluster is and discriminating between an elongated object and an object with a rough edge. If a cluster has filaments or a rough edge then, this ratio will be closer to zero.

The ratio of area to minimal bounding rectangle area is given as

$$R_{mbr} = \frac{A_{px}}{A_{mbr}}, \tag{4.17}$$

where A_{px} is pixels area (Equation 4.2) and A_{mbr} is the area of the minimal bounding rectangle (Equation 4.5). Similar to R_{cvx}, the ratio of area to minimal bounding rectangle area is also useful in computing how spread out an object is. Further, the R_{cvx} is also useful in ascertaining how square an object is. If a cluster is close to square, the ratio will be near 1. If a cluster is more spread out or filamentary, this ratio will be closer to 0.

The next value we will look at is the ratio of the corners to the perimeter,

$$R_{cp} = \frac{C}{P}, \tag{4.18}$$

where C is the number of corners (Section 4.3.4) and P is perimeter (Equation 4.6). The ratio of corners to perimeter gives information on how many twists and turns the perimeter undergoes. The idea behind this metric is that if there is a high ratio of corners to perimeter, then this means that the cluster's boundary is constantly changing direction. If this ratio is low, then this indicates that the cluster has long straight patches along the boundary.

Waviness [72, 73] is defined as

$$W = \frac{P}{P_{\text{cvx}}}, \tag{4.19}$$

where P is the perimeter (Equation 4.6) and P_{cvx} is the convex perimeter (Equation 4.8). Waviness computes how different (e.g. a grooved boundary) the perimeter is from the convex perimeter. Values closer to 1 indicate that the boundary is more regular and values less than 1 indicate a potentially irregular boundary.

Image moments can provide geometric information about the objects' area, center of mass, etc. The p, q moment of an image is

$$m_{p,q}(K) = \sum_{i=1}^{l} \sum_{j=1}^{n} j^p i^q K_{i,j}, \tag{4.20}$$

where K is an l by n matrix and p and q are the order of the moment being computed [74, 75].

Central moments are moments that are normalized around the center of gravity of the image. The p, q central moment is computed as

$$\mu_{p,q}(K) = \sum_{i=1}^{l} \sum_{j=1}^{n} (j - \varphi(K)_y)^p (i - \varphi(K)_x)^q K_{i,j}, \tag{4.21}$$

where K is an l by n matrix, and φ is the centroid (Equation 4.10) [74, 75] .

The second order central moments provide us with information about the object's orientation. If we divide these central moments (Equation 4.21) by area ($m_{0,0}$),[6] then we can find information about the object independent of its size and position, given by the following equation:

$$\bar{\mu}_{p,q} = \frac{\mu_{p,q}}{m_{0,0}}. \tag{4.22}$$

Utilizing the the mathematical moments and normalized moments, we can come up with two additional equations, the normalized compactness,

$$M_{cpt} = \frac{1}{2\pi} \frac{m_{0,0}}{\bar{\mu}_{2,0} + \bar{\mu}_{0,2}}, \tag{4.23}$$

and the normalized eccentricity,

$$M_{ecc} = \frac{\sqrt{(\bar{\mu}_{2,0} - \bar{\mu}_{0,2})^2 + 4\bar{\mu}_{1,1}^2}}{\bar{\mu}_{2,0} + \bar{\mu}_{0,2}}. \tag{4.24}$$

[6] Note that $m_{0,0}$ is equivalent to pixel area, Equation 4.2.

These two equations are normalized versions of their counterparts given in [74]. Similar to R_{fil} and R_{cir}, these two equations give us information on how compact or elongated the cluster is.

4.5 Artifacts from Shape Discretization

It may not be immediately obvious, but we are faced with several challenges related to the fact that these clusters are discrete objects. Because the clusters are discrete, we have no way of knowing the true shape of a cluster, especially for small clusters. Suppose that we have a cluster made up of a single element. There is no way to know if the single element represents a square, or circle, or some other shape, and there is not enough resolution to determine the true shape. Take for example Figure 4.7. It is impossible to distinguish any of the clusters in the final column from each other, as they are all a single pixel. So despite the fact that they were all generated from a different shape, as a single element they are identical.

While it is impossible to know for sure the classification of single elements, there are measures we can take to measure clusters as accurately as possible to help minimize discretization artifacts. One such method is a matter of choosing how to measure the cluster. The measure of a digital shape can be computed using either of the following techniques. In one method, depicted in Figures 4.8a–f, the ideal shape (red) is centered such that the northernmost, southernmost, easternmost, and westernmost points on the shape touch the edges of pixels. This type of alignment will be referred to as an edge-aligned perimeter. In the second approach, illustrated by Figures 4.8g–l, the northernmost, southernmost, easternmost, and westernmost points on the ideal shapes touch the centers of pixels. This type of alignment will be referred to as a center-aligned perimeter. The perimeters for the shapes are then approximated by summing the Euclidean distance (Equation 4.7) between the centers of neighboring boundary pixels, as depicted by the black lines in Figures 4.8a–l. In all cases, the area is computed by

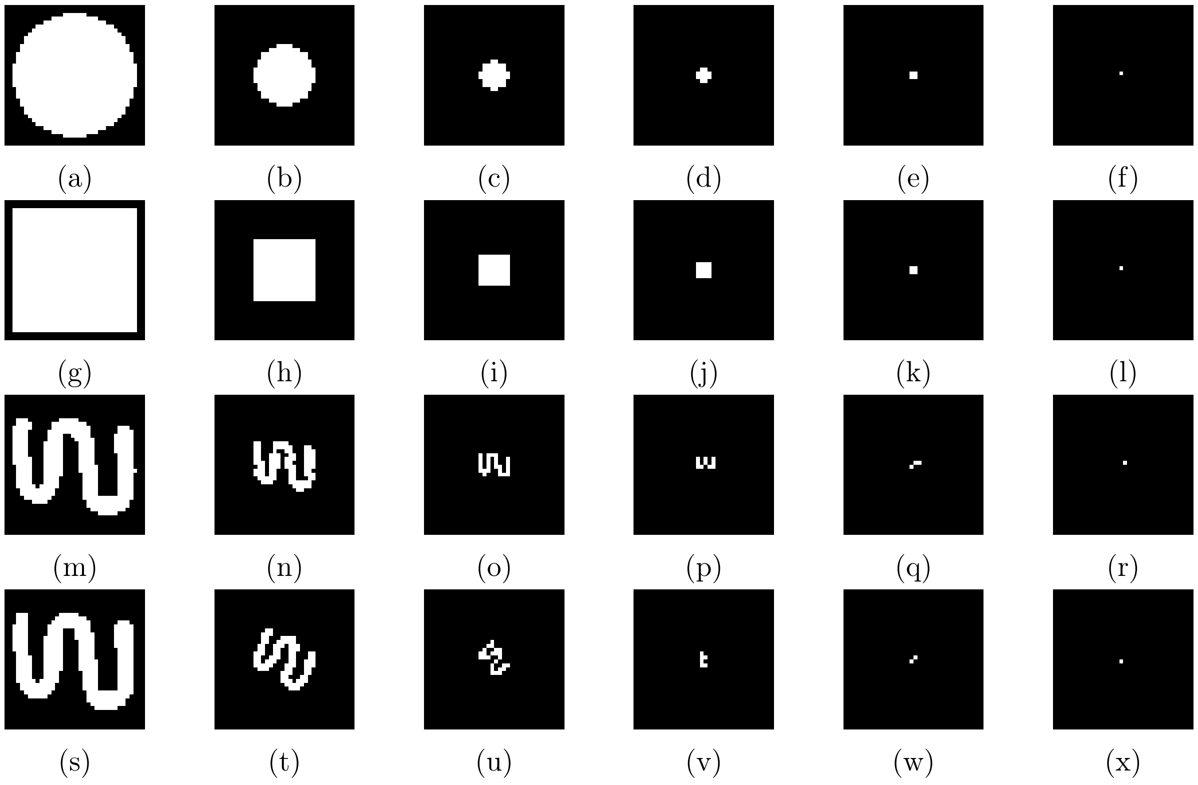


Figure 4.7: Calibration shapes at multiple sizes.
This graphic depicts several objects being scaled down, with the left side being the images at a high resolution and the right being the same images scaled down to a single pixel. These images were generated by scaling down the image on the left by a factor of 2 for each iteration. The bottom row was also rotated in addition to the scaling. One can see that the clusters in the last column are indistinguishable from each other.

calculating the number of pixels inscribed or partially inscribed by the ideal shape.

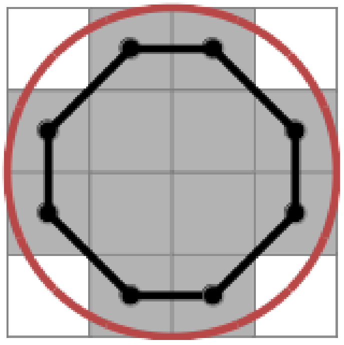
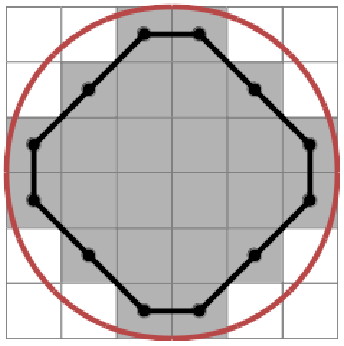
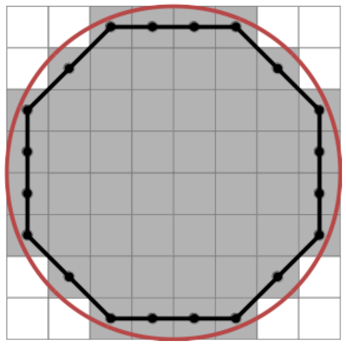
Table 4.1 reports the percent error,

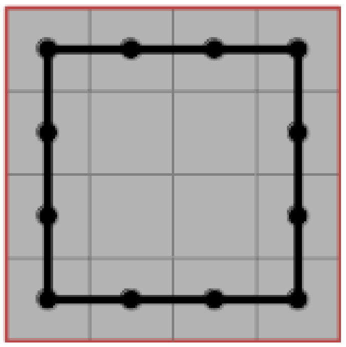
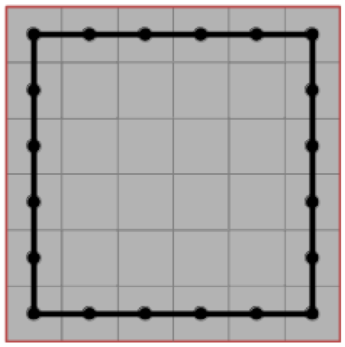
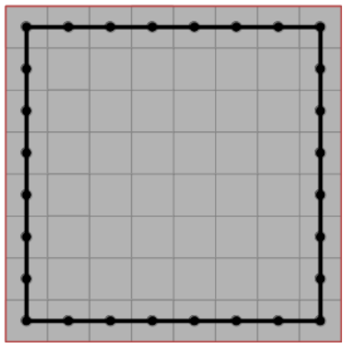
$$E_{\text{pct}} = \left(\frac{\text{Measured} - \text{Expected}}{\text{Expected}}\right)(100\%)\,, \tag{4.25}$$

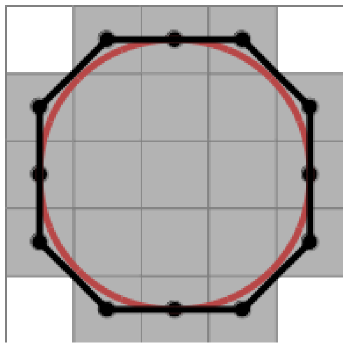
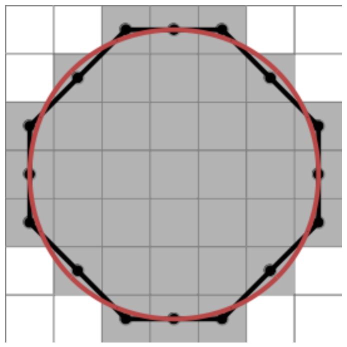
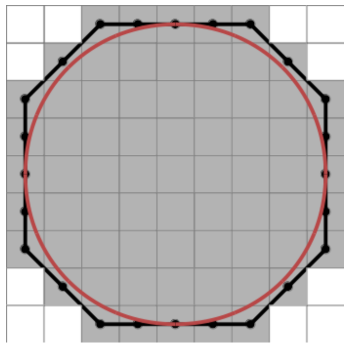
between the approximate and expected perimeter and area computations associated with the respective figures. The edge-aligned perimeter type of measurement shown in Figures 4.8a–f has a high accuracy when measuring the area, but low accuracy for the perimeter. This trend is opposite for the center-aligned type of measurement demonstrated in Figures 4.8g–l, with a better approximation of the perimeter and a worse approximation for the area.

Based on these results, it is expected that there will be some trade-offs in minimizing the errors associated with measuring the perimeter and area of an ideal mathematically modeled object. These figures represent the upper and lower ranges one would expect for the area and perimeter of a circle or square which has been calculated using these methods. It is worth noting that as the number of pixels used to approximate the circle or square increases, the accuracy of both the perimeter and area measurements also increase.

One last measure we took in minimizing the effects of these artifacts was to compute the geometric properties on the topological boundaries, mentioned in Section 4.2, of the clusters. Take for example the 1 element cluster. The area, perimeter, and diameter of this would be 1, 1, and 1, respectively. This means that our $R_{fil} = \frac{(4)(1)}{(1)(1)} = 4$ and $R_{cir} = \frac{(4)(\pi)(1)}{1^2} = 12.57$. However, these values are both greater than 1. The theoretical maximum value for both filament ratio and circularity ratio is 1; anything above this value is undefined. This comes about because we are unable to truly calculate the perimeter and diameter of a single pixel element. However, if we utilize the topological boundary we get an area of 4, a perimeter of 8, and a diameter of 2.83. Now $R_{fil} = \frac{(4)(4)}{(8)(2.83)} = 0.71$ and $R_{cir} = \frac{4\pi 4}{8^2} = 0.79$. One can see that utilizing the topological boundary in addition to the boundary can help to provide additional insights into the shapes of the clusters.


(a)

(b)

(c)


(d)

(e)

(f)


(g)

(h)

(i)

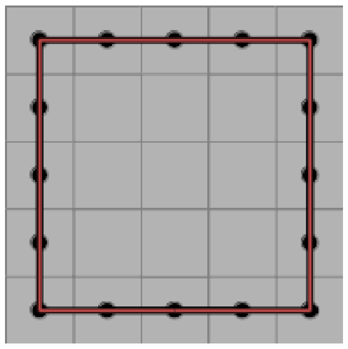
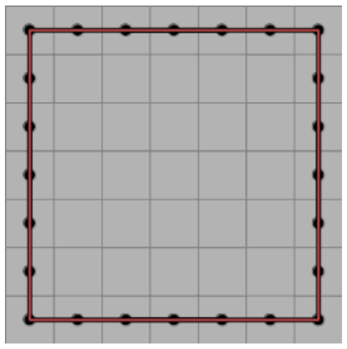
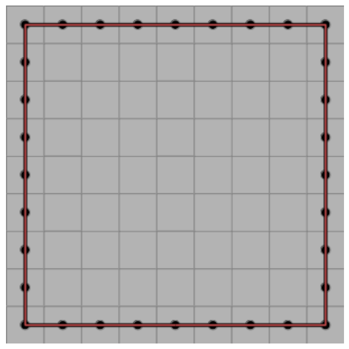

(j)

(k)

(l)

Figure 4.8: Example of shape alignment.
Two sets of circles and squares approximated at three different resolutions using the edge-aligned perimeter and center-aligned perimeter. Edge aligned perimeters for circles with diameter 4 (a), 6 (b), and 8(c) and squares with diameter 4 (d), 6 (e), 8 (f). Centered aligned perimeters for circles with diameter 4 (g), 6 (h), 8 (i) and squares with length 4 (j), 6 (k), and 8 (l). The red line denotes the ideal perimeter and the black line denotes the measured perimeter [59].

Table 4.1: Percent error of the shapes in Figure 4.8

As the diameter increases in the edge-aligned and center-aligned perimeters, the magnitude of percent error decreases in the perimeters and areas for the circles, and either the area or the perimeter for the squares [59].

Shape	Alignment	Diameter/Length	Area	Perimeter
circle	edge	4	-4.51%	-23.13%
circle	edge	6	-15.12%	-18.76%
circle	edge	8	3.45%	-7.24%
square	edge	4	0.00%	-25.00%
square	edge	6	0.00%	-16.67%
square	edge	8	0.00%	-12.50%
circle	center	4	67.11%	8.68%
circle	center	6	30.86%	2.46%
circle	center	8	11.41%	8.62%
square	center	4	56.25%	0.00%
square	center	6	36.11%	0.00%
square	center	8	26.56%	0.00%

4.6 Calibration

A calibration was performed to assess the accuracy and reliability when segmenting objects with known geometric properties. The comparisons were made with a circle (Figure 4.9a), a large circle, an ellipse (Figure 4.9b), and a square (Figure 4.9c). The results were contrasted with their respective ideal mathematical models. This comparison can be seen in Table 4.2.

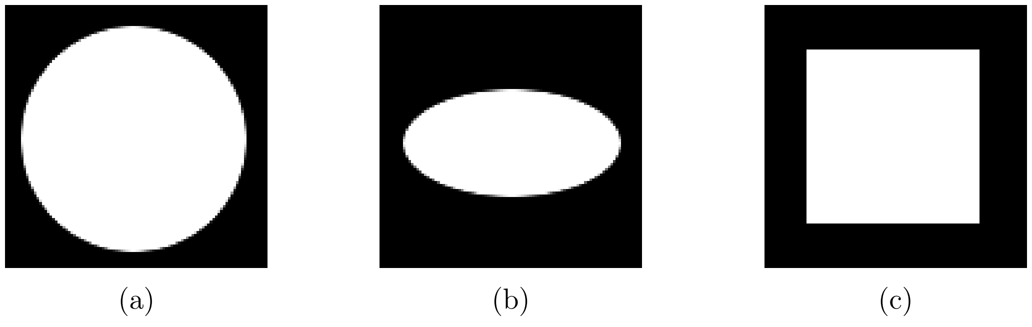

(a) (b) (c)

Figure 4.9: Shapes used in the calibration process.
(a) A circle (diameter of 85). (b) An ellipse with major axis of length 83 and minor axis of length 41. (c) A square with side length of 66 [59].

The data in Table 4.2 was calculated using the percent error equation (Equation 4.25). The measured dimensions and areas align well with the predicted values. The most notable discrepancy in Table 4.2 is in regard to the perimeter. This deviation aligns well with the discretization artifacts noted in Section 4.5.

Table 4.2: Calibration data compared to mathematical models

Calculated percent error of Figures 4.9a, 4.9b, and 4.9c measured against ideal mathematical models [59].

Shape	Area	Diameter	$D_{\max}$	$D_{\min}$	Perimeter
Circle	4.41%	2.14%	2.14%	0.00%	6.26%
Large Circle	0.17%	0.09%	0.09%	-0.10%	5.42%
Ellipse	2.82%	2.87%	-0.85%	0.56%	4.21%
Square	0.02%	-1.58%	-1.58%	-0.07%	-1.52%

CHAPTER 5
ANALYSIS

5.1 Analysis Methods and Variables

For this experiment, we primarily looked at mammograms of two cohorts of patients with biopsy proven, malignant tumors: a set of patients with ductal carcinoma in situ (DCIS) and a set with invasive ductal carcinoma (IDC). The purpose of this analysis was to discover if changes in the roughness of breast tissue or density fluctuations in mammograms could be used to detect signs of the tumors before the changes became visible in the mammograms. To perform this analysis, we utilized the mathematical tools outlined in Chapters 2, 3, and 4.

We divided each mammogram into a grid of squares with a horizontal and vertical step size of 32 pixels. We then constructed subregions for each grid. Each subregion in this grid had a width and height of 360 pixels. Further, this 360×360 pixel square had an interior

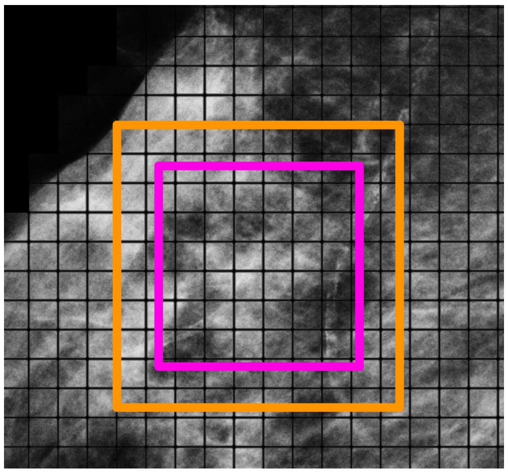


Figure 5.1: A subregion of a mammogram with a grid overlay.
This figure depicts an example of the griding used during the AWTMM analysis pipeline. The black lines are a graphical representation of the 32×32 pixel steps. The orange rectangle is a graphical representation of the 360×360 pixel subregion. The pink square represents the 256×256 region for calculating the final H-value.

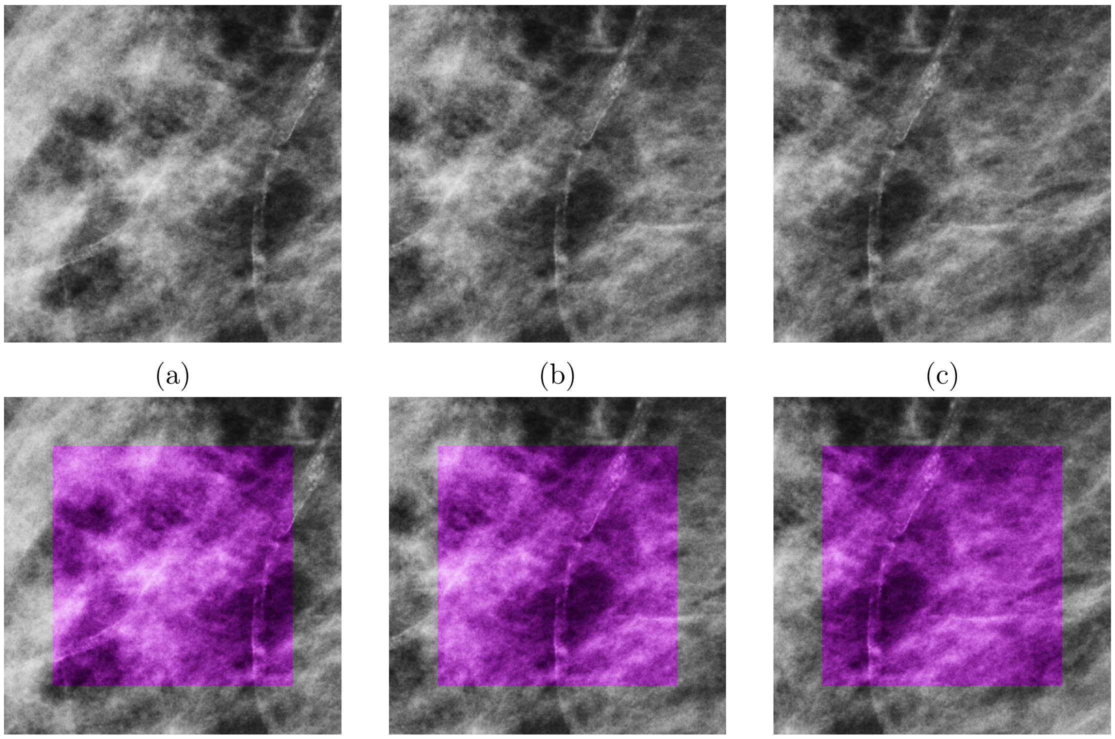


Figure 5.2: Visual representation of the 32 step grid.
This graphic depicts one subregion of the 32 step grid. The images are all 360×360 mammogram subregions. The bottom row depicts the 256×256 region used to compute the H-value. The pixel coordinates of the upper left hand corner of the gray squares 5.2a, 5.2b, and 5.2c are $(1374, 1425)$, $(1406, 1425)$, and $(1438, 1425)$, respectively.

square, placed at its center, with a width and height of 256 pixels. An example of this setup can be seen in Figure 5.1, where the black lines represent grid lines, the orange square represents a single 360×360 subregion, and the pink square represents the inner 256×256 square. Figure 5.2 illustrates how the subregions were generated, namely by shifting the orange square in Figure 5.1 over in a given direction by 32 pixels. Figure 5.2a shows the region outlined by the orange square in Figure 5.1, while Figures 5.2b and 5.2c show the next two subregions to the immediate right of Figure 5.2a. The purple regions in Figures 5.2d-5.2f show the 256×256 square in each respective region.

The WTMM is computed on the outer 360 pixel square, while only the inner 256 region is used to compute the H-value of the region. This is done to prevent misclassification due to mathematical discontinuities at the edges of the 360 region.

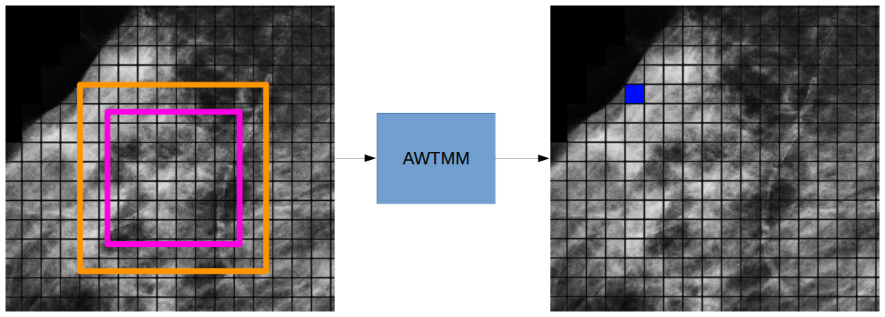


Figure 5.3: Visual representation of the placement of H-value in 32 pixel grid. Here we depict the placement of the H-value for the mathematical mapping onto the matrix corresponding to the (x, y) coordinates of the 32×32 pixel grid.

Next let A be an m by n matrix where m is the number of vertical subregions and n is the number of horizontal subregions. Further, the elements of the matrix A correspond to the (x, y) coordinates of the 32×32 pixel grid. Once the H-value has been computed for a subregion, this value is mapped to the index of the (x, y) coordinates of the 32×32 pixel square at the upper left-hand corner of the 360×360 pixel region. An example of this mapping can be seen in Figure 5.3, where the images serve as visual representations for the mathematical mapping onto A. The process is repeated, with each subregion being mapped to a unique 32×32 pixel square in the mammogram.

Now that we have an understanding of how the images were broken up into overlapping subregions, and how the subregions were classified, we can now look into the procedure for analyzing the H-value matrices generated from the analysis of the mammograms. To analyze the mammograms, the first metric we examined was how the H-values of the mammograms changed over time. We began by labeling each mammogram with a number corresponding to the number of years from the final mammogram in the series in which the tumor was detected. The number of participants in each time step is given in Table 5.1. The total number of mammograms utilized in this study was 328, or 164 for the sides which developed the tumor and another 164 for the opposite breasts. After calculating the H-values for each of the mammograms, we performed a histogram analysis of H-values on each mammogram in our data set. We discovered that patients' tissue composition varied

Table 5.1: The number of patients in each of the time groups.

t (years)	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1	0
N	1	7	9	13	19	18	19	17	15	23	23

widely from year to year; that is, the H-values of mammograms appear to follow no meaningful pattern. There appeared only a loose correlation between the H-value of the mammogram from one year to the next, as can be seen in Figure 5.4. Here Figure 5.4a depicts histograms density for fatty tissue, Figure 5.4b for disrupted tissue, and Figure 5.4c for dense tissue. We expected that the H-values would only fluctuate slightly from year to year and that a trend in the data would be easily identifiable, but this was not the case.

As no discernible pattern was able to be determined, we investigated whether there was any visible pattern detectable in the aggregated data. Since breast tissue density is a risk factor for the development of breast cancer, we decided that examining density would be the next step. In our previous research, we were able to use the combined metric,

$$\frac{\sum(\text{Red Squares})}{\sum(\text{Red Squares}) + 0.8\sum(\text{Blue Squares})}, \tag{5.1}$$

as a way to estimate breast density [16]. Utilizing this metric, we calculated the results of the analysis to generate Figure 5.5a. Breast tissue generally decreases in women as they age [2]; in Figure 5.5a we see an overarching trend where the dense tissue appears to decrease over time. One interesting observation we made about this graph was the dip in median breast density around $t = -2$ years in the side where the tumor was detected. As this trend is not present in the opposite side, we hypothesized that this trend might be an indication of the emergent tumor.

In addition to the density, we examined the relationship between the ratio of dense tissue to that of disrupted tissue. We compute this ratio as

$$\frac{\rho(Y)^{1/16}\rho(R)^{1/2}}{\rho(B) + \rho(Y) + \rho(R)}, \tag{5.2}$$

where $\rho(B)$, $\rho(Y)$, and $\rho(R)$ represent the histogram density of the H-values making up the fatty tissue, disrupted tissue, and dense tissue, respectively. In previous studies (e.g. [16, 39]), we have seen evidence of the relationship between both disrupted and dense tissue and the occurrence of cancer. We hypothesized that there should be some non-linear combination of dense and disrupted tissue which should describe this phenomenon. This equation represents the non-linear combination of the ratios of dense and disrupted tissues to the total breast density. We have no evidence to suggest that this equation will be useful in other applications and may only pertain to this data set. However, it furthers the hypothesis that both dense and disrupted tissues may be an indication of breast cancer.

The final equation we examined related to breast density. This equation was generated from reasoning similar to the rationale that led to Equation 5.2. However, here we now are utilizing the ratio of fatty tissue quarter scaled and multiplied by dampened dense tissue. The equation we came up for this is given as

$$\frac{\rho(B)^{1/4}\rho(Y)^{1/14}\rho(R)^{1/8}}{(\rho(B)+\rho(Y)+\rho(R))^{1/4}}. \tag{5.3}$$

Here we see that the median dampened tissue ratio for the opposite side has a general declining trend, whereas the tumor side appears to be more chaotic and does not decrease in the same way. Much like Equation 5.2, we have no evidence to suggest that this equation will be useful in other applications and again may only pertain to this particular data set. In both equations 5.2 and 5.3, the exponents were obtained through a trial and error/data-mining process.

In addition to breast tissue density, we examined the graphs for the following groups:

- The shape factors for the mean, median, and max clusters for both the disrupted and dense tissues;
- The shape factors for the mean and median cluster sizes for all clusters with area $>$ 40 pixels;
- The densities of histograms with the H-values for the bin size: 1/3, 1/4, ..., 1/20;
- The differences in histogram densities between the tumor side and opposite side for the H-values with the bin size: 1/3, 1/4, ..., 1/20;

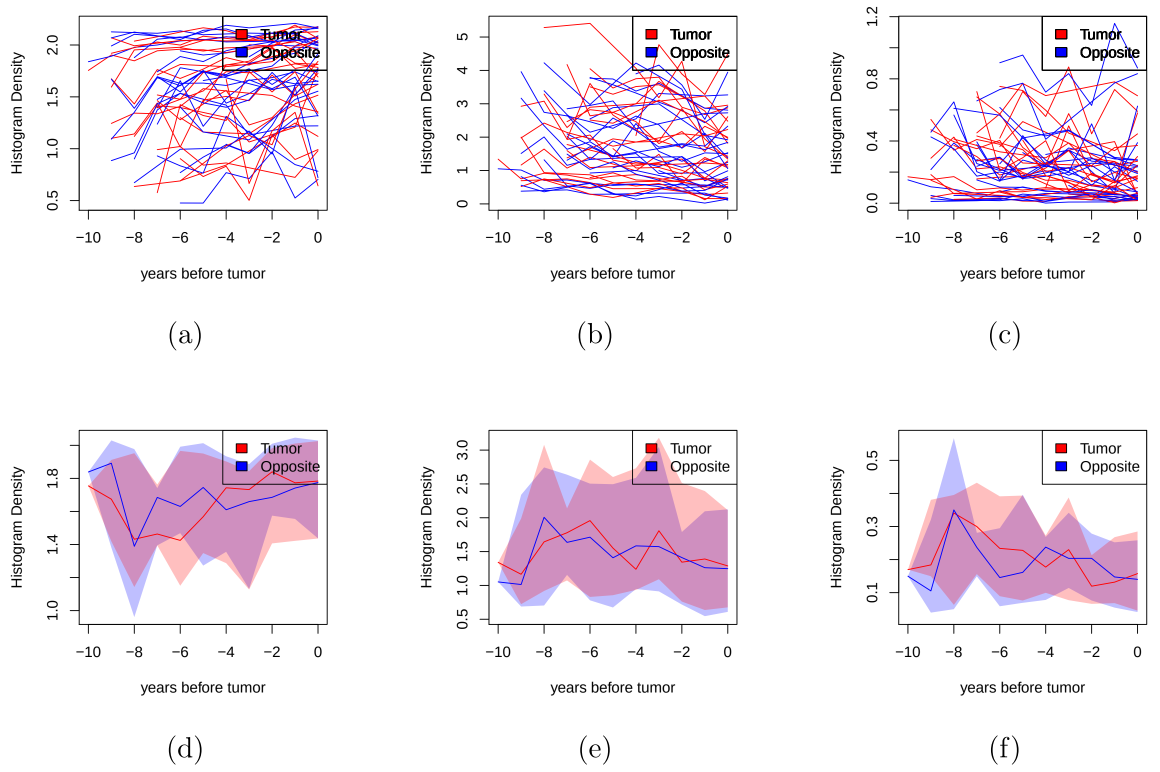
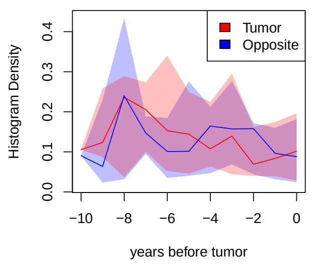


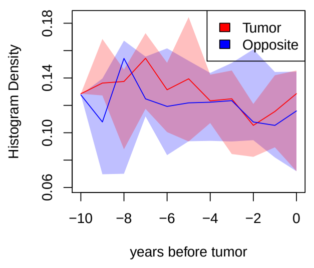
Figure 5.4: Graphs comparing the three tissue types as a function of time. The top row depicts the average density individual histogram densities plotted as functions of time. We generated a 3-bin histogram split at 0.0-0.45, 0.45-0.55, and 0.55-1.0. Graphs 5.4a and 5.4d are the histogram frequencies for the fatty tissue (0.0-0.45). Graphs 5.4b and 5.4e are the histogram frequencies for the disrupted tissue (0.45-0.55). Graphs 5.4c and 5.4f are the histogram frequencies for the dense tissue (0.55-1.0). The bottom row of graphs depicts the aggregated data of the top row. The dark red and blue lines depict the medians of the tumor and opposite sides. The shaded red and blue areas are the first and third quartiles for the aggregated data.

- The number of fatty, disrupted, dense, and no-scaling regions in each mammogram;
- The percent fatty, disrupted, dense, and no-scaling regions in each mammogram;
- The registration between the opposite and tumor sides for each set of mammograms was performed. We compared the differences in H-values and regions between the mammograms.

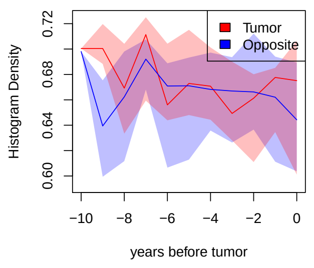
These categories provided us with a total of 1481 variables to examine. However, many of these variables are correlated, and were included as a way to error check the data.



(a)



(b)



(c)

Figure 5.5: A comparison of data generated from Equations 5.1, 5.2, and 5.3. For all of these figures, the red line represents the median of the tumor data, the blue line represents the median of the opposite data, and the shaded red and blue areas represent the first and third quartiles of the cancer and opposite data, respectively. Figure 5.5a depicts density computed with the combined metric, Equation 5.1. Figure 5.5b depicts the comparison of dense and disrupted tissue given by Equation 5.2. Figure 5.5c show the comparison of dense and disrupted tissue given by Equation 5.3.

Since none of the variables we examined seemed to exhibit a reasonable level of coherence from year to year, we treated the data extracted from each mammogram as independent from the rest of the mammograms in that patient's time series. This led us to analyze an aggregated form of the data where we filtered the mammograms into three categories: Normal/Benign[1] (B), Precancer/Pretumor (PC), and Cancer/Tumor (C). We classified the final mammogram in each of the participants' history as the Cancer/Tumor group. We classified the three years preceding the tumor detection as the Precancer/Pretumor group. Finally, the Normal/Benign groups were all remaining mammograms. Table 5.2 provides the number of mammograms in each group. Finally, variables from each category were compared pairwise using the Mann-Whitney-Wilcoxon test. Each variable on the tumor side was compared to the opposite side using the Mann-Whitney-Wilcoxon test as well.

[1]Several of the mammograms had benign abnormalities detected years before malignant masses appeared. To be medically correct we have grouped these under the label Normal/Benign since both types of mammograms had the potential to appear in this group. However, no benign tumors appeared in this group.

Table 5.2: The number of patients in each of the classification groups.

Group	B	PC	C
N	86	55	23

The Mann-Whitney-Wilcoxon test is used to compare two sets of independent observations to determine if they follow the same distribution. It is performed by first sorting observations from smallest to largest, and then assigning a numeric rank (starting from 1) to each observation. In the case of a tie, an average rank is assigned to each observation involved in the tie. The U value is then calculated to determine statistical significance. The U value is given by

$$U = n_1 n_2 \frac{n_1(n_1+1)}{2} - \sum_{i=1}^{n_1} R_i \tag{5.4}$$

where n_1 is the number of observations in the first set, n_2 is the number of observations in the second set, and R_i is the rank of the i^{th} observation in the first set [76]. The U value is then compared against a table of U values [76]. If the lookup value is less than the U value, then the null hypothesis is accepted and the result is statistically significant [76].

This method provided a robust comparison between the groups within variables. Once the p-value was obtained from the Mann-Whitney-Wilcoxon test, we sorted through the list of p-values and found all variables where there was at least one statistically significant group comparison ($p < .05$), paired with at least one other group comparison with a significant, or a nearly significant value ($p < .15$). A summary of the results variables which fulfilled this requirement can be seen in Tables 6.3, 6.4, 6.6, and 6.8.

CHAPTER 6
RESULTS

For all of the tables given in this chapter, we use the following notation listed in Table 6.1 to represent statistical significance. Since we did not discover any variables with $p < 0.0001$, these will be the only categories we will utilize.

Table 6.1: Table of significance levels.

Criteria	Representation
$p \geq 0.05$	
$0.05 > p \geq 0.01$	*
$0.01 > p \geq 0.001$	**
$0.001 > p \geq 0.0001$	***

We examined 1481 variables, so we needed an efficient way to examine all of the variables to locate the variables which were statistically significant. This was done by seeking the variables which had 2 or more statistically significant p-values between groups (B, PC, and C) utilizing the Mann-Whitney-Wilcoxon test. Where applicable, we compared the groups on the tumor side independently of the opposite side. Further, we also compared the tumor side to the opposite side for each of the three groups (B, PC, and C) with the Mann-Whitney-Wilcoxon test. While many of the variables we discovered were correlated, we decided that including all statistically significant variables would yield the clearest picture and would help to inform the results.

In the analysis, we looked at the three different tissue types, fatty ($H < 0.45$), disrupted ($0.45 \leq H \leq 0.55$), and dense ($H > 0.55$). In Chapter 5, we discussed how we developed the masks based on these three categories. For this analysis, we found the clusters (Section 4.2) and associated shape properties (Section 4.4) and computed the shape factors (Section 4.3) for each cluster. In particular, we examined the shape factors and shape properties for the cluster with the largest pixel area in each mammogram (LC), the clusters with a pixel area greater than or equal to 40 (LC40), and the average of all the clusters in a mammogram (AC). Further, we also computed the shape properties and shape factors using the topological boundary (Section 4.2). These results are included with the LC, LC40, and AC results. However, the 'Alt' column in tables Tables 6.3, 6.4, 6.6, and 6.8, denote the shape properties computed using the topological boundary with a 'T'.

Figure 6.1 depicts three hypothetical clusters. The pixel area of the largest cluster (A_{px} of LC) of Figure 6.1 is the orange cluster, with a pixel area of 6. The average pixel area of the clusters (A_{px} of AC) would be ≈ 4.33. One can see that we have no clusters with area greater than 40, so in this case the area of the largest cluster with 40 or more pixels (A_{px} TLC40) would be N/A.

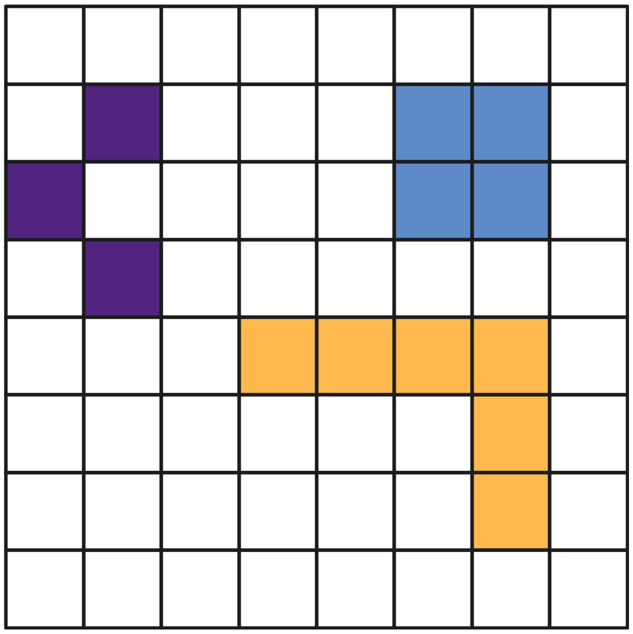


Figure 6.1: Example of Clusters
Here three separate clusters are depicted. The largest cluster is the orange cluster, with an area of 6 pixels.

Figure 6.2 depicts the four different hgroups (Equation 3.25) discussed in Chapter 3.4 which were used in this analysis. We analyzed the clusters in the disrupted tissue ($0.45 \leq H \leq 0.55$, depicted in green in Figure 6.2), the dense tissue ($H > 0.55$, depicted in red in Figure 6.2), and the no-scaling regions (depicted in gray in Figure Figure 6.2). Since the fatty tissue area is often one connected mass, it did not make sense to analyze the shape factors and shape properties of these clusters. We computed both the shape

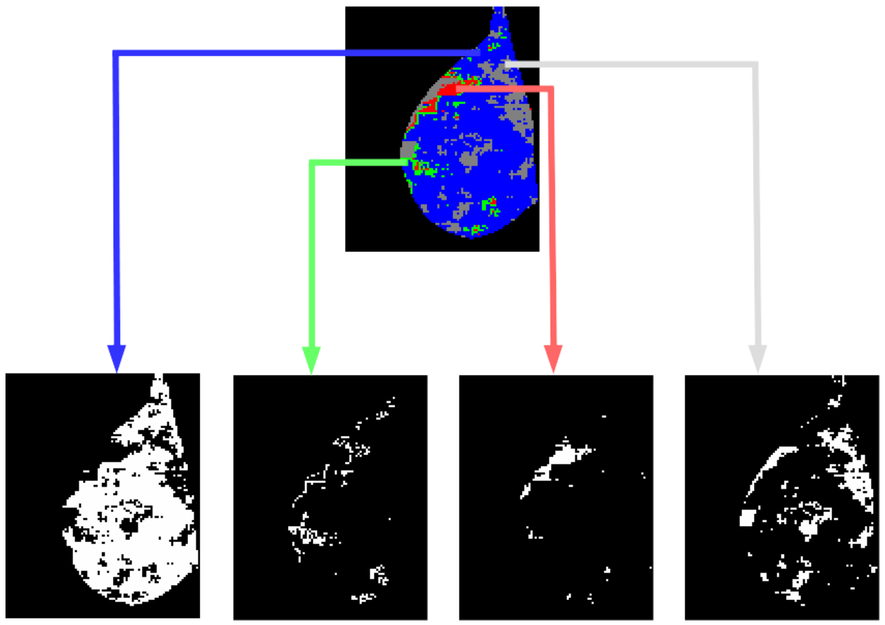


Figure 6.2: Example of AWTMM Mammogram Clusters
Here the output of the AWTMM is being split into four different sets of clusters. The blue represents the fatty tissue ($H < 0.45$), the green represents the disrupted tissue ($0.45 \leq H \leq 0.55$), the red represents the dense tissue ($H > 0.55$), and the gray represents the no-scaling regions.

properties and shape factors, topological and regular, for each individual cluster for each analyzed mammogram. This allowed us to find the LC, LC40, AC, TLC, TLC40, and TAC for each mammogram. The only clusters whose analysis returned significant values were the dense tissue clusters ($H > 0.55$). Sections 6.4 and 6.3 discuss the results of this analysis. For the remainder of the paper the results from LC, LC40, AC, TLC, TLC40, and TAC will be referring to the dense tissue clusters, that is, those clusters with H>0.55.

6.1 Histogram Density

The first set of metrics we examined were the histogram densities (normalized histogram frequency). These histogram densities were obtained by building histograms of the H-values for each of the mammograms at multiple histogram bin sizes. For this analysis, only H-values between 0 and 1 were considered. The bin sizes we examined were were $\{\frac{1}{3}, \frac{1}{4}, \frac{1}{5}, ..., \frac{1}{20}\}$. An example of this can be seen in Figure 6.4b, where the bin size is

$\frac{1}{20}$. In this particular case, the H values of Figure 6.3 were aggregated into a histogram. Here the histogram density of each bin is a variable used in the analysis. Compare this to the histogram of the fatty, disrupted, and dense tissue seen in Figure 6.4a. Note that the three bins corresponding to the designations for fatty, disrupted, and dense tissue, respectively. From this image it is clear that the majority of the tissue considered in this analysis was classified as fatty, while smaller proportions were dense or disrupted.

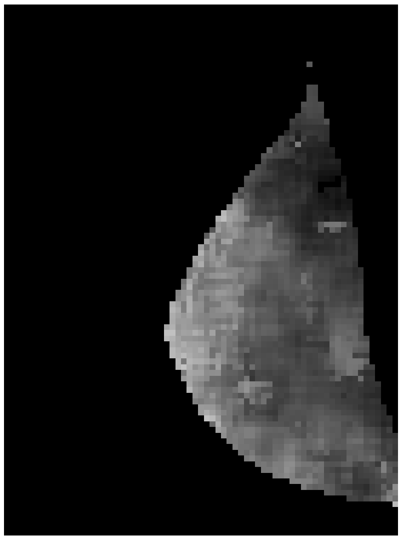
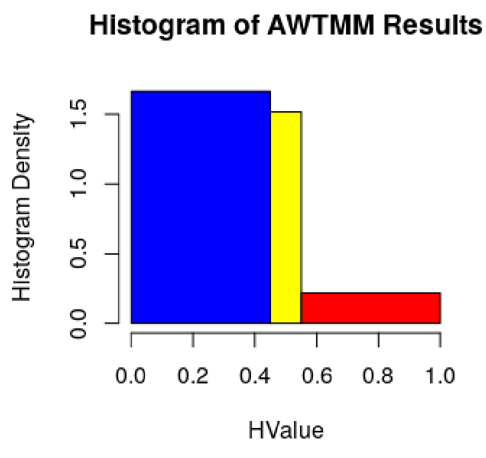
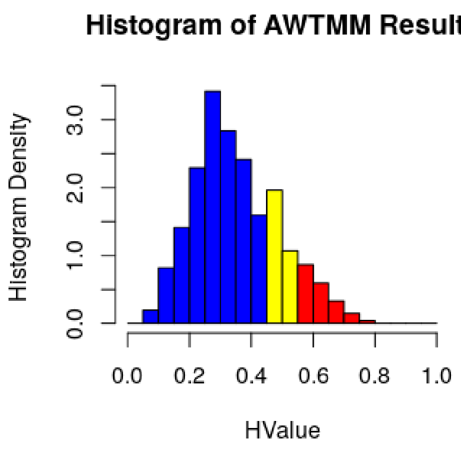


Figure 6.3: A graphical representations of H-values
This image depicts the H-value output of the AWTMM. With exception to the area outside of the breast, the darker values are closer to 0 and the lighters values are closer to 1.

Once all of the bins were calculated, we found the histogram density for each of the bins for each of the mammograms. Each bin was used as a variable for the analysis. Tables 6.2 and 6.3 contain the summary of the results with levels of significance between groups and Figure 6.5 provides a side-by-side comparison of these variables. For this analysis, we ran the Mann-Whitney-Wilcoxon test between all three groups (B, PC, and C) independently for both the tumor (columns T-BvPC, T-BvC, and T-PCvC) and opposite sides (columns O-BvPC, O-BvC, and O-PCvC). Further, we also compared the opposite side to the tumor side (columns BvB, PCvPC, and CvC). From this analysis, it appears that there is a significant difference between the B and PC groups and the B and C groups, but this difference is only observable on the tumor side and is only present in the dense tissue



(a)



(b)

Figure 6.4: Histogram density plots of H-values
This figure depicts the histogram density plots of the H-values from Figure 6.3. The blue regions represent the fatty tissue regions ($H < 0.45$), the yellow regions represent the disrupted tissue regions ($0.45 \leq H \leq 0.55$), and the red regions represent the dense tissue regions ($H > 0.55$).

regions of the mammograms. A majority of the T-BvPC comparisons have p-values < 0.01. This may indicate that there are observable changes in the tissue microenvironment present only on the tumor side. If we look at the graphs associated with these rows, we can see that the density of the opposite side stays roughly constant from year to year while the tumor side has a sharp decrease in density when transitioning from the B group to the PC group. This decrease is then reversed when transitioning from PC to C. Row 10 (0.667-1.0) of Tables 6.2 and 6.3 best summarizes these results, with $p = 0.00249$ for the T-BvPC comparison and $p = 0.03724$ for the T-BvC comparison. We also observed, as shown in Figure 6.5j, that there appears to be some recoil in the tissue microenvironment T-BvC column, as there appears to be less of a difference between the normal and cancer groups on the tumorous side.

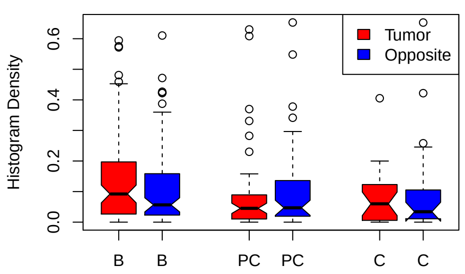
Finally, we also observed in row 5 (0.857-0.929) that there is a difference between the tumor and opposite sides with respect to the PC groups ($p = 0.03607$). Using Figure 6.5f, we can see that there are overall higher levels of this smooth dense tissue (dense tissue with a higher H-value) on the opposite side when compared to the tumor side. While we hypothesize that this is likely an outlier in the data, given the lack of support from other similar metrics, this could indicate that the body is attempting to push back against the developing tumor by increasing the overall amount of dense tissue in the breasts.

Table 6.2: Histogram Density Significance

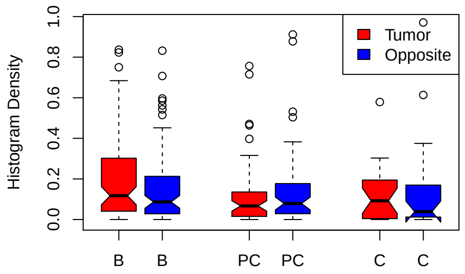
Row	Histogram Density	T-BvPC	T-BvC	T-PCvC	O-BvPC	O-BvC	O-PCvC	BvB	PCvPC	CvC
1	0.7-0.75	**	*							
2	0.692-0.769	**	*							
3	0.846-0.923	*	*							
4	0.8-0.867	**	*							
5	0.857-0.929	*	*						*	
6	0.737-0.789	**	*							
7	0.812-0.875	*	*							
8	0.667-0.833	**	*							
9	0.7-0.8	**	*							
10	0.667-1.0	**	*							

Table 6.3: Histogram Density P-Values

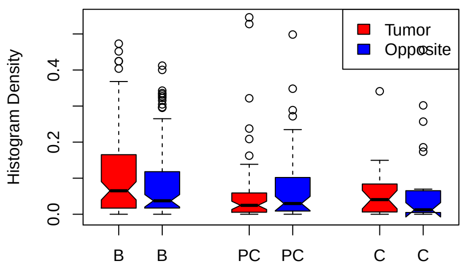
Row	Histogram Density	T.BvPC	T.BvC	T.PCvC	O.BvPC	O.BvC	O.PCvC	BvB	PCvPC	CvC
1	0.7-0.75	0.00516	0.04032	0.92998	0.61347	0.18006	0.28284	0.15934	0.27641	0.82581
2	0.692-0.769	0.002	0.04001	0.83484	0.5611	0.12695	0.31599	0.14533	0.17089	0.86034
3	0.846-0.923	0.01454	0.03899	0.80833	0.9947	0.70357	0.71064	0.56379	0.06512	0.22967
4	0.8-0.867	0.00435	0.04343	0.86973	0.61966	0.50209	0.76989	0.4364	0.09308	0.49877
5	0.857-0.929	0.01078	0.04941	0.98958	0.58265	0.6949	0.50659	0.36062	0.03607	0.36284
6	0.737-0.789	0.00121	0.04516	0.68446	0.46307	0.08531	0.31515	0.19708	0.16391	0.75754
7	0.812-0.875	0.01656	0.04086	0.75313	0.54662	0.34104	0.65957	0.71425	0.1572	0.51742
8	0.667-0.833	0.00192	0.04455	0.86082	0.48289	0.17412	0.35447	0.1677	0.24493	0.98247
9	0.7-0.8	0.00246	0.03861	0.80087	0.43298	0.09544	0.27796	0.18431	0.2485	0.8604
10	0.667-1.0	0.00249	0.03724	0.90405	0.41704	0.14965	0.43333	0.25653	0.23068	0.96495



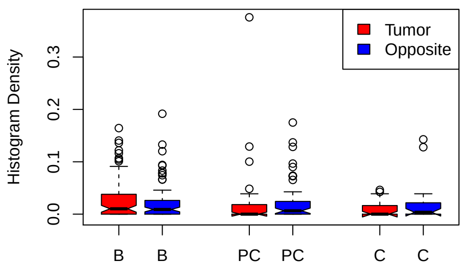
(a) 0.7-0.8



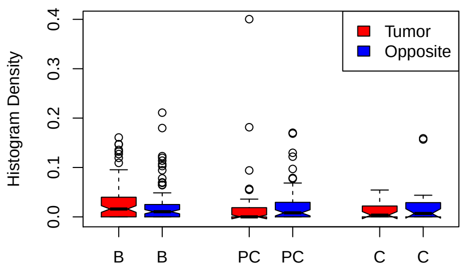
(b) 0.7-0.75



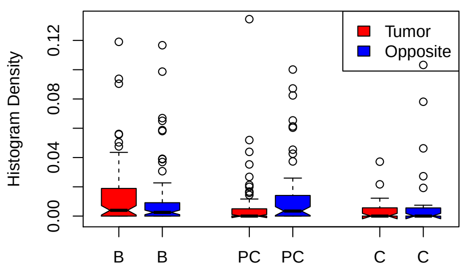
(c) 0.737-0.789



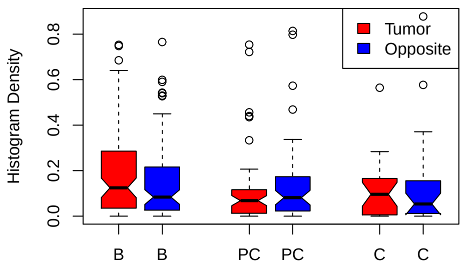
(d) 0.812-0.875



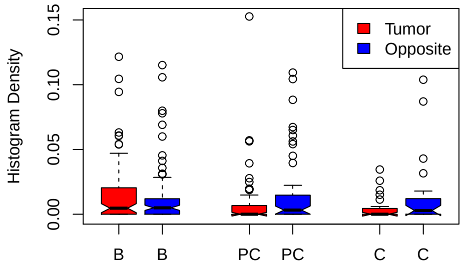
(e) 0.8-0.867



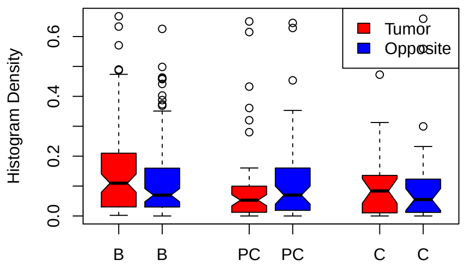
(f) 0.857-0.929,



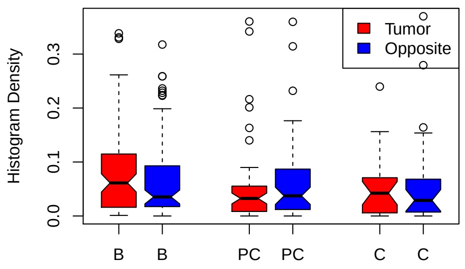
(g) 0.692-0.769



(h) 0.846-0.923



(i) 0.667-0.833



(j) 0.667-1.0

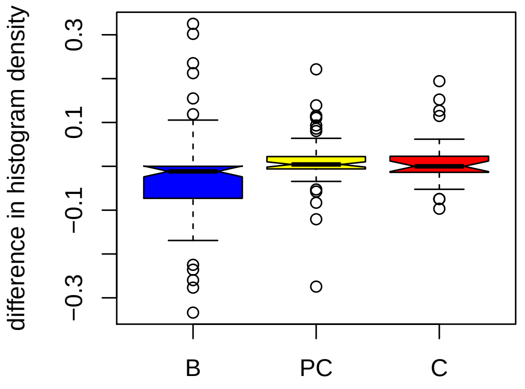
Figure 6.5: Box and whisker charts for histogram densities.
This figure depicts the box and whisker charts of histogram densities for a variety of bins. The bins from 6.5a to 6.5j are: 0.7-0.8, 0.7-0.75, 0.737-0.789, 0.812-0.875, 0.8-0.867, 0.857-0.929, 0.692-0.769, 0.846-0.923, 0.667-0.833, and 0.667-1.0. Histogram bins are H-values associated with dense tissue. This pattern indicates that changes in breast tissue are primarily occurring in dense tissue regions of the mammograms and on the tumor side.

6.2 Difference in Histogram Density

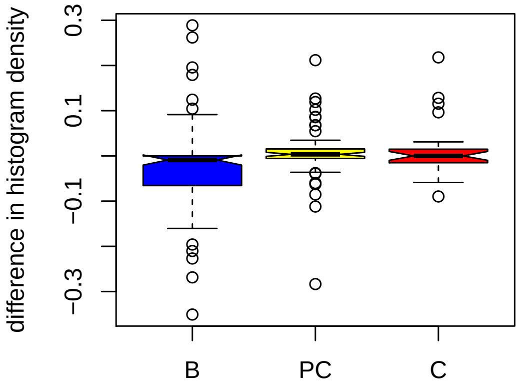
The next set of metrics we examined were the difference between the opposite side and tumor side's H-values in terms of histogram density, which we label as Δ Histogram Density. This data was generated by subtracting the H-value of the opposite side from the tumor side of the histogram data from Section 6.1. Here we found that there appears to be a statistically significant difference between the tumor and opposite side with respect to the dense tissue. In particular, Table 6.4 suggests that there may be a change in the dense tissue composition in the years before cancer is detected by a radiologist. If we examine row 1 (0.737-0.789) of Table 6.4, we can see that the BvPC column has a *p*-value of 0.00031. Further, if we examine the B data in Figure 6.6a, we can see that the median difference between tumor and opposite sides is less than zero, indicating that the tumor side may have more dense tissue than the opposite breast. When looking at PC and C, we can see that these two charts have positive differences, indicating that there is more dense tissue in the opposite breast during these time groups. This provides us with additional evidence that changes are occurring on the tumor side which are not present on the opposite side.

Table 6.4: Significance levels of the difference in histogram density metric.

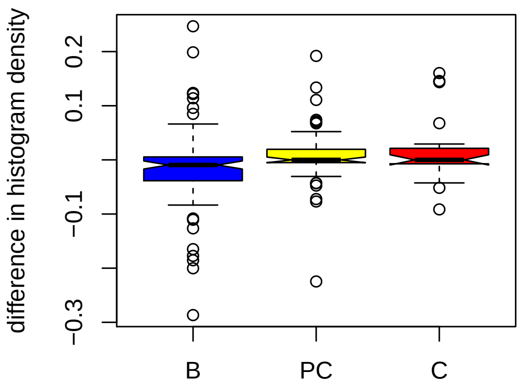
Row	Δ Histogram Density	BvPC	BvC	PCvC	BvPC	BvC	PCvC
1	0.737-0.789	0.00031	0.03915	0.48897	***	*	
2	0.733-0.8	0.00059	0.04481	0.62922	***	*	
3	0.714-0.857	0.00293	0.04609	0.92571	**	*	
4	0.6-0.8	0.00356	0.04615	0.87807	**	*	



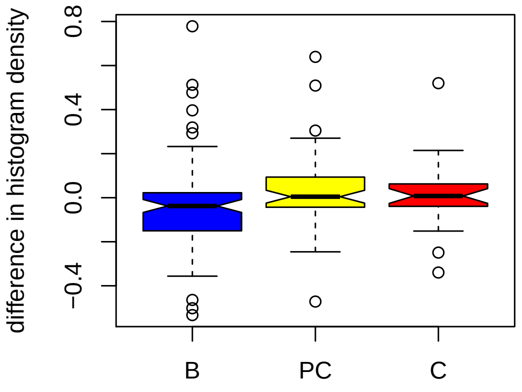
(a) 0.737-0.789



(b) 0.733-0.8



(c) 0.714-0.857



(d) 0.6-0.8

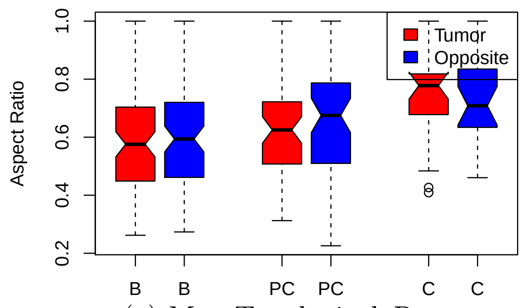
Figure 6.6: Box and whisker charts depicting the difference in histogram density. These graphs depict the difference between the opposite side and the tumor side for the specified histogram bins. Figures 6.6a, 6.6b, 6.6c, and 6.6d depict the difference between densities for histogram bins 0.737-0.789, 0.733-0.8, 0.714-0.857, and 0.6-0.8.

6.3 Shape Factors

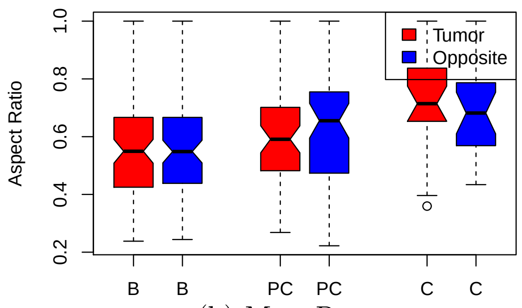
The next set of metrics we examined were the shape factors. Here, Tables 6.5 and 6.6 indicate that there may be some changes in the shape of the dense breast tissue that occur on both the tumor and opposite sides.

6.3.1 Aspect Ratio of LC

We begin by investigating the aspect ratio (Equation 4.15) of LC, rows 1 and 2 of Tables 6.5 and 6.6 and Figures 6.7b and 6.7a. The B and PC groups appear to be less elongated when compared to the C group ($p = 0.00029$ and $p = 0.002$, respectively). On the opposite side, the PC and C groups appear to be more rounded compared to the B group ($p = 0.01545$ and $p = 0.00166$, respectively). This means that there is evidence to suggest that the LC in each mammogram appears to become more rounded, with the clusters on the tumor side rounding out at a faster rate than those on the opposite side.



(a) Max Topological R_{asp}

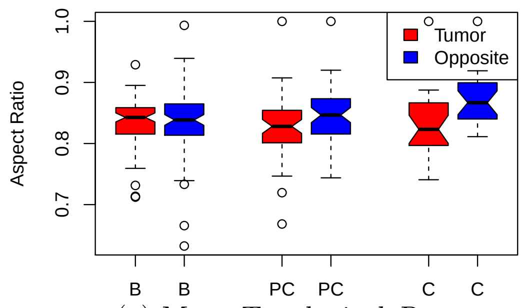


(b) Max R_{asp}

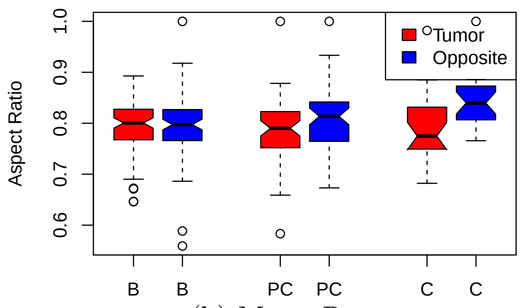
Figure 6.7: Box and whisker charts of aspect ratio R_{asp} for largest dense cluster.

6.3.2 Aspect Ratio of the AC

The next metric we will examine is the aspect ratio (Equation 4.15) of AC, rows 3 and 4 of Tables 6.5 and 6.6 and Figures 6.8a and 6.8b. Here we see signs that the AC on the opposite side in the B and PC groups are less elongated compared to the AC in the C group ($p = 0.00087$ and $p = 0.01432$, respectively). There also appears to be a significant difference between the tumor and opposite side with respect to the PC ($p = 0.04571$) and C groups ($p = 0.00069$). This indicates that on average, the clusters on the opposite side are becoming less elongated, whereas the on the tumor side, the clusters are retaining roughly the same aspect ratio.



(a) Mean Topological R_{asp}



(b) Mean R_{asp}

Figure 6.8: Box and whisker charts of aspect ratio R_{asp} for average dense cluster.

6.3.3 Mean Eccentricity of the LC

Next we will look at the mean eccentricity (Equation 4.24) of the LC, row 5 of Tables 6.5 and 6.6 and Figure 6.9. According to Table 6.5, on the tumor side there is a significant difference between the B and PC groups compared to the C group ($p = 0.00070$ and $p = 0.00382$, respectively). This indicates that there may be a significant decrease in eccentricity when transitioning from the PC to C group, meaning that the LC are becoming more rounded. Further, we can observe that on the opposite side, there is a significant difference between the PC and C groups compared to the B group ($p = 0.02270$ and $p = 0.00166$, respectively). This suggests that the PC and C groups are less eccentric than the B group on the opposite side. The mean eccentricity of the LC behaves much like the aspect ratio of the LC for all three groups, with the LC becoming less eccentric in later stages.

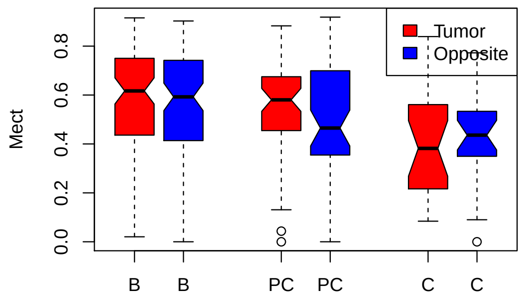


Figure 6.9: Box and whisker charts of mean eccentricity M_{ecc} for largest dense cluster.

6.3.4 Mean Eccentricity of the AC

Next we will examine the mean eccentricity (Equation 4.24) of the AC, row 6 of Tables 6.5 and 6.6 and Figure 6.10. Here we observe a significant difference between the B and PC groups compared to the C group ($p = 0.00104$ and $p = 0.00511$, respectively). Further, we observe a significant difference between the tumor and opposite side with respect to the C group, $p = 0.00127$. Looking at the graphs, we see that there appears to be some small increase in mean eccentricity on the tumor side, whereas on the opposite side this trend is reversed. This could indicate that the AC on the tumor side is becoming less rounded, while on the opposite side the AC is becoming rounder.

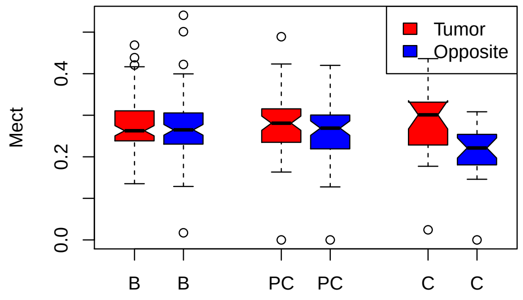


Figure 6.10: Box and whisker charts of mean eccentricity, M_{ecc}, for average dense cluster.

6.3.5 Mean Compactness of the LC

Next we will examine the mean compactness (Equation 4.23) of the LC, row 7 of Tables 6.5 and 6.6 and Figure 6.11. We observed a significant difference on the tumor side between the B and PC groups compared to the C group ($p = 0.00066$ and $p = 0.04611$, respectively). This could indicate that the LC on the tumor side are becoming more compact. Looking at Figure 6.11, we observe that this appears to be the trend. Further, on the opposite side, we observe a significant difference between the B and C groups, $p = 0.02343$. While there is some difference, note that it is not as drastic as on the tumor side.

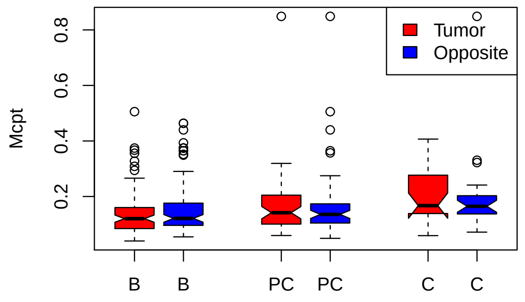


Figure 6.11: Box and whisker charts of mean compactness, M_{cpt}, for largest dense cluster.

6.3.6 Corner to Perimeter Ratio of the LC

Next we examined the ratio of corners to perimeter of the LC (Equation 4.18), row 8 of Tables 6.5 and 6.6 and Figure 6.12. Here we observed a significant difference between the B and PC groups compared to the C group ($p = 0.02439$ and $p = 0.00348$, respectively). This could indicate that the perimeter is becoming longer, that the number of corners is increasing, or that these two properties are changing at the same time in the PC group on the tumor side. Since we have indication that the perimeter is becoming longer, it seems reasonable that much of the change we see here correlates to the elongation of the clusters.

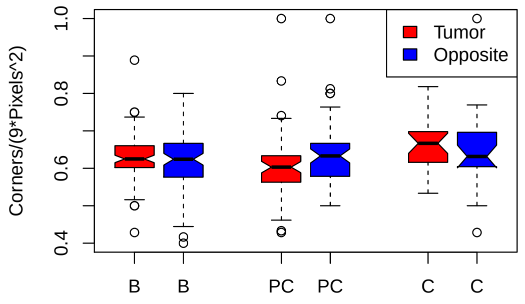


Figure 6.12: Box and whisker charts of corner to perimeter ratio, R_{cp}, for largest dense cluster.

6.3.7 Filament Ratio of the AC

The final shape property we will examine is the filament ratio (Equation 4.14) of the AC, row 9 of Tables 6.5 and 6.6 and Figure 6.13. For this variable, we observe a significant difference on the opposite side between the B and PC groups compared to the C group ($p = 0.04511$ and $p = 0.04981$, respectively). Referring to Figure 6.13, we see that this may indicate that the AC on the opposite side is becoming more circular. Further, there appears to be a non-significant trend on the tumor side in the opposite direction. This was further corroborated when comparing the C groups of the tumor and opposite sides, $p = 0.00420$. This may indicate that in the case of the C group, the AC on the tumor side in the C group may be more more filamentary than the AC on the opposite side.

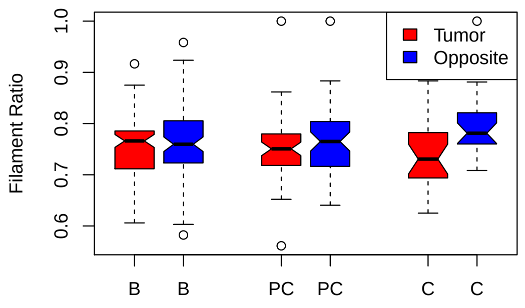


Figure 6.13: Box and whisker charts of filament ratio, R_{fil}, for AC.

6.3.8 Summary of Shape Factors

In summary, we see that on both the tumor and opposite sides that the largest clusters appear to become less eccentric, more compact and rounded overall. However, this reduction in elongation seems to happen only when transitioning from the PC to the C group for the tumor side, whereas the change seems more gradual and less extreme on the opposite side. We observed that the average cluster on the tumor side appears to become more elongated, though not significantly. The average cluster on the opposite side appears to become more circular.

Table 6.5: Significance levels for selected shape factors.

Row	Variable	Alt	CG	T-BvPC	T-BvC	T-PCvC	O-BvPC	O-BvC	O-PCvC	BvB	PCvPC	CvC
1	R_{asp}		Max		***	**	*	**				
2	R_{asp}	T	Max		***	**	*	**				
3	R_{asp}		Mean					***	*		*	***
4	R_{asp}	T	Mean					**	*			**
5	M_{ecc}		Max		***	**	*	**				
6	M_{ecc}		Mean					**	**			**
7	M_{cpt}		Max		***	*		*				
8	R_{cp}	T	Max		*	**						
9	R_{fil}	T	Mean					*	*			**

Table 6.6: P-values for selected shape factors.

Row	Variable	Alt	CG	T.BvPC	T.BvC	T.PCvC	O.BvPC	O.BvC	O.PCvC	BvB	PCvPC	CvC
1	R_{asp}		Max	0.32262	0.00029	0.00298	0.01545	0.00166	0.23657	0.93229	0.13737	0.37947
2	R_{asp}	T	Max	0.29257	0.00030	0.00130	0.02134	0.00158	0.14954	0.91267	0.17762	0.38545
3	R_{asp}		Mean	0.27455	0.21353	0.68515	0.29292	0.00087	0.01432	0.95442	0.04751	0.00069
4	R_{asp}	T	Mean	0.21949	0.25122	0.72996	0.46043	0.00249	0.01347	0.90288	0.07435	0.00174
5	M_{ecc}		Max	0.36236	0.00070	0.00382	0.02270	0.00166	0.27557	0.82879	0.10586	0.51345
6	M_{ecc}		Mean	0.40383	0.31428	0.75065	0.64494	0.00104	0.00511	0.51949	0.10586	0.00127
7	M_{cpt}		Max	0.06881	0.00066	0.04611	0.41878	0.02343	0.06020	0.30505	0.79944	0.52762
8	R_{cp}	T	Max	0.06748	0.02439	0.00348	0.46292	0.23563	0.7258	0.57491	0.10325	0.40964
9	R_{fil}	T	Mean	0.30736	0.18994	0.4763	0.94897	0.04511	0.04918	0.66874	0.1008	0.00420

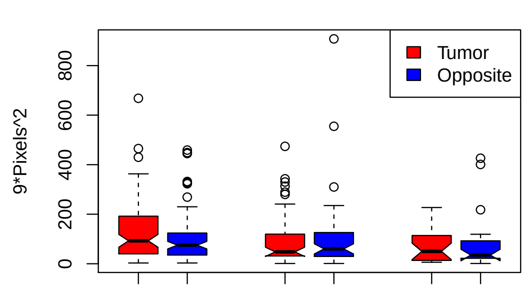
6.4 Shape Dimensions

The next set of properties we examine are the dimensional properties of the clusters in the dense tissue. While the shape factors from Section 6.3 are a combination of many of the dimensions discussed in this section, the dimensions may provide additional insights that the shape factors alone cannot. We can use the information from the shape properties to help understand the change occurring between the different time groups.

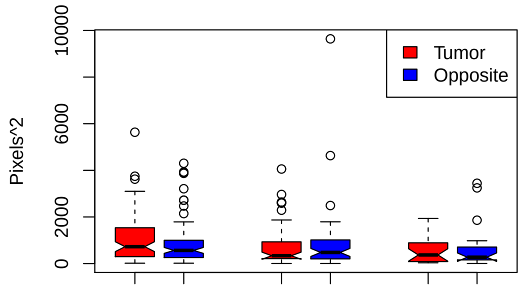
6.4.1 Area of the LC

We start off by looking at the area of the LC. In particular, we will look at rows 1, 3, 4, and 5 of Tables 6.7 and 6.8, and their associated histograms, Figure 6.14. Tables 6.7 and 6.8 suggest that the LC in the B group on the tumor side has a large area that decreases when transitioning to PC and C ($p < 0.05$ for all). This change does not appear to manifest itself on the opposite side. Note that it appears that while the overall dense tissue remains constant over time, the disrupted tissue appears to undergo a non-significant increase during the PC phase. This could indicate that the LC are larger than the LC on the opposite side, though not significantly, and that these clusters become smaller over time.

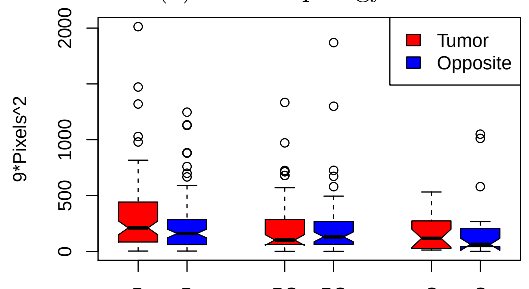
If we refer to Figure 6.15, one may observe that the AC here remain relatively constant. When we looked at the p-values, we observed no significant change in the area of the AC. This means that while the LC is decreasing in size, the AC is not changing. Given the results in Section 6.3, this may indicate that the LC are indeed becoming smaller and more compact over time.



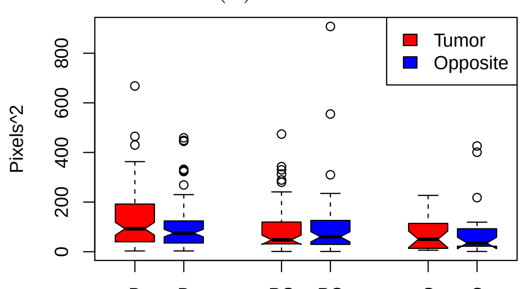
(a) Max Topology A



(b) Max A

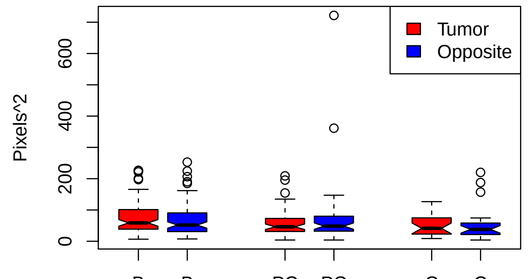


(c) Max Topology A_{mbr}

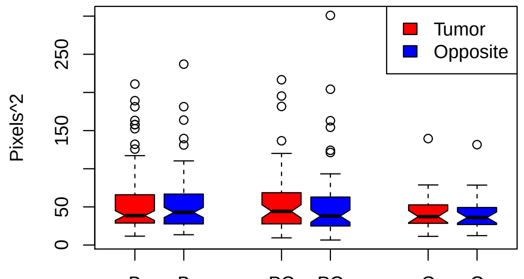


(d) Max A_{px}

Figure 6.14: Topological area shape properties.
This figure depicts the box and whisker charts for the topological area shape properties.



(a) Average dense tissue cluster A.

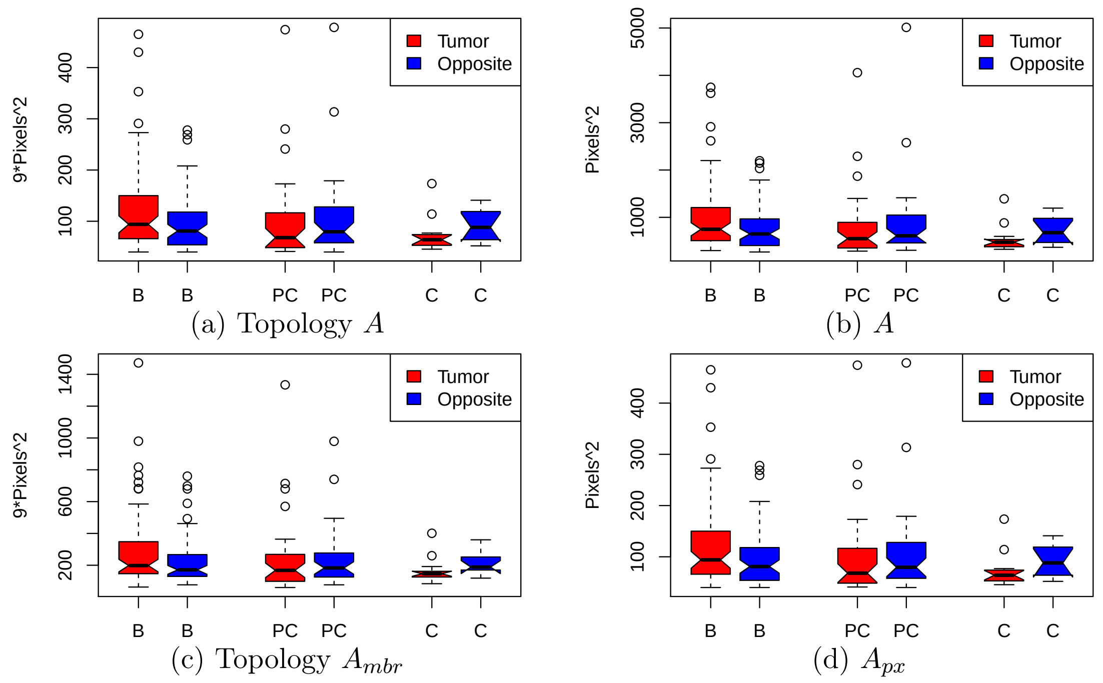


(b) Average disrupted tissue A.

Figure 6.15: Mean area shape properties.
This figure depicts the box and whisker charts for the average area of all clusters.

6.4.2 Area of the Median LC40

Next, we looked at the area of the median LC40. For this data, we will refer to rows 8, 10, 12, and 13 of Tables 6.7 and 6.8 and Figure 6.16. These LC40 show similar behavior to the LC in Section 6.4.1. We observed a significant decrease on the tumor side between the PC and C groups compared to the B group ($p < 0.05$ for both comparisons). This could indicate that these LC40 are remaining roughly the same size on the opposite side, while they are shrinking in size on the tumor side.

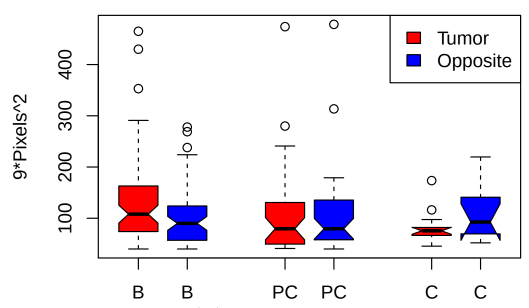


(a) Topology A (b) A (c) Topology A_{mbr} (d) A_{px}

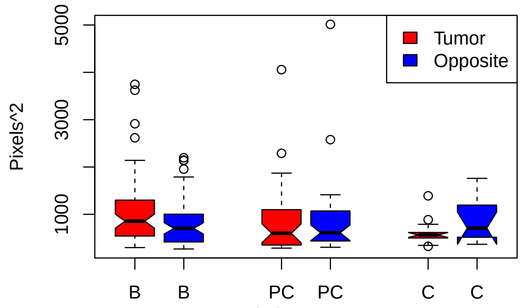
Figure 6.16: Median Areas > 40 pixels.

6.4.3 Area of the Average LC40

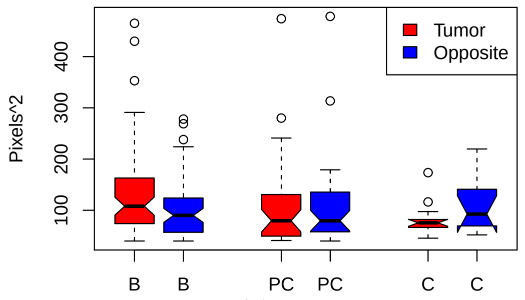
The next set of shape properties relate to the area of the average LC40. This set of variables corresponds to rows 15, 16, and 19 of Tables 6.7 and 6.8 and Figure 6.17c. On the tumor side, the PC and C groups appear to be significantly different from the B group ($p < 0.04$ and $p < 0.02$, respectively). Similar to Sections 6.4.1 and 6.4.2, the LC40 on the tumor side appear to decrease in size over time, while the LC40 remain close to the same size on the opposite side, as can be seen in Figure 6.17c.



(a) Topology A



(b) A

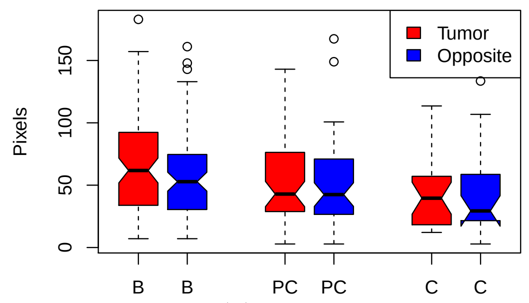


(c) A_{px}

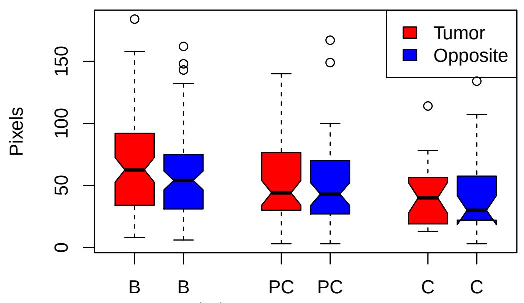
Figure 6.17: Mean Areas > 40 pixels.

6.4.4 Diameter of the LC

The next shape property examined is the diameter of the LC. These variables correspond to rows 6 and 7 of Tables 6.7 and 6.8 and Figure 6.18. We observed a significant difference on the tumor side between the PC and C groups compared to the B group ($p < 0.05$ and $p < 0.01$, respectively). Further, this difference does not appear to be present on the opposite side. Looking at Figure 6.18, it appears that the diameter of the LC is decreasing over time for both the opposite and tumor side. However, this decrease appears more pronounced on the tumor side. This corroborates with the results in Section 6.3, as a reduction in the size of the diameter would be expected for an cluster that is becoming less elongated.



(a) Max D



(b) Max D_{max}

Figure 6.18: Longest significant diameters.

6.4.5 Results of Perimeter Data

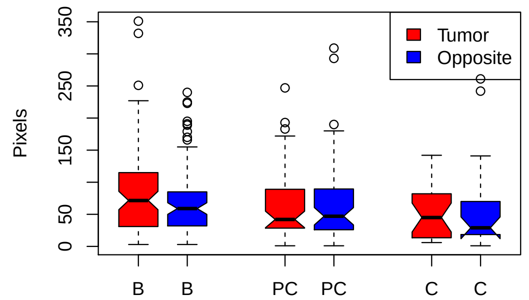
The next dimension we examined was the perimeter, Equation 4.6. Consider rows 2, 9, 11, and 17 from Tables 6.7 and 6.8 and Figure 6.19. The first of these variables we will examine is the perimeter of the LC. The data suggests that the average perimeter of the LC decreases when transitioning from B to PC ($p = 0.04181$) and from B to C ($p = 0.02872$) on the tumor side, as shown in Figure 6.19a.

The next three charts, Figures 6.19b, 6.19c, and 6.19d (rows 9, 11, and 17 of tables 6.7 and 6.8), indicate that the mean and median of LC40 appear to decrease on the tumor side, while either remaining constant or slightly increasing on the opposite side. In particular, we observe a difference between the PC and C groups as compared to the B group (p<0.05 for both comparisons). In the graphs, there does appear to be some minor increase in the length of the perimeter of the LC40. However, if the perimeter on the opposite side is increasing, it is not at a statistically significant level.

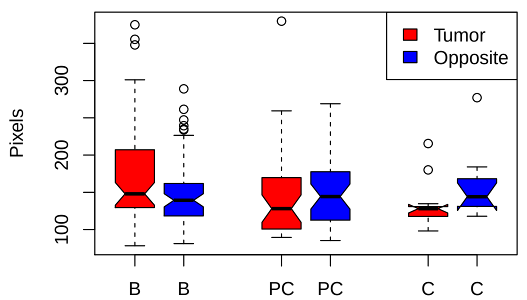
An additional observation we made was that the length of the AC perimeter appears to remain roughly constant over time, as shown in Figure 6.20. This means that while the LC appear to be decreasing in perimeter length on the tumor side, the AC does not reflect this change. This is another indication that the AC on the tumor side may be becoming more compact and less elongated.

Lastly, we observed a significant difference between the tumor and opposite sides with respect to the C group when looking at rows 9 and 11 of Tables 6.7 and 6.8, and Figures

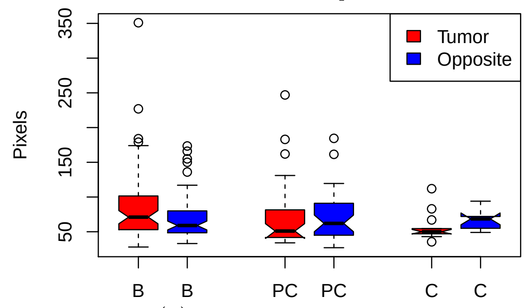
6.19c, and 6.19d. This significant difference, $p < 0.02552$, indicates that the perimeter of the LC40 on the tumor side may be shorter that the perimeter of the LC40 on the opposite side. This is in line with our speculation that the LC40 on the tumor side are decreasing in size and becoming more compact while the LC40 on the opposite side are either maintaining their size or increasing in size marginally.



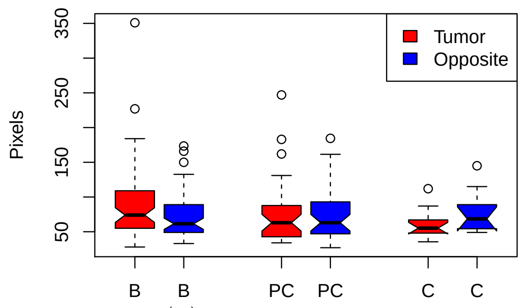
(a) Max P_{px}



(b) Median > 40 P_{cvx}



(c) Median > 40 P_{px}



(d) Mean > 40 P_{px}

Figure 6.19: Selected perimeter shape properties.

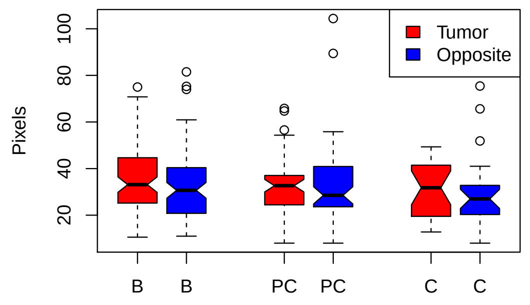
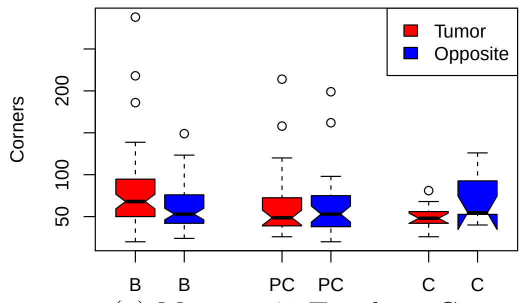


Figure 6.20: The average perimeter size of the dense tissue clusters.

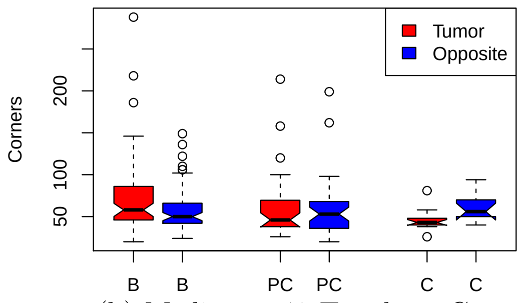
Our data indicated that there was no statistically significant difference between the groups.

6.4.6 Corners of the Average and Median LC40

The final shape properties that we examined were the average and median number of corners of the LC40. The data we will be referring to can be found in Figures 6.21a and 6.21b and rows 14 and 19 of Tables 6.7 and 6.8 and Figure 6.19. Here we observed significant differences on the tumor side between the PC and C groups compared to the B group. In the case of the median number of corners of the LC40, on the tumor side we observed the decrease from the B group to the PC and C groups with significance of $p = 0.03604$ and $p = 0.00963$, respectively. For the average number of corners of the LC40 on the tumor side, we observed a decrease from the B group to the PC and C groups with significance of $p = 0.03067$ and $p = 0.01464$, respectively. While in both of these cases, it appears that the number of corners of the clusters decrease over time on the tumor side, they remain relatively constant on the opposite side. We also examined the average number of corners of the AC, which can be found in Figure 6.22. There were no significant differences in the average number of corners for any of the group pairings. This may indicate that the average number of corners per cluster remains constant while the LC40 on the tumor side appear to have a slight decline. This could be another indicator that the LC40 on the tumor side are becoming less complicated shapes with fewer changes in directions. This decrease in number of corners could also be due to the decreasing area of the dense clusters on the tumor side.



(a) Mean $>$ 40 Topology C



(b) Median $>$ 40 Topology C

Figure 6.21: Mean and median corners of clusters with area $>$ 40 pixels.

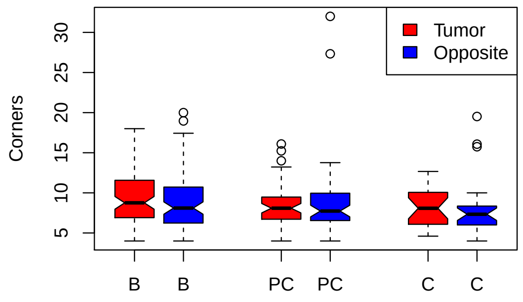


Figure 6.22: The average number of corners of the dense tissue clusters.

6.4.7 Summary of Shape Dimensions

In summary, we see that the average cluster on the tumor side appears to maintain roughly the same area, perimeter, and diameter. Further, the largest clusters on the tumor side appear to decrease in area, perimeter, and diameter. On the opposite side, the neither the large clusters nor average clusters show signs of significant changes.

Table 6.7: Significance levels of selected shape properties.

Row	Variable	Alt	Cluster	T-BvPC	T-BvC	T-PCvC	O-BvPC	O-BvC	O-PCvC	BvB	PCvPC	CvC
1	A_{px}		Max	*	*							
2	P_{px}		Max	*	*							
3	A	T	Max	*	*							
4	A_{mbr}	T	Max	*	*							
5	A		Max	*	*							
6	D		Max	*	**							
7	D_{max}		Max	*	**							
8	A		Median $>$ 40	*	**							
9	P_{cvx}		Median $>$ 40	*	*							*
10	A_{px}		Median $>$ 40	*	**							
11	P_{px}		Median $>$ 40	*	*							*
12	A	T	Median $>$ 40	*	**							
13	A_{mbr}	T	Median $>$ 40	*	*							
14	C	T	Median $>$ 40	*	**							*
15	A		Mean $>$ 40	*	*							
16	A_{px}		Mean $>$ 40	*	*							
17	P_{px}		Mean $>$ 40	*	*							
18	A	T	Mean $>$ 40	*	*							
19	C	T	Mean $>$ 40	*	*							

Table 6.8: P-values of selected shape properties.

Row	Variable	Alt	Cluster	T.BvPC	T.BvC	T.PCvC	O.BvPC	O.BvC	O.PCvC	BvB	PCvPC	CvC
1	A_{px}		Max	0.03384	0.02238	0.4237	0.71045	0.15712	0.20362	0.08443	0.64741	0.91251
2	P_{px}		Max	0.04181	0.02872	0.44952	0.64182	0.22688	0.31328	0.10882	0.80172	0.80899
3	A	T	Max	0.03384	0.02238	0.4237	0.71045	0.15712	0.20362	0.08443	0.64741	0.91251
4	A_{mbr}	T	Max	0.04789	0.03243	0.48304	0.59082	0.14541	0.25897	0.11654	0.77184	0.88641
5	A		Max	0.03077	0.02261	0.42371	0.70571	0.14131	0.20561	0.08987	0.57414	0.96495
6	D		Max	0.04719	0.0056	0.17763	0.53471	0.11247	0.23443	0.10647	0.97854	0.80899
7	D_{max}		Max	0.04601	0.00419	0.15738	0.49071	0.09642	0.21349	0.11192	0.97377	0.81748
8	A		Median $>$ 40	0.014	0.00518	0.65093	0.66014	0.72849	0.92748	0.06606	0.14143	0.11506
9	P_{cvx}		Median $>$ 40	0.04084	0.0164	1	1	0.37652	0.49685	0.08138	0.42606	0.02552
10	A_{px}		Median $>$ 40	0.01562	0.00747	0.78063	0.65481	0.67938	0.85559	0.06604	0.16575	0.10678
11	P_{px}		Median $>$ 40	0.02801	0.017	0.97223	0.94449	0.34178	0.45088	0.07595	0.33807	0.02172
12	A	T	Median $>$ 40	0.01562	0.00747	0.78063	0.65481	0.67938	0.85559	0.06604	0.16575	0.10678
13	A_{mbr}	T	Median $>$ 40	0.04607	0.02386	0.8165	0.97661	0.46193	0.58508	0.12893	0.29207	0.06726
14	C	T	Median $>$ 40	0.03604	0.00963	0.5536	0.895	0.25716	0.34908	0.06561	0.58831	0.01298
15	A		Mean $>$ 40	0.03484	0.01551	0.70184	0.94741	0.76606	0.6031	0.0801	0.46738	0.18621
16	A_{px}		Mean $>$ 40	0.03873	0.01929	0.67619	0.93283	0.70994	0.5499	0.07163	0.5207	0.20804
17	P_{px}		Mean $>$ 40	0.04932	0.03247	0.84363	0.7778	0.42266	0.3232	0.06973	0.6767	0.0824
18	A	T	Mean $>$ 40	0.03873	0.01929	0.67619	0.93283	0.70994	0.5499	0.07163	0.5207	0.20804
19	C	T	Mean $>$ 40	0.03067	0.01464	0.68445	0.59255	0.28991	0.25799	0.06284	0.79109	0.06705

CHAPTER 7
CONCLUSION AND DISCUSSION

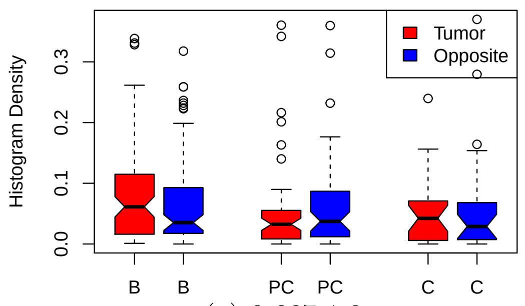
7.1 Summary of Results

7.1.1 Histogram Density Results

For the histogram results, discussed in Section 6.1, only the histogram bins where $H > 0.55$ (R) showed significance. We observed that the bin $0.667 < H < 1$ encompassed the range of all the other significant bins presented in Table 6.3. This overlap of the other density variables (Table 6.3) makes it a good candidate to summarize the results.

On the tumor side, for Normal/Benign and Precancer/Pretumor, $p = 0.00249$, and for Normal/Benign and Cancer/Tumor, $p = 0.03724$, meaning that we observed a reduction in the amount of tissue classified with $0.667 < H < 1$ when transitioning from both Normal/Benign to Precancer/Pretumor and Normal/Benign to Cancer/Tumor. On the opposite side, we observed a roughly constant density level of $0.667 < H < 1$ between all three groups. However, the opposite side does appear to have a slight, non-significant reduction in the $0.667 < H < 1$ tissue. This is expected, as breast density tends to decrease with age.

The quantity of "very dense" (i.e., $0.667 < H < 1$) mammographic tissue is a clinically meaningful metric. The very dense mammographic tissue varies significantly as a function of time in the tumorous breast in the years prior and leading up to the tumor diagnostic. Meanwhile no such significant variation of very dense mammographic tissue is detected in the opposite, healthy breast. This lends credence that this very dense mammographic metric should be considered as a candidate variable for future attempts at the implementation of an automated cancer pre-detection tool.



(a) 0.667-1.0

Figure 7.1: Box and whisker chart for the histogram density of the "very dense" ($0.667 < H < 1$) mammograpic tissue regions.

7.1.2 Elongation of Largest Cluster of Long-ranged Correlated Mammogram Regions

The largest cluster of long-ranged correlated tissue can be best summarized using the aspect ratio metric (4.15). Recall from Chapter 6 that we examined the clusters formed from monofractal anti-correlated ($H < 0.45$, labeled as fatty tissue or B), monofractal long-range correlated ($H > 0.55$, labeled as dense tissue or R) and monofractal uncorrelated ($0.45 \leq H \leq 0.55$, labeled as disrupted tissue or Y). Only the clusters formed from the monofractal long-range correlated data showed significance when comparing two or more pairs of variables as outlined in Section 4.6. On the tumor side, we observed a significant difference between Benign/Normal and Cancer/Tumor groups ($p = 0.00029$) and Precancer/Pretumor and Cancer/Tumor groups ($p = 0.00298$). Further, we observed significant differences on the opposite side between the Benign/Normal and Precancer/Pretumor groups ($p = 0.01545$) and the Benign/Normal and Cancer/Tumor groups ($p = 0.00166$). In all of the metrics, it appeared that the largest cluster of long-ranged correlated tissue became less elongated and more compact. This provides us with evidence that the largest cluster of long-ranged correlated tissue in each mammogram appears to become more rounded, with the clusters on the tumor side rounding out at a faster rate than those on the opposite side.

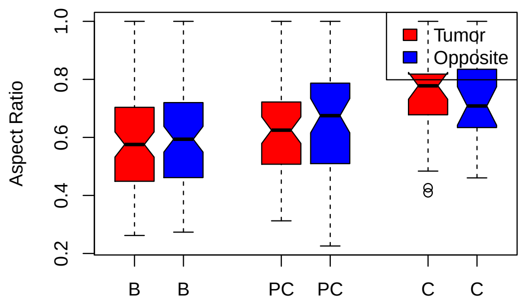


Figure 7.2: Box and whisker chart of the aspect ratio of the largest cluster of long-ranged correlated tissue

7.1.3 Elongation of the Average Cluster of Long-ranged Correlated Mammogram Regions

The mean eccentricity (Equation 4.24) of the average cluster of long-range correlated regions serves as a good summary variable for the elongation of average cluster of long-range correlated regions. We observed on the opposite side a significant difference between the Benign/Normal and Cancer/Tumor groups ($p = 0.00104$) and between the Precancer/Pretumor and Cancer/Tumor groups ($p = 0.00511$). Further, we observed a significant difference between the tumor and opposite sides in the Cancer/Tumor group ($p = 0.00127$). Looking at Figure 7.3, we see that there appears to be some small increase in mean eccentricity on the tumor side, whereas on the opposite side this trend is reversed. This could indicate that the average cluster of long-ranged correlated mammographic tissue regions on the tumor side is becoming less rounded, while on the opposite side the average cluster of long-ranged correlated mammographic tissue regions are becoming rounder.

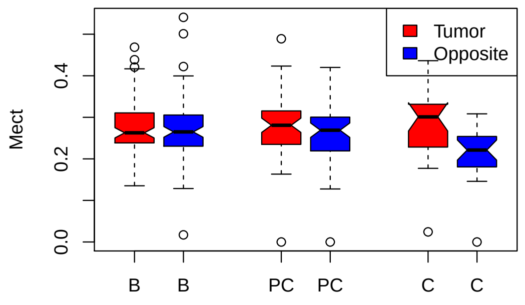


Figure 7.3: Box and whisker chart of mean eccentricity, M_{ecc}, for average cluster of long-ranged correlated mammographic tissue regions.

7.1.4 Shape Properties of Clusters of Long-ranged Correlated Mammogram Regions

We examined the shape properties of the average clusters of long-ranged correlated mammographic tissue regions, largest clusters of long-ranged correlated mammographic tissue regions, clusters of area greater than 40 pixels of clusters of long-ranged correlated mammographic tissue regions. These shape properties include the area (Equation 4.2), perimeter (Equation 4.6), and diameter (Section 4.3.3). These shape properties (Section 4.3) are used to compute the shape factors (6.3) and are thus highly correlated with the shape factors. Upon analyzing them, we discovered that they changed in ways which were in line with the shape factors. Since the shape factors are calculated from the shape properties, this makes sense. There were no surprising results from the shape properties. The large clusters appeared to decrease in size on both the tumor and opposite side, though not at the same rate. On average, clusters remained around the same on both the tumor and opposite side, although again they changed at different rates.

7.2 Conclusion

There is evidence to suggest that the AWTMM algorithm and associated mammogram analysis software are able to detect signs of breast cancer in timeline data before an official diagnosis from a radiologist. Given that breast cancer is notoriously difficult to detect in dense tissue, we find these results to be encouraging.

In Chapter 6, we outlined the changes in the breast microenvironment determined from analyzing the output of the AWTMM. We observed significant differences between the tumor and the opposite groups, and also changes between the B, PC, and C time groups in terms of the dense tissue regions. The evidence suggests that the makeup of the dense tissue on the tumor side undergoes more extreme changes when compared to the opposite side and that these changes occur independently of the opposite side, years in advance of an official cancer diagnosis. The largest cluster on the tumor side becomes even more compact than the largest cluster on the opposite side and undergoes a rapid transition between the precancer and cancer time groups. Further, the evidence suggests that the largest clusters on the tumor side are on average larger than the largest clusters on the opposite side during the B time group, suggesting that in some cases, cancer may be growing for many years in a breast before it is visible to a radiologist.

In addition to the largest clusters, the average cluster with pixel area greater than 40 pixels appears to decrease on the tumor side while average clusters on the opposite side maintain roughly the same area between the B, PC, and C time groups. However, the overall average area on both the tumor and opposite sides remains relatively constant over time. Further, the average dense cluster on the tumor side appears to have some propensity toward becoming more filamentary, while on the opposite side the dense tissue clusters appear to become more rounded. This could indicate that the tumor side is developing more small dense clusters over time. We hypothesize that these may be smaller dense tissue filaments, present due to the spreading tumor.

7.3 Future Work

In addition to the variables we examined, we also attempted to see if it would be possible to classify a mammogram into one of the three time groups (B, PC, and C) based on the set of parameters. We initially used linear and quadratic discriminate analysis (LDA and QDA, respectively) to attempt the classification. However, because of the assumptions of the LDA and QDA, we were only able to use 22 variables. One of the primary assumptions of the LDA and QDA is that one can have at most $n - 1$ variables, where n is the size of the smallest group. Upon running both the LDA and QDA, the mammograms were able to be successfully classified at better than random chance, even when cross-validating (jackknifing) the results. However, because of the small sample size, this analysis served as a preliminary result at best and was not included in this thesis.

We did not explore other machine learning techniques for the classification at the time of the publication. One of the primary reasons is the number of data points. There are not enough data points to build both a training and a validation set. While jackknifing a neural network is possible, it seemed more prudent to focus instead on the statistical analysis of the data instead of attempting to build models with insufficient data.

Two complicating factors for this analysis were the number of mammograms we analyzed and the assumption of independence for the different time groups. A similar analysis with a larger population would likely produce stronger results. However, given the consistency of the results, we do not feel this point invalidates the study. Further, given our understanding of breast cancer and its development, the results appear consistent with that model. As for the latter concern, we feel that there is a high enough level of independence from year to year for this assumption to be warranted. Given enough patient data, we could confirm this assumption by using a single mammogram from each participant to ensure independence.

This research lays the groundwork to analyze additional data utilizing the AWTMM. One area that is currently open for exploration is analyzing these images with finer gridding instead of our current 32-pixel step size. Further improvements to the code, such as parallelization of the WTMM chaining procedure, would allow us to improve the throughput of our mammogram analysis software. With a finer resolution, we may be able to more accurately detect how the dense tissue and its shape change over time. A larger data set including mammograms from patients with biopsy-proven benign tumors would enable us to build a more encompassing model and improve the accuracy of our results. More participants would also allow us to control for factors such as age, family history, and cancer type.

As part of the future work we will examine a larger patient population. This population will include a normal group, benign group, and a cancer group. These populations will be made up of patients who never developed a tumor over their mammography history;

patients who developed a benign tumor over the course of their mammography history; and finally a group of patients who developed a cancerous tumor over the course of the mammography history. We hope that comparing the microenvironment of the breast tissue from these groups will provide us with a more robust dataset, with the potential of classifying pre-malignant tissue.

While further research is needed to build a model which can pre-detect breast cancer, the fact that we were able to see changes in both the presence of dense tissue and the shape of the dense tissue indicates that a pre-detection model is not outside the realm of possibility. We hope these results will shed light on the spread and development of tumorous tissue in mammograms.

REFERENCES

[1] Siegel R. L., Miller K. D. and Jemal A, “Cancer statistics, 2017,” *CA: A Cancer Journal for Clinicians* **67** no. 1, (Jan., 2017) 7–30. https://doi.org/10.3322/caac.21387.

[2] *Breast Cancer Facts & Figures 2017-2018*. American Cancer Society, 2017. Alanta.

[3] Bertrand K. A., Scott C. G., Tamimi R. M., Jensen M. R., Pankratz V. S., Norman A. D., Visscher D. W., Couch F. J., Shepherd J., Chen Y.-Y., Fan B., Wu F.-F., Ma L., Beck A. H., Cummings S. R., Kerlikowske K. and Vachon C. M, “Dense and nondense mammographic area and risk of breast cancer by age and tumor characteristics,” *Cancer Epidemiology Biomarkers & Prevention* **24** no. 5, (Feb., 2015) 798–809. https://doi.org/10.1158/1055-9965.epi-14-1136.

[4] Boyd N. F., Guo H., Martin L. J., Sun L., Stone J., Fishell E., Jong R. A., Hislop G., Chiarelli A., Minkin S. and Yaffe M. J, “Mammographic density and the risk and detection of breast cancer,” *New England Journal of Medicine* **356** no. 3, (Jan., 2007) 227–236. https://doi.org/10.1056/nejmoa062790.

[5] Autier P., Boniol M., Middleton R., Doré J.-F., Héry C., Zheng T. and Gavin A, “Advanced breast cancer incidence following population-based mammographic screening,” *Annals of Oncology* **22** no. 8, (Jan, 2011) 1726–1735. https://doi.org/10.1093/annonc/mdq633.

[6] Erbas B., Provenzano E., Armes J. and Gertig D, “The natural history of ductal carcinoma in situ of the breast: a review,” *Breast Cancer Research and Treatment* **97** no. 2, (Dec., 2005) 135–144. https://doi.org/10.1007/s10549-005-9101-z.

[7] Allred D. C, “Ductal carcinoma in situ: Terminology, classification, and natural history,” *JNCI Monographs* **2010** no. 41, (Oct., 2010) 134–138. https://doi.org/10.1093/jncimonographs/lgq035.

[8] Collins L. C., Tamimi R. M., Baer H. J., Connolly J. L., Colditz G. A. and Schnitt S. J, “Outcome of patients with ductal carcinoma in situ untreated after diagnostic biopsy,” *Cancer* **103** no. 9, (2005) 1778–1784. https://doi.org/10.1002/cncr.20979.

[9] Sanders M. E., Schuyler P. A., Dupont W. D. and Page D. L, “The natural history of low-grade ductal carcinoma in situ of the breast in women treated by biopsy only revealed over 30 years of long-term follow-up,” *Cancer* **103** no. 12, (2005) 2481–2484. https://doi.org/10.1002/cncr.21069.

[10] Eusebi V., Feudale E., Foschini M. P., Micheli A., Conti A., Riva C., Di Palma S. and Rilke F, “Long-term follow-up of in situ carcinoma of the breast,” *Semin Diagn Pathol* **11** no. 3, (Aug, 1994) 223–235.

[11] Tamimi R. M., Colditz G. A., Hazra A., Baer H. J., Hankinson S. E., Rosner B., Marotti J., Connolly J. L., Schnitt S. J. and Collins L. C, "Traditional breast cancer risk factors in relation to molecular subtypes of breast cancer," *Breast Cancer Research and Treatment* **131** no. 1, (Aug., 2011) 159–167. `https://doi.org/10.1007/s10549-011-1702-0`.

[12] "Comprehensive molecular portraits of human breast tumours," *Nature* **490** no. 7418, (Sept., 2012) 61–70. `https://doi.org/10.1038/nature11412`.

[13] Dieci M. V., Orvieto E., Dominici M., Conte P. and Guarneri V, "Rare breast cancer subtypes: Histological, molecular, and clinical peculiarities," *The Oncologist* **19** no. 8, (June, 2014) 805–813. `https://doi.org/10.1634/theoncologist.2014-0108`.

[14] "U.s. census bureau quickfacts: Maine." `https://www.census.gov/quickfacts/ME`.

[15] Birdwell R. L., Ikeda D. M., O'Shaughnessy K. F. and Sickles E. A, "Mammographic characteristics of 115 missed cancers later detected with screening mammography and the potential utility of computer-aided detection," *Radiology* **219** no. 1, (Apr., 2001) 192–202. `https://doi.org/10.1148/radiology.219.1.r01ap16192`.

[16] Marin Z., Batchelder K. A., Toner B. C., Guimond L., Gerasimova-Chechkina E., Harrow A. R., Arneodo A. and Khalil A, "Mammographic evidence of microenvironment changes in tumorous breasts," *Medical Physics* **44** no. 4, (2017) 1324–1336. `http://dx.doi.org/10.1002/mp.12120`.

[17] Gerasimova-Chechkina E., Toner B., Marin Z., Audit B., Roux S. G., Argoul F., Khalil A., Gileva O., Naimark O. and Arneodo A, "Comparative multifractal analysis of dynamic infrared thermograms and x-ray mammograms enlightens changes in the environment of malignant tumors," *Frontiers in Physiology* **7** (Aug, 2016) . `https://doi.org/10.3389/fphys.2016.00336`.

[18] Kestener P., Lina J. M., Saint-Jean P. and Arneodo A, "Wavelet-based multifractal formalism to assist in diagnosis in digitized mammograms," *Image Analysis & Stereology* **20** no. 3, (2001) 169–174. `https://www.ias-iss.org/ojs/IAS/article/view/674`.

[19] Bissell M. J and Hines W. C, "Why don't we get more cancer? a proposed role of the microenvironment in restraining cancer progression," *Nature Medicine* **17** no. 3, (Mar, 2011) 320–329. `https://doi.org/10.1038/nm.2328`.

[20] Vronique M.-S, "The stem cell niche: The black master of cancer," in *Cancer Stem Cells Theories and Practice*. InTech, Mar, 2011. `https://doi.org/10.5772/13622`.

[21] Alowami S., Troup S., Al-Haddad S., Kirkpatrick I. and Watson P. H, "Mammographic density is related to stroma and stromal proteoglycan expression," *Breast Cancer Research* **5** no. 5, (Oct, 2003) . `https://doi.org/10.1186/bcr622`.

[22] Kinzler K. W and Vogelstein B, "Landscaping the cancer terrain," *Science* **280** no. 5366, (May, 1998) 1036–1037.

[23] Plourde S. M., Marin Z., Smith Z. R., Toner B. C., Batchelder K. A. and Khalil A, "Computational growth model of breast microcalcification clusters in simulated mammographic environments," *Computers in Biology and Medicine* **76** (Sep, 2016) 7–13. https://doi.org/10.1016/j.compbiomed.2016.06.020.

[24] Stone J. E., Gohara D. and Shi G, "OpenCL: A parallel programming standard for heterogeneous computing systems," *Computing in Science & Engineering* **12** no. 3, (May, 2010) 66–73. https://doi.org/10.1109/mcse.2010.69.

[25] Graham R., Shipman G., Barrett B., Castain R., Bosilca G. and Lumsdaine A, "Open MPI: A high-performance, heterogeneous MPI," in *2006 IEEE International Conference on Cluster Computing*. IEEE, 2006. https://doi.org/10.1109/clustr.2006.311904.

[26] Muzy J. F., Bacry E. and Arneodo A, "Wavelets and multifractal formalism for singular signals: Application to turbulence data," *Physical Review Letters* **67** no. 25, (Dec, 1991) 3515–3518. https://doi.org/10.1103/physrevlett.67.3515.

[27] Arnéodo A., Decoster N. and Roux S, "A wavelet-based method for multifractal image analysis. I. methodology and test applications on isotropic and anisotropic random rough surfaces," *The European Physical Journal B* **15** no. 3, (May, 2000) 567–600. https://doi.org/10.1007/s100510051161.

[28] Kestener P., Lina J. M., Saint-Jean P. and Arneodo A, "Wavelet-based multifractal formalism to assist in diagnosis in digitized mammograms," *Image Analysis & Stereology* **20** no. 3, (May, 2011) 169. https://doi.org/10.5566/ias.v20.p169-174.

[29] Kestener P and Arneodo A, "Three-dimensional wavelet-based multifractal method: The need for revisiting the multifractal description of turbulence dissipation data," *Physical Review Letters* **91** no. 19, (Nov, 2003) . https://doi.org/10.1103/physrevlett.91.194501.

[30] Kestener P and Arneodo A, "Generalizing the wavelet-based multifractal formalism to random vector fields: Application to three-dimensional turbulence velocity and vorticity data," *Physical Review Letters* **93** no. 4, (Jul, 2004) . https://doi.org/10.1103/physrevlett.93.044501.

[31] Karahaliou A. N., Boniatis I. S., Skiadopoulos S. G., Sakellaropoulos F. N., Arikidis N. S., Likaki E. A., Panayiotakis G. S. and Costaridou L. I, "Breast cancer diagnosis: Analyzing texture of tissue surrounding microcalcifications," *IEEE Transactions on Information Technology in Biomedicine* **12** no. 6, (Nov, 2008) 731–738.

[32] Ke L., He W. and Kang Y, "Mass auto-detection in mammogram based on wavelet transform modulus maximum," in *2009 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pp. 5760–5763. Sept, 2009.

[33] Eltoukhy M. M., Faye I. and Samir B. B, "A comparison of wavelet and curvelet for breast cancer diagnosis in digital mammogram," *Computers in Biology and Medicine*

40 no. 4, (Apr, 2010) 384–391.
https://doi.org/10.1016/j.compbiomed.2010.02.002.

[34] Kilic N., Gorgel P., Ucan O. N. and Sertbas A, "Mammographic mass detection using wavelets as input to neural networks," *Journal of Medical Systems* **34** no. 6, (Dec, 2010) 1083–1088. https://doi.org/10.1007/s10916-009-9326-1.

[35] Tsai N.-C., Chen H.-W. and Hsu S.-L, "Computer-aided diagnosis for early-stage breast cancer by using wavelet transform," *Computerized Medical Imaging and Graphics* **35** no. 1, (2011) 1 – 8.
http://www.sciencedirect.com/science/article/pii/S0895611110000856.

[36] AlZubi S., Islam N. and Abbod M, "Multiresolution analysis using wavelet, ridgelet, and curvelet transforms for medical image segmentation," *International Journal of Biomedical Imaging* **2011** (2011) 1–18. https://doi.org/10.1155/2011/136034.

[37] Issac Niwas S., Palanisamy P., Chibbar R. and Zhang W. J, "An expert support system for breast cancer diagnosis using color wavelet features," *Journal of Medical Systems* **36** no. 5, (Oct, 2012) 3091–3102. https://doi.org/10.1007/s10916-011-9788-9.

[38] Eltoukhy M. M., Faye I. and Samir B. B, "A statistical based feature extraction method for breast cancer diagnosis in digital mammogram using multiresolution representation," *Computers in Biology and Medicine* **42** no. 1, (2012) 123 – 128.
http://www.sciencedirect.com/science/article/pii/S0010482511002125.

[39] Batchelder K. A., Tanenbaum A. B., Albert S., Guimond L., Kestener P., Arneodo A. and Khalil A, "Wavelet-based 3d reconstruction of microcalcification clusters from two mammographic views: New evidence that fractal tumors are malignant and euclidean tumors are benign," *PLoS ONE* **9** no. 9, (Sep, 2014) e107580.
https://doi.org/10.1371/journal.pone.0107580.

[40] Gerasimova E., Audit B., Roux S. G., Khalil A., Gileva O., Argoul F., Naimark O. and Arneodo A, "Wavelet-based multifractal analysis of dynamic infrared thermograms to assist in early breast cancer diagnosis," *Frontiers in Physiology* **5** (May, 2014) . https://doi.org/10.3389/fphys.2014.00176.

[41] Grant J., Verrill C., Coustham V., Arneodo A., Palladino F., Monier K. and Khalil A, "Perinuclear distribution of heterochromatin in developing c. elegans embryos," *Chromosome Research* **18** no. 8, (Dec, 2010) 873–885.
https://doi.org/10.1007/s10577-010-9175-2.

[42] Kestener P., Conlon P. A., Khalil A., Fennell L., McAteer R. T. J., Gallagher P. T. and Arneodo A, "CHARACTERIZING COMPLEXITY IN SOLAR MAGNETOGRAM DATA USING a WAVELET-BASED SEGMENTATION METHOD," *The Astrophysical Journal* **717** no. 2, (Jun, 2010) 995–1005.
https://doi.org/10.1088/0004-637x/717/2/995.

[43] Khalil A., Aponte C., Zhang R., Davisson T., Dickey I., Engelman D., Hawkins M. and Mason M, "Image analysis of soft-tissue in-growth and attachment into highly porous alumina ceramic foam metals," *Medical Engineering & Physics* **31** no. 7, (Sep, 2009) 775–783. https://doi.org/10.1016/j.medengphy.2009.02.007.

[44] Khalil A., Grant J. L., Caddle L. B., Atzema E., Mills K. D. and Arneodo A, "Chromosome territories have a highly nonspherical morphology and nonrandom positioning," *Chromosome Research* **15** no. 7, (Nov, 2007) 899–916. https://doi.org/10.1007/s10577-007-1172-8.

[45] Khalil A., Joncas G., Nekka F., Kestener P. and Arneodo A, "Morphological analysis of HiFeatures. II. wavelet-based multifractal formalism," *The Astrophysical Journal Supplement Series* **165** no. 2, (Aug, 2006) 512–550. https://doi.org/10.1086/505144.

[46] Moore K. A, "Characterization of mammographic breast lesions and their microenvironment: an application of a wavelet-based multifractal formalism," Master's thesis, The University of Maine, 2013. https://digitalcommons.library.umaine.edu/etd/1910.

[47] McAteer R. T. J., Kestener P., Arneodo A. and Khalil A, "Automated detection of coronal loops using a wavelet transform modulus maxima method," *Solar Physics* **262** no. 2, (Mar, 2010) 387–397. https://doi.org/10.1007/s11207-010-9530-7.

[48] Snow C. J., Goody M., Kelly M. W., Oster E. C., Jones R., Khalil A. and Henry C. A, "Time-lapse analysis and mathematical characterization elucidate novel mechanisms underlying muscle morphogenesis," *PLoS Genetics* **4** no. 10, (Oct, 2008) e1000219. https://doi.org/10.1371/journal.pgen.1000219.

[49] Snow C. J., Peterson M. T., Khalil A. and Henry C. A, "Muscle development is disrupted in zebrafish embryos deficient for fibronectin," *Developmental Dynamics* **237** no. 9, (Sep, 2008) 2542–2553. https://doi.org/10.1002/dvdy.21670.

[50] Nicolis O., Ramírez-Cobo P. and Vidakovic B, "2d wavelet-based spectra with applications," *Computational Statistics & Data Analysis* **55** no. 1, (Jan, 2011) 738–751. https://doi.org/10.1016/j.csda.2010.06.020.

[51] Decoster N., Roux S. and Arnéodo A, "A wavelet-based method for multifractal image analysis. II. applications to synthetic multifractal rough surfaces," *The European Physical Journal B* **15** no. 4, (Jun, 2000) 739–764. https://doi.org/10.1007/s100510051179.

[52] Roux S., Arnéodo A. and Decoster N, "A wavelet-based method for multifractal image analysis. III. applications to high-resolution satellite images of cloud structure," *The European Physical Journal B* **15** no. 4, (Jun, 2000) 765–786. https://doi.org/10.1007/s100510051180.

[53] Qian W., Kallergi M., Clarke L. P., Li H.-D., Venugopal P., Song D. and Clark R. A, “Tree structured wavelet transform segmentation of microcalcifications in digital mammography,” *Medical Physics* **22** no. 8, (Aug, 1995) 1247–1254. https://doi.org/10.1118/1.597562.

[54] Bacry E., Muzy J. F. and Arnï£¡odo A, “Singularity spectrum of fractal signals from wavelet analysis: Exact results,” *Journal of Statistical Physics* **70** no. 3-4, (Feb., 1993) 635–674. https://doi.org/10.1007/bf01053588.

[55] Muzy J. F., Bacry E. and Arneodo A, “Multifractal formalism for fractal signals: The structure-function approach versus the wavelet-transform modulus-maxima method,” *Physical Review E* **47** no. 2, (Feb, 1993) 875–884. https://doi.org/10.1103/physreve.47.875.

[56] Muzy J., Barcy E. and Arneodo A, “The multifractal formalism revisited with wavelets,” *International Journal of Bifurcation and Chaos* **04** no. 02, (Apr, 1994) 245–302. https://doi.org/10.1142/s0218127494000204.

[57] *Academic Press Library in Signal Processing, Volume 1: Signal Processing Theory and Machine Learning.* Academic Press, 2013. https://www.amazon.com/Academic-Press-Library-Signal-Processing/dp/0123965020?SubscriptionId=AKIAIOBINVZYXZQZ2U3A&tag=chimbori05-20&linkCode=xm2&camp=2025&creative=165953&creativeASIN=0123965020.

[58] Triola M. F, *Essentials of Statistics (4th Edition) (Triola Statistics Series).* Pearson, 2011. https://www.amazon.com/Essentials-Statistics-4th-Triola/dp/0321641493?SubscriptionId=AKIAIOBINVZYXZQZ2U3A&tag=chimbori05-20&linkCode=xm2&camp=2025&creative=165953&creativeASIN=0321641493.

[59] Toner B. C, “Geometric properties of cells exhibiting huntington’s disease,” Master’s thesis, The University of Maine, 2015. https://digitalcommons.library.umaine.edu/etd/2304/.

[60] Jain A. K, *Fundamentals of Digital Image Processing.* Pearson, 1988. https://www.amazon.com/Fundamentals-Digital-Image-Processing-Anil/dp/0133361659?SubscriptionId=AKIAIOBINVZYXZQZ2U3A&tag=chimbori05-20&linkCode=xm2&camp=2025&creative=165953&creativeASIN=0133361659.

[61] Pratt W. K, *Digital Image Processing: PIKS Inside, 3rd Edition.* Wiley-Interscience, 2001. https://www.amazon.com/Digital-Image-Processing-PIKS-Inside/dp/0471374075?SubscriptionId=AKIAIOBINVZYXZQZ2U3A&tag=chimbori05-20&linkCode=xm2&camp=2025&creative=165953&creativeASIN=0471374075.

[62] B. J, *Digital Image Processing.* Springer, New York, New York, 6th ed., 2005.

[63] Eckhardt U and Latecki L. J, “Digital topology,” 1994.

[64] Suzuki S and Abe K, “Topological structural analysis of digitized binary images by border following,” *Computer Vision, Graphics, and Image Processing* **30** no. 1, (1985) 32–46.

[65] Braden B, “The surveyor's area formula,” *The College Mathematics Journal* **17** no. 4, (Sept., 1986) 326. `https://doi.org/10.2307/2686282`.

[66] Graham R and Yao F, “Finding the convex hull of a simple polygon,” *Journal of Algorithms* **4** no. 4, (1983) 324–331.

[67] Sklansky J, “Finding the convex hull of a simple polygon,” *Pattern Recognition Letters* **1** no. 2, (1982) 79–83.

[68] Barber C. B., Dobkin D. P., Dobkin D. P. and Huhdanpaa H, “The quickhull algorithm for convex hulls,” *ACM Trans. Math. Softw.* **22** no. 4, (Dec., 1996) 469–483. `http://doi.acm.org/10.1145/235815.235821`.

[69] Shin S and Woo T, “Finding the convex hull of a simple polygon in linear time,” *Pattern Recognition* **19** no. 6, (1986) 453–458.

[70] Khalil A., Grant J., Caddle L., Atzema E., Mills K. and Arneodo A, “Chromosome territories have a highly nonspherical morphology and nonrandom positioning,” *Chromosome Res* **15** (2007) 899–916.

[71] Friel J, *Practical Guide to Image Analysis.* ASM International, Materials Park, Ohio, 2nd ed., 2000.

[72] Heilbronner R and Barrett S, *Image Analysis in Earth Sciences: Microstructures and Textures of Earth Materials.* Springer, New York, New York, 2013.

[73] Raja J., Muralikrishnan B. and Fu S, “Recent advances in separation of roughness, waviness and form,” *Precision Engineering* **26** no. 2, (2002) 222–235.

[74] Teague M. R, “Image analysis via the general theory of moments∗,” *J. Opt. Soc. Am.* **70** no. 8, (Aug, 1980) 920–930. `http://www.osapublishing.org/abstract.cfm?URI=josa-70-8-920`.

[75] Ming-Kuei Hu , “Visual pattern recognition by moment invariants,” *IRE Transactions on Information Theory* **8** no. 2, (February, 1962) 179–187.

[76] Mann H. B and Whitney D. R, “On a test of whether one of two random variables is stochastically larger than the other,” *The Annals of Mathematical Statistics* **18** no. 1, (Mar., 1947) 50–60. `https://doi.org/10.1214/aoms/1177730491`.

APPENDIX A
CONVOLUTION APPENDIX

Before we delve into the mechanics of the WTMM, it is crucial to understand what a convolution is and the difference between a convolution performed in Fourier space and a convolution performed in discrete space. This section serves as a primer to help the reader visualize the purpose of a convolution and develop an understanding of how this procedure is performed in both discrete and continuous spaces. It is important to note that the wavelet transform has no formal discrete definition. This is because the mathematics of the wavelet transform break down in discrete space, and thus must be performed in a continuous space. For that reason, we will approach this section from both an algorithmic and mathematical perspective. In this way, the reader will be able gain an understanding of the underlying mechanics from a lay perspective.

The first step of the wavelet transform is to perform a convolution with some kernel ψ over a digital image f. A kernel can be thought of as smaller image, or filter, with r rows and c columns. Ideally, the elements should sum to zero, although this is not always the case. The convolution of f at pixel $\{x, y\}$, is calculated as

$$g_{x,y} = \psi * f_{x,y} = \sum_{i=1}^{r} \sum_{j=1}^{c} f_{(y-\lfloor \frac{r}{2} \rfloor + i),(x-\lfloor \frac{c}{2} \rfloor + j)} \psi_{i,j} \tag{A.1}$$

where $g_{x,y}$ is the output value of the single pixel of the convolved region. This convolution must be applied to all of the pixels in the image f. If the elements of the convolution do not add up to zero, one must normalize the convolution by the sum of the kernel,

$$g_{x,y} = \psi * f_{x,y} = \left| \frac{\sum_{i=1}^{r} \sum_{j=1}^{c} f_{(y-\lfloor \frac{r}{2} \rfloor + i),(x-\lfloor \frac{c}{2} \rfloor + j)} \psi_{i,j}}{\sum_{k \in \psi} \psi_k} \right| \tag{A.2}$$

For speed, this transformation is often performed by a convolution utilizing the Fourier transformation. An advantage of using the Fourier transformation is that it helps to prevent edge artifacts of the original image due to the finite kernel size. Note that by using A.1, there is no mathematical definition for those elements whose index exceeds the bound of f. That is, there are no elements outside of the bounds of f, e.g., $(-1,-1)$. In other words, a 3×3 pixel kernel will 'erode' away 2 rows and 2 columns from the original image, as shown in the following example, and a 5×5 pixel kernel will create convolution artifacts 4 rows and 4 columns from the original image. More specifically, a $k \times k$ kernel will remove $k-1$ rows and columns from the original image. Other strategies exist for dealing with the edges, such as wrapping the image or duplicating the values around the edge, but these can cause the emergence of image artifacts or defects.

To illustrate the difficulty that ψ poses at the edges, suppose we have the kernel

$$\psi = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix},$$

and the image

$$f = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}.$$

If we convolve these two matrices together, the result will be

$$g = \psi * f = \begin{bmatrix} 3 & 2 & 2 & 2 & 2 & 3 \\ 2 & 1 & 1 & 1 & 1 & 2 \\ 2 & 1 & 1 & 1 & 1 & 2 \\ 2 & 1 & 1 & 1 & 1 & 2 \\ 3 & 2 & 2 & 2 & 2 & 3 \end{bmatrix}.$$

In this case, the sum of the elements of ψ is 1. This means that the net effect of the convolution should be 1 on the unit matrix (a matrix of all ones). However, along the edges of g there exist non-1 values. This is because in this case we treated the elements outside of

f as though they were 0 valued elements, that is

$$f' = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix},$$

where the elements of value 1 are the original matrix. Thus the values obtained at the edges are not necessarily the 'true' values but rather are artifacts created by the inability of ψ to handle mathematically what is happening outside of f.

Next we will take a look at a convolution with a larger kernel. In this case we will use the partial derivatives of the 2D Gaussian

$$\psi_1(x,y) = \frac{\partial \phi(x,y)}{\partial x}, \tag{A.3}$$

and

$$\psi_2(x,y) = \frac{\partial \phi(x,y)}{\partial y}. \tag{A.4}$$

We will call this convolution of f with ψ, $\psi * f = T_\psi[f](a)$ for some size scale a. Figures A.2b and A.2c show the effects of applying A.1a and A.1b to A.2a, respectively. Note Figures A.2b and A.2c were convolved without the Fourier transform and as a result we can see some distortion along the y edges of Figure A.2b. Likewise, we can see a similar distortion along the x edges of Figure A.2c. These distortions are the artifacts mentioned earlier. These images are included here to emphasize the importance of utilizing the Fourier transform to perform the convolution. Our analysis did not suffer from these artifacts as we utilized the convolution with the Fourier transform. Equivalently, one could first apply a Gaussian to the image then take the derivative in the x and y direction of the

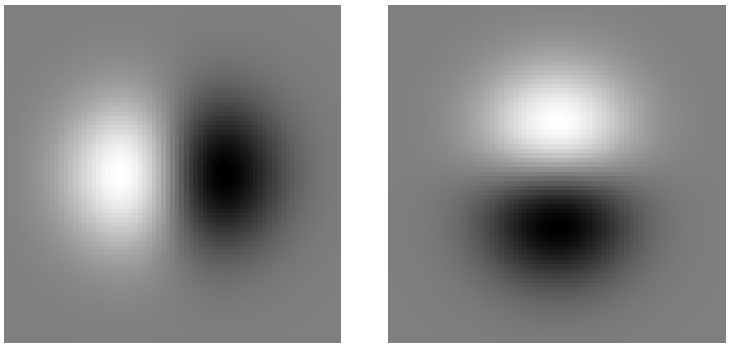

(a) (b)

Figure A.1: Partial derivatives of the 2D Gaussian.
Figure A.1a: Graphical representation of the first derivative of the Gaussian taken in the x direction. Figure A.1b: Graphical representation of the first derivative of the Gaussian taken in the y direction.

smoothed image. The reason why taking the derivative of the Gaussian first is the default method is that this reduces the number of operations performed on the image.

In this analysis, we are not limited to a single size scale of the 2D Gaussian. Figure A.2 depicts the effect of applying the 2D Gaussian at multiple scales. One can see that as we increase the scale of the Gaussian, the finer details become muted, overpowered by the more prominent features contained in the image.

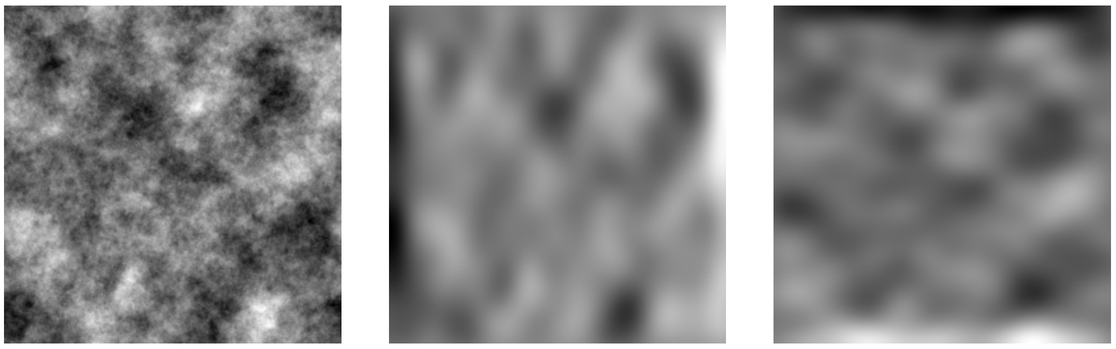

(a) (b) (c)

Figure A.2: An image alongside its partial derivatives.
Figure A.2a is an image featuring Brownian noise. Figure A.2b is the convolution of A.2a with A.1a. Figure A.2c is the convolution of A.2a with A.1b

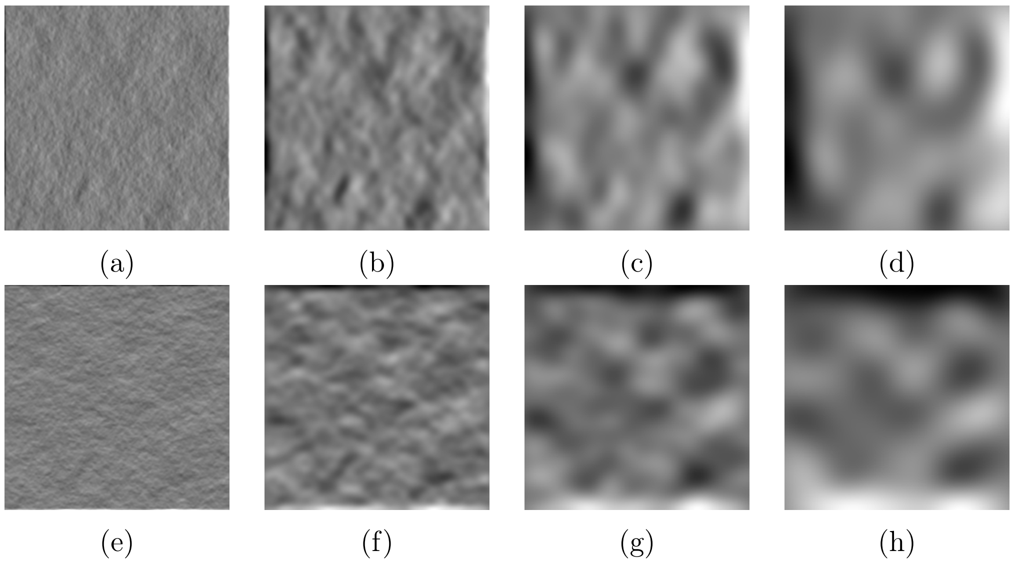

(a) (b) (c) (d)
(e) (f) (g) (h)

Figure A.3: Gaussian filters applied to A.2a at scales $a = \{7, 29, 73, 127\}$.
From left to right, the figure depicts 2D Gaussian kernels at larger and larger scales, $a = \{7, 29, 73, 127\}$. Figures A.3a–A.3d are the x derivatives of the Gaussian filters. Figures A.3e–A.3h are the y derivatives of the Gaussian filters.

APPENDIX B

CODE APPENDIX

B.1 AWTMM OpenCL Code

```
#pragma OPENCL EXTENSION cl_khr_fp64 : enable



//============================================================================
// This section of code is where the structs live
//
//============================================================================

/***
 * This structure contains an (x,y) ordered pair.
 */
struct PointF{
  double x;     //The x coordinate of the ordered pair.
  double y;     //The y coordinate of the ordered pair.
};

/***
 * This structure contains a set of 50 ordered pairs
 * e.g. {(x_1,y_1),(x_2,y_2),(x_3,y_3),...,(x_50,y_50)}
 */
struct PointFList{
    struct PointF vData[50];     //The array of ordered pairs
    unsigned long vDataSize;     //The number of ordered pairs (preallocated to 50)
};

/***
 * This structure is for holding pairs of elements indexes
 */
struct SizeTPair{
  unsigned long first;     //The index of the first element
  unsigned long second;    //The index of the second element.
};

/***
 * This structure is used for containing the linear model data
 */
struct LinearModel{

    double vBetaHat;         //Slope of the regression line.
    double vAlphaHat;        //Y-Intercept
    double vSxx;             //Standard deviation x
    double vSyy;             //Standard deviation y
    double vRSquared;        //R^2 value
    double vRVal;            //R value

    struct PointF vMeanXY;   //The mean of the x,y data.
    double vSumXY;           //Sum xy
    double vSumXSquared;     //Sum x^2
    double vSumYSquared;     //Sum y^2

    double vMinX;            //Min x range
    double vMaxX;            //Max x range

};

/**
 * This structure contains the 32 regression lines
 */
struct RegressionList{

    double vAMin;
    double vAMax;
    double vSlope[32];
    double vRSquared[32];
    double vIntercept[32];
    double vDeltaSlope[32];
    double vDeltaRSquared[32];
    double vDeltaIntercept[32];

};

/***
 * This structure is for holding the calculated curvedata
 */
struct CurveData{
    double vHValue;             //h(0,a)
    double vDValue;             //D(0,a)
    double vR2Value;            //How well the linear model fit

    double vAMin;               //min a for the h(q,a) and D(q,a) curves
    double vAMax;               //max a for the h(q,a) and D(q,a) curves

    double vFitness;            //The fitness of the curve

    char vGroup;                //R,Y,B,O,N
    char vComment[100];         //No scaling reason

    double vWeightedDelta;      //The weigthed delta
    double vWeightedH;          //Weighted h(0,a)
    double vWeightedD;          //Weighted D(0,a)
    double vWeightedR2;         //How well the weighted linear model fit
    double vAvgDelta;           //On average, how far spred out are the data
    double vAvgH;               //What is the average h-value
    double vAvgD;               //What is the average D-value
    double vAvgR2;              //What is the average R^2

    double vStdevWeightedDelta; //The standard deviation of the weghted delta
    double vStdevWeightedH;     //The standard deviation of the weighted h(0,a)
    double vStdevWeightedD;     //The standard deviation of the weighted D(0,a)
    double vStdevWeightedR2;    //The standard deviation of the weighted D(0,a)
    double vStdevDelta;         //The standard deviation of the weighted linear model fit
    double vStdevH;             //The standard deviation of the average h-value
    double vStdevD;             //The standard deviation of the average D-value
    double vStdevR2;            //The standard deviation of average R^2

    bool vSuitable;              //Flag for signaling if the curvedata is valid

};

//=============================================================================
// This section of code is where the statistical computations live
//
//=============================================================================


/***
 * This function computes the difference between the elements of an array
 * The variable aDst is modified
 * @Param aData - The data to compute the delta of
 * @Param aDst  - The destination vector for the computed deltas
 * @Param aArraySize - The size of the data and destination arrays.
 */
void get_delta(__global double* aData,
               __global double* aDst,
               unsigned long aArraySize){

    //Subtract the first element from the last element
    aDst[0] = fabs(aData[0]-aData[aArraySize-1]);

    //Subtract each element from the next
    for(unsigned long i = 1; i < aArraySize; i++){
        aDst[i] = fabs(aData[i]-aData[i-1]);
    }

}

/**
 * Sums the values in an array.
 * @param aData - Array to find the sum of.
 * @param aDataSize - The number of elements in aData
 * @return The sum of all the elements of the vector.
 */
double sum_vector(__global double* aData,
                  unsigned long aDataSize){

    double lRet = 0;

    for(unsigned long i = 0; i < aDataSize; i++){
        lRet += aData[i];
    }

    return lRet;
}

/**
 * Calculates the average of an ordered set of pairs.
 * mu_x = (1/(f-l)) âĹŚ((x_i) | i=f to l)
 * mu_y = (1/(f-l)) âĹŚ((y_i) | i=f to l)
 * where x_i = aData.x, y_i = aData.y, f = aFirst, and l = aLast
 *
 * @param aData -Array of points.
 * @param aFirst -First element to calculate the mean from
 * @param aLast - Last element to calculate the mean to
 * @return Returns a PointF that represents the mean of the x and y ordered pairs.
 *         between the first and last elments.
 */
struct PointF calc_average_pt(__global struct PointF* aData,
                              unsigned long aFirst,
                              unsigned long aLast){

    struct PointF locAverage = {0,0};

    long unsigned int lSize = aLast-aFirst+1;

    for(unsigned long i = aFirst; i < aLast+1; i++){
        locAverage.x += aData[i].x/(double)lSize;
        locAverage.y += aData[i].y/(double)lSize;
    }

    return locAverage;
}

/**
 * Computes the mean of the array. The mean is computed as:
 * \sum_{i=0}^{k}(aData_i)/aDataSize.
 * @param aData - Data to average.
 * @param aDataSize - The number of elements in aData.
 * @return The average.
 */
double mean(__global double* aData,
            unsigned long aDataSize){

    double lRet = 0;
    for(unsigned long i = 0; i < aDataSize; i++){
        lRet += aData[i];
    }

    return lRet/aDataSize;

}

/**
 * Computes the weighted mean of the array. The mean is computed as:
 * \sum_{i=0}^{k}(aData_i*aWeights_i)/\sum_{i=0}^{k}(aWeights_i).
 * If the two vectors are not of the same size then a runtime error will be thrown.
 * @param aData Data to average.
 * @param aWeights Weight of the corresponding data points.
 * @return Weighted average.
 */
double weighted_mean(__global double* aData,
                     __global double* aWeights,
                     unsigned long aArraySize){

    double lRet = 0;
    double lWeightSum = sum_vector(aWeights, aArraySize);

    for(unsigned long i = 0; i < aArraySize; i++){
        lRet += (aData[i]*aWeights[i])/lWeightSum;
    }

    return lRet;
}

/**
 * Calculates the simple variance of a array.
 * @param aData -Array we want to find the variance of.
 * @param aDataSize -Number of elements in aData
 * @return Returns the variance of aData.
 */
double variance(__global double* aData,
                unsigned long aDataSize){

    double lAverage = mean(aData,aDataSize);
    double lResults = 0;

    for(unsigned long i=0; i<aDataSize; i++){
        lResults += (aData[i]-lAverage)*(aData[i]-lAverage);
    }

    lResults = lResults/(aDataSize-1);

    return lResults;

}

/**
 * Calculates the simple standard deviation of a double array.  This is accomplished by
 * sqrt(Variance)
 * @param aData -Array we want to find the standard deviation of.
 * @param aDataSize -Number of elements in aData
 * @return Returns the standard deviation of aData.
*/
double stdev(__global double* aData,
             unsigned long aDataSize){
    return sqrt(variance(aData,aDataSize));
}

/**
 * Computes the weighted standard deviation of a vector. The weighted standard deviation
 * is computed as: \sum_{i=0}^{k}((aData_i-WM)^2*aWeights_i)/(WS-1) where WM is the weighted mean
 * and WS is the weighted sum.
 * If the two vectors are not of the same size then a runtime error will be thrown.
 * @param aData - The data to compute the weighted standard deviation of
 * @param aWeights - The weights to compute the standard deviation of
 * @return The standard deviation of the data
 */
double weighted_stdev(__global double* aData,
                      __global double* aWeights,
                      unsigned long aArraySize){

    double lRet = 0;
    double lWeightedSum = sum_vector(aWeights,aArraySize);
    double lWeightedMean = weighted_mean(aData, aWeights, aArraySize);

    for(unsigned long i = 0; i < aArraySize; i++){
        lRet += ((aData[i]-lWeightedMean)*(aData[i]-lWeightedMean))*aWeights[i]/(lWeightedSum-1);
    }

    return sqrt(lRet);

}



//==============================================================================
// This section of code us used to classify the h(q,a) and D(q,a) curves
// It containes the OpenCL kernel for classifying and the associated fitness
// functions.
//==============================================================================


/***
 * Classifies the h-value input into its associated tissue group (B,Y,R)
 * B < .45; R > .55; and Y is between .45 and .55.
 *
 * @param aHValue - The h-value to classify
 * @return Returns the character (B,Y, or R) associated with the tissue type.
 */
char classify_h_group(double aHValue){

    char lRet = 0x00;

    if(aHValue > .55){
        lRet = 'R';
    } else if (aHValue < .45){
        lRet = 'B';
    } else {
        lRet = 'Y';
    }

    return lRet;

}

/***
 * Maps a number into the range aMin-aMax
 *
 * @param aVal - The value to remap
 * @param aMin - The minimum value of the new range
 * @param aMax - The maximum value of the new range
 * @return Returns a scaled version of aVal e.g. (aVal-aMin)/(aMax-aMin)
 */
double scale_value(double aVal, double aMin, double aMax){
    return (aVal-aMin)/(aMax-aMin);
}

/***
 * Computes the fitness of aValue.
 * If the fitness is positve then aVal is fit otherwise it is not fit.
 * The more negative the fitness, the less fit the value is.
 *
 * @param aVal - The value to compute the fitness of
 * @param aMin - The minimum fit value
 * @param aMax - The maximum fit value
 * @return Returns a double representing the fitness.
 */
double calc_fit(double aVal, double aMin, double aMax){

    double lVal = scale_value(aVal,aMin,aMax);
    double lRet = 0;

    if(lVal > 1){
        lRet = 1-lVal;
    } else if (lVal < 0){
        lRet = lVal;
    } else {
        lRet = 1;
    }

    return lRet;
}

/***
 * Classifies the h(q,a) and D(q,a) curves.
 * This function determines if the curves produce a valid fit to the modle
 * If it does then aDst->vSuitable will be true
 * aDst is modified
 *
 * @param aDst - The destination of the calculated fitness
 * @param aMinWeightedR2 - The fitness for checking if R^2 and weighted R^2 are suitalbe
 * @param aMinHValue - Minimum suitable h-value.
 * @param aMaxHValue - Maximum suitable h-value.
 * @param aMinDValue - Minimum suitable D-value.
 * @param aMaxDValue - Maximum suitable D-value.
 * @param aThreshWeightedH - Checks if weigthed standarad deviation of h is suitable
 */
void classify_curvedata(__global struct CurveData* aDst,
                        double aMinWeightedR2,
                        double aMinHValue,
                        double aMaxHValue,
                        double aMinDValue,
                        double aMaxDValue,
                        double aThreshWeightedH){

    //Initialize our return values
    bool lSmallDelta = false;
    bool lBadH = false;
    bool lBadHStdev = false;
    bool lBadD = false;
    bool lNoScaling = false;
    bool lBadR2 = false;
    bool lBadWR2 = false;
    aDst->vSuitable = true;


    for(int i = 0; i < 5; i++){
        aDst->vComment[i] = '-';
    }
    aDst->vComment[5] = 0;

    //Compute the fitness of the curve
    double lFitCalc = sqrt((aDst->vR2Value*aDst->vR2Value)
                      +(aDst->vWeightedR2*aDst->vWeightedR2));
    double lScaleH = calc_fit(aDst->vHValue,aMinHValue,aMaxHValue);
    double lScaleD = calc_fit(aDst->vDValue,aMinDValue,aMaxDValue);
    double lScaleWSDH = calc_fit(aDst->vStdevWeightedH,0,aThreshWeightedH);

    aDst->vFitness = (lScaleH+lFitCalc+lScaleD+lScaleWSDH)/(4.414213562);


    //Is the h-value suitable?
    if( aDst->vHValue < aMinHValue || aDst->vHValue > aMaxHValue){

        aDst->vSuitable = false;
        aDst->vComment[2]= 'H';

    }

    //Is the weigthed standarad deviation of h suitable?
    if( aDst->vStdevWeightedH > aThreshWeightedH){
        aDst->vSuitable = false;
        aDst->vComment[3]= 'S';
    }

    //Is our D-value suitable?
    if( aDst->vDValue < aMinDValue || aDst->vDValue > aMaxDValue ){
        aDst->vSuitable = false;
        aDst->vComment[4]= 'D';
    }

    //Is our R^2 value suitalbe?
    if(aDst->vR2Value < aMinWeightedR2){
        aDst->vSuitable = false;
        aDst->vComment[0]= 'R';
    }

    //Is our Weighted R^2 value suitable?
    if(aDst->vWeightedR2 < aMinWeightedR2){

        aDst->vSuitable = false;
        aDst->vComment[1]= 'W';
    }

    //Were any of the suitablity conditions not met?
    //If so classify as No scaling (N)
    if(!aDst->vSuitable){
        aDst->vGroup = 'N';

    //Otherwise classify the h-value
    } else {
        aDst->vGroup = classify_h_group(aDst->vHValue);
    }



    //return aDst.vSuitable;

}


/***
 * This function classifies the h(q,a) and D(q,a) curves
 * The variable aDst is modified
 * @Param aHData - The h(q,a) curves
 * @Param aDData - The D(q,a) curves
 * @Param aWeights - Weights for mean and stdev
 * @Param aWeightSize - Number of weights
 * @Param aDst - The classifier destination
 * @Param aParams - The classification parmeters
 */
__kernel void get_curvedata(__global struct RegressionList* aHData,
                            __global struct RegressionList* aDData,
                            __global double* aWeights,
                            __global unsigned long* aWeightSize,
                            __global struct CurveData* aDst,
                            __global double* aParams){

    //Getting the thread ID
    unsigned int i = get_global_id(0);

    //Initializing aDst
    aDst[i].vAMax = aHData[i].vAMax;
    aDst[i].vAMin = aHData[i].vAMin;

    aDst[i].vHValue = aHData[i].vSlope[10];
    aDst[i].vDValue = aDData[i].vSlope[10];
    aDst[i].vR2Value = aHData[i].vRSquared[10];

    //Finding the delta of both H and D.
    get_delta(aHData[i].vSlope,aHData[i].vDeltaSlope,*aWeightSize);
    get_delta(aHData[i].vIntercept,aHData[i].vDeltaIntercept,*aWeightSize);
    get_delta(aHData[i].vRSquared,aHData[i].vDeltaRSquared,*aWeightSize);

    get_delta(aDData[i].vSlope,aDData[i].vDeltaSlope,*aWeightSize);
    get_delta(aDData[i].vIntercept,aDData[i].vDeltaIntercept,*aWeightSize);
    get_delta(aDData[i].vRSquared,aDData[i].vDeltaRSquared,*aWeightSize);


    //Computing the Weighted Means
    aDst[i].vWeightedDelta = weighted_mean(aHData[i].vDeltaSlope,aWeights,*aWeightSize);
    aDst[i].vWeightedH = weighted_mean(aHData[i].vSlope,aWeights,*aWeightSize);
    aDst[i].vWeightedR2 = weighted_mean(aHData[i].vRSquared,aWeights,*aWeightSize);

    aDst[i].vAvgDelta = mean(aHData[i].vDeltaSlope,*aWeightSize);
    aDst[i].vAvgH = mean(aHData[i].vSlope,*aWeightSize);
    aDst[i].vAvgR2 = mean(aHData[i].vRSquared,*aWeightSize);

    aDst[i].vWeightedD = weighted_mean(aDData[i].vSlope,aWeights,*aWeightSize);
    aDst[i].vAvgD = mean(aDData[i].vSlope,*aWeightSize);

    //Computing the weighted Standard deviations
    aDst[i].vStdevWeightedDelta = weighted_stdev(aHData[i].vDeltaSlope,aWeights,*aWeightSize);
    aDst[i].vStdevWeightedH = weighted_stdev(aHData[i].vSlope,aWeights,*aWeightSize);
    aDst[i].vStdevWeightedR2 = weighted_stdev(aHData[i].vRSquared,aWeights,*aWeightSize);

    aDst[i].vStdevDelta = stdev(aHData[i].vDeltaSlope,*aWeightSize);
    aDst[i].vStdevH = stdev(aHData[i].vSlope,*aWeightSize);
    aDst[i].vStdevR2 = stdev(aHData[i].vRSquared,*aWeightSize);

    aDst[i].vStdevWeightedD = weighted_stdev(aDData[i].vSlope,aWeights,*aWeightSize);
    aDst[i].vStdevD = stdev(aDData[i].vSlope,*aWeightSize);

    //Classifying the curve
    classify_curvedata(aDst+i,aParams[0],aParams[1],aParams[2],aParams[3],aParams[4],aParams[5]);

}



//==============================================================================
// This section of code us used to compute the regression lines
// It containes the OpenCL kernel for computing the regression lines
//
//==============================================================================

/**
 * Calculates the line of best fit through a set of ordered pairs
 * {(aData[aFirst].x, aData[aFirst].y),(aData[aFirst+1].x, aData[aFirst+1].y)
 *     ,...,
 *   (aData[aLast].x, aData[aLast].y)} using the least squares method.
 * This function will modify aDst
 *
 * Sxx = âĹŚ(x^2|i=1 to n) - n (x-bar)^2
 * <br>
 * Syy = âĹŚ(y^2|i=1 to n) - n (y-bar)^2
 * <br>
 * B-hat = âĹŚ(xy|i=1 to n) - (n x-bar y-bar)/Sxx
 * <br>
 * A-hat = y-bar - B-hat*x-bar
 * <br>
 * R^2 = (B-hat) âĹŽ(Sxx/Syy)
 *
 * @param aDst - The container for holding the results of the regression
 * @param aData - The set of ordered pairs to find the linear model of
 * @param aDataSize - The number of element
 * @param aFirst - The first element of the regression
 * @param aLast - The last element of the regression.
 */
__kernel void calculate_least_squares_regression(__global struct LinearModel* aDst,
                                                 __global struct PointF* aData,
                                                 unsigned long aDataSize,
                                                 unsigned long aFirst,
                                                 unsigned long aLast){
```

```

    //Initialize the return argument
    aDst->vSumXY = 0;
    aDst->vSumXSquared = 0;
    aDst->vSumYSquared = 0;
    aDst->vBetaHat = 0;
    aDst->vAlphaHat = 0;
    aDst->vSxx = 0;
    aDst->vSyy = 0;
    aDst->vRSquared = 0;
    aDst->vRVal = 0;


    //Performing the linear model calculations

  unsigned long lSize = aLast-aFirst+1;
    aDst->vMeanXY = calc_average_pt(aData,aFirst, aLast);

    for(unsigned long i = aFirst; i < aLast+1; i++){
        aDst->vSumXY = aDst->vSumXY + ( aData[i].x * aData[i].y );
        aDst->vSumXSquared = aDst->vSumXSquared + ( aData[i].x * aData[i].x );
        aDst->vSumYSquared = aDst->vSumYSquared + ( aData[i].y * aData[i].y );

    }


    aDst->vBetaHat = (aDst->vSumXY - (lSize*aDst->vMeanXY.x*aDst->vMeanXY.y))
                    /(aDst->vSumXSquared - (lSize*aDst->vMeanXY.x*aDst->vMeanXY.x));
    aDst->vAlphaHat = aDst->vMeanXY.y - (aDst->vBetaHat*aDst->vMeanXY.x);
    aDst->vSxx = aDst->vSumXSquared - (lSize*aDst->vMeanXY.x*aDst->vMeanXY.x);
    aDst->vSyy = aDst->vSumYSquared - (lSize*aDst->vMeanXY.y*aDst->vMeanXY.y);
    aDst->vRVal = aDst->vBetaHat*sqrt(aDst->vSxx/aDst->vSyy);
    aDst->vRSquared = aDst->vRVal*aDst->vRVal;


    aDst->vMinX = aData[aFirst].x;
    aDst->vMaxX = aData[aLast].x;

}

/***
 * This function computes the linear model and regression list for the
 * specified element.
 * Both aRL and aLM are modified during this calculation.
 *
 * @param aRL - The set of regression lines
 * @param aLM - The linear model
 * @param aBounds - The set of subsets to compute the regression lines over
 * @param aPairSize - The number of subsets in aBounds
 * @param aData - The set of sets of ordered pairs to compute the regression lines for
 * @param aDataSize - The number of sets of contained in aData
 */
__kernel void calc_curvedata_rl(__global struct RegressionList* aRL,
                                __global struct LinearModel* aLM,
                                __global struct SizeTPair* aBounds,
                                __global unsigned long* aPairSize,
                                __global struct PointFList* aData,
                                __global unsigned long* aDataSize){

    //Find the current row and column to compute the regression list of
    unsigned long i = get_global_id(0);
    unsigned long j = i/(*aDataSize);
    unsigned long k = i%(*aDataSize);

    //Compute the regression of the line between the range specified in aBounds
    calculate_least_squares_regression(aLM+i,
                                       aData[k].vData,
                                       aData[k].vDataSize,
                                       aBounds[j].first ,
                                       aBounds[j].second);

    //Set the values in the return variable
    aRL[j].vSlope[k] = aLM[i].vBetaHat;
    aRL[j].vIntercept[k] = aLM[i].vAlphaHat;
    aRL[j].vRSquared[k] = aLM[i].vRSquared;

    if(k==0){
        aRL[j].vAMin = aData[k].vData[aBounds[j].first].x;
        aRL[j].vAMax = aData[k].vData[aBounds[j].second].x;
    }


}
```

BIOGRAPHY OF THE AUTHOR

Brian Toner was born in Las Vegas, Nevada and moved to Maine when he was four years old. He completed his high school in Dover-Foxcroft, Maine at Foxcroft Academy in July of 2002. He earned his B.A. in Mathematics from the University of Maine in 2013 and his M.A. in Mathematics from the University of Maine in 2015. As a graduate student, Brian served as a Teaching Assistant in the Department of Computer Science and served as a Graduate Research Assistant and programmer in the CompuMAINE laboratory. Before coming to the University, Brian spent six years in the military, after which he decided it was time to move on and pursue traveling and education. Brian traveled the United States for two years before taking up studies at the University of Maine. Though travel made up much of his activities as a youth, he enjoys a slower life in a more centralized area. He spends his free time in Orono writing software, books, blogs and silly songs. Brian Christopher Toner is a candidate for the Doctor of Philosophy degree in Computer Science from the University of Maine in December 2019.