

# Support Vector Machine and Parametric Wavelet-Based Texture Classification of Stem Cell Images

by

Christopher G. Jeffreys

B.S. Mathematics

United States Air Force Academy, 2002

SUBMITTED TO THE SLOAN SCHOOL OF MANAGEMENT  
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF  
MASTER OF SCIENCE IN OPERATIONS RESEARCH  
AT THE  
MASSACHUSETTS INSTITUTE OF TECHNOLOGY  
JUNE 2004

© Christopher G. Jeffreys, 2004. All rights reserved.

The author hereby grants to MIT permission to reproduce and distribute publicly paper  
and electronic copies of this thesis document in whole or in part.

Author: Christopher G. Jeffreys

Sloan School of Management  
Interdepartmental Program in Operations Research  
May 14, 2004

Certified by: Roy E. Welsch

Roy E. Welsch  
Professor of Statistics and Management Science  
Thesis Supervisor

Certified by: Rami S. Mangoubi

Rami S. Mangoubi  
Senior Member of the Technical Staff, C. S. Draper Laboratory  
Thesis Supervisor

Certified by: Mukund N. Desai

Mukund N. Desai  
Distinguished Member of the Technical Staff, C. S. Draper Laboratory  
Thesis Supervisor

Accepted by: \_\_\_\_\_

James B. Orlin  
Edward Pennell Brooks Professor of Operations Research  
Codirector, Operations Research Center

[This page intentionally left blank.]

# Support Vector Machine and Parametric Wavelet-Based Texture Classification of Stem Cell Images

by  
Christopher G. Jeffreys

Submitted to the Sloan School of Management  
on May 14, 2004, in partial fulfillment of the  
requirements for the degree of  
Master of Science in Operations Research

## Abstract

Stem cell research is one of the most promising and cutting-edge fields in the medical sciences. It is believed that this innovative research will lead to life-saving treatments in the coming years. As part of their work, stem cell researchers must first determine which of their stem cell colonies are of sufficiently high quality to be suitable for experimental studies and therapeutic treatments. Since colony texture is a major discriminating feature in determining quality, we introduce a non-invasive, semi-automated texture-based stem cell colony classification methodology to aid researchers in colony quality control.

We first consider the general problem of textural image segmentation. In a new approach to this problem, we characterize image texture by the subband energies of the image's wavelet decomposition, and we employ a non-parametric support vector machine to perform the classification that yields the segmentation. We also adapt a parametric wavelet-based classifier that utilizes the Kullback-Leibler distance. We apply both methods to a set of benchmark textural images, report low segmentation error rates and comment on the applicability of and tradeoffs between the non-parametric and parametric segmentation methods. We then apply the two classifiers to the segmentation of stem cell colony images into regions of varying quality. This provides stem cell researchers with a rich set of descriptive graphical representations of their colonies to aid in quality control. From these graphical representations, we extract colony-wise textural features to which we add colony-wise border features. Taken together, these features characterize overall colony quality. Using these features as inputs to a multiclass support vector machine, we successfully categorize full stem cell colonies into several quality categories. This methodology provides stem cell researchers with a novel, non-invasive quantitative quality control tool.

Thesis Supervisor: Roy E. Welsch  
Title: Professor of Statistics and Management Science

Thesis Supervisor: Rami S. Mangoubi  
Title: Senior Member of the Technical Staff, C. S. Draper Laboratory

Thesis Supervisor: Mukund N. Desai  
Title: Distinguished Member of the Technical Staff, C. S. Draper Laboratory

[This page intentionally left blank.]

## Acknowledgments

First, I would like to thank Rami Mangoubi of the Draper Laboratory for his ceaseless involvement in the research that led to the development of this thesis. His care for and diligence toward research are unmatched. I should also acknowledge Mukund Desai of the Draper Laboratory for sharing his deep insights into the problem.

Additionally, I thank Professor Roy Welsch of MIT for taking on the role of faculty advisor and for his helpful comments and suggestions in the development and writing of this thesis.

The stem cell application of this thesis would not have been possible without the efforts of Prof Paul Sammak of the Pittsburgh Development Center of the Magee-Womens Research Institute at the University of Pittsburgh. I am grateful for the time he gave from his already busy schedule to gather all of the stem cell images used in this work and to provide the medical insight needed to tackle the problem.

I am appreciative of the United States Air Force for allowing me the opportunity to spend two years studying at such an esteemed institution as MIT among the marvellous professors, students and administrators of the Operations Research Center. My thanks also extend to the Charles Stark Draper Laboratory for its commitment to graduate education and for providing the means, both physically and monetarily, for my studies.

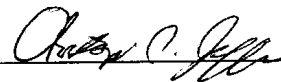
To my fellow students at Draper and MIT, thanks for the many academic and non-academic discussions and experiences. I wish you all the best.

Finally, for emotional and spiritual support, thanks go to my family and my Lord Jesus Christ.

This thesis was prepared at The Charles Stark Draper Laboratory, Inc., under Internal Company Sponsored Research Project 13177, GC DLF Support.

Publication of this thesis does not constitute approval by Draper or the sponsoring agency of the findings or conclusions contained herein. It is published for the exchange and stimulations of ideas.

As a member of the Air Force, I acknowledge that the views expressed in this thesis are mine and do not reflect the official policy or position of the United States Air Force, the Department of Defense or the United States Government.



---

Christopher G. Jeffrey

[This page intentionally left blank.]

# Contents

<b>1</b>	<b>Introduction</b>	<b>13</b>
1.1	Medical Applications of Texture Classification . . . . .	14
1.2	Thesis Overview . . . . .	15
1.3	Thesis Contributions . . . . .	17
<b>2</b>	<b>Machine Learning for Texture Analysis</b>	<b>19</b>
2.1	The Machine Learning Classification Paradigm . . . . .	19
2.1.1	The “Black Box” Illustration . . . . .	20
2.1.2	Feature Representation . . . . .	21
2.1.3	Classification . . . . .	21
2.2	Texture Feature Representation . . . . .	23
2.2.1	Image Processing Terminology . . . . .	23
2.2.2	Defining Texture . . . . .	24
2.2.3	Common Feature Paradigms and Feature Sets . . . . .	26
2.2.4	Wavelet Energy Features . . . . .	30
2.3	Non-Parametric Classification: The Support Vector Machine . . . . .	34
2.3.1	Common Non-Parametric Classification Methods . . . . .	34
2.3.2	The Support Vector Machine . . . . .	35
2.4	Parametric Classification: The Kullback-Leibler Distance Classifier . . . . .	40
2.4.1	The Generalized Gaussian Distribution for Wavelet Coefficient Modelling . . . . .	41
2.4.2	The Kullback-Leibler Distance . . . . .	42
<b>3</b>	<b>Benchmark Analysis</b>	<b>45</b>
3.1	Problem Description . . . . .	45
3.1.1	Benchmark Textural Images . . . . .	46
3.1.2	Image Preprocessing . . . . .	46
3.1.3	Training and Test Images . . . . .	48
3.2	Training and Test Algorithms . . . . .	49
3.2.1	Support Vector Machine Method . . . . .	49
3.2.2	Kullback-Leibler Distance Method . . . . .	50
3.2.3	Parameter Tuning . . . . .	52
3.3	Benchmark Results . . . . .	55
3.3.1	Basic Results . . . . .	55
3.3.2	Training Set Size Results . . . . .	59

3.3.3	Conclusions . . . . .	60
<b>4</b>	<b>Stem Cell Application: Qualitative Interior Textural Segmentation</b>	<b>61</b>
4.1	Medical and Biological Motivation . . . . .	61
4.1.1	Stem Cell Background . . . . .	61
4.1.2	Research Contributions . . . . .	63
4.1.3	Stem Cell Visual Quality Criteria . . . . .	63
4.1.4	Research Scope: A Two-Tiered Approach . . . . .	64
4.2	Interior Textural Segmentation Methodology . . . . .	66
4.2.1	Description of Images . . . . .	66
4.2.2	Image Preprocessing . . . . .	66
4.2.3	Training Data Extraction . . . . .	67
4.2.4	SVM and KLD Algorithmic Settings . . . . .	69
4.3	Interior Textural Segmentation Results . . . . .	70
4.3.1	Runs Performed . . . . .	70
4.3.2	Reclassified Image Outputs . . . . .	70
4.3.3	Reclassified Image Comparisons . . . . .	74
<b>5</b>	<b>Stem Cell Application: Quantitative Image Categorization</b>	<b>83</b>
5.1	Colony-Wise Feature Extraction . . . . .	84
5.1.1	Interior Texture Feature Extraction . . . . .	85
5.1.2	Border Feature Extraction . . . . .	88
5.2	Expert Rating of Colony Quality . . . . .	91
5.3	Colony-Wise Classification Methodology . . . . .	92
5.3.1	Multiclass SVM Formulation . . . . .	92
5.3.2	Run Setup . . . . .	93
5.4	Quantitative Image Categorization Results . . . . .	98
5.4.1	Mean Optimality Results . . . . .	99
5.4.2	Individual Optimality Results . . . . .	99
5.4.3	On Separating Categories 1 and 2 . . . . .	100
5.4.4	Conclusions and Implementation . . . . .	101
<b>6</b>	<b>Conclusions and Future Work</b>	<b>103</b>
6.1	Future Technical Challenges . . . . .	104
6.2	Future Applications . . . . .	106
<b>A</b>	<b>The Support Vector Machine</b>	<b>107</b>
A.1	Binary Classification Formulations . . . . .	107
A.1.1	Linear Primal Formulation . . . . .	108
A.1.2	Linear Dual Formulation . . . . .	109
A.1.3	Soft Margin Primal and Dual Formulations . . . . .	111
A.1.4	Nonlinear Soft Margin Dual Formulation . . . . .	111
A.2	Multiclass Formulations . . . . .	113
A.2.1	Error Correcting Output Codes . . . . .	113
A.2.2	One-vs-All Formulation . . . . .	114



# List of Figures

1-1	Hierarchical Two Tiered Stem Cell Analysis Paradigm . . . . .	16
2-1	The Basic “Black Box” . . . . .	20
2-2	Partitioned “Black Box” . . . . .	20
2-3	Sliding Neighborhood Feature Extraction . . . . .	25
2-4	Illustration of Textural Primitives . . . . .	28
2-5	Signal Processing Feature Representation . . . . .	29
2-6	Single Resolution Analysis vs. Multi-Resolution Analysis . . . . .	30
2-7	$n$ -level Wavelet Pyramid . . . . .	31
2-8	Two-Dimensional DWT Schematic . . . . .	31
2-9	3-level Wavelet Pyramid . . . . .	32
2-10	Neural Network Architecture . . . . .	35
2-11	Separating Hyperplanes in $\mathbb{R}^2$ . . . . .	37
2-12	Generalized Gaussian Distribution . . . . .	42
3-1	Example Two-Texture Composite Test Image . . . . .	46
3-2	Example Training Images . . . . .	46
3-3	18 Two-Texture Composite Test Images . . . . .	49
3-4	Support Vector Machine Training Algorithm . . . . .	50
3-5	Support Vector Machine Test Algorithm . . . . .	51
3-6	Kullback-Leibler Distance Training Algorithm . . . . .	52
3-7	Kullback-Leibler Distance Test Algorithm . . . . .	53
3-8	Brodatz Validation Image Pairs . . . . .	53
3-9	Training Set Size Effects . . . . .	59
4-1	Typical Stem Cell Colony . . . . .	62
4-2	Good and Bad Stem Cell Colonies . . . . .	64
4-3	Two Tiered Stem Cell Analysis Methodology . . . . .	65
4-4	Tier 1 Image Preprocessing . . . . .	67
4-5	Two Types of Good Stem Cell Texture . . . . .	67
4-6	Two Types of Bad Stem Cell Texture . . . . .	68
4-7	Tier 1 Graphical Outputs Example Colony . . . . .	71
4-8	Binary Reclassified Images . . . . .	72
4-9	Locally Normalized Confidence Reclassified Images . . . . .	72
4-10	Globally Normalized Confidence Reclassified Images . . . . .	73
4-11	Quality Movie . . . . .	74

4-12	Stem Cell Colonies Used for Reclassified Image Comparison Study . . . . .	75
4-13	Colony 1 Classification Method Effects . . . . .	76
4-14	Colony 2 Classification Method Effects . . . . .	76
4-15	Colony 3 Classification Method Effects . . . . .	77
4-16	Colony 4 Classification Method Effects . . . . .	77
4-17	Colony 1 Training Set Effects . . . . .	79
4-18	Colony 2 Training Set Effects . . . . .	79
4-19	Colony 3 Training Set Effects . . . . .	80
4-20	Colony 4 Training Set Effects . . . . .	80
4-21	Colony 1 Window Size Effects . . . . .	81
4-22	Colony 2 Window Size Effects . . . . .	81
4-23	Colony 3 Window Size Effects . . . . .	81
4-24	Colony 4 Window Size Effects . . . . .	82
5-1	Two Tiered Stem Cell Analysis Methodology (Review) . . . . .	84
5-2	Confidence Reclassified Images of Two Toy Colonies . . . . .	86
5-3	Border Window Extraction Process . . . . .	89
5-4	Border Window Orientation . . . . .	90
5-5	Tier 2 Colony-Wise Features . . . . .	94
5-6	General Tier 2 Performance Matrices . . . . .	97
5-7	Example Mean Optimal Performance Matrix . . . . .	97
5-8	Example Individual Optimal Performance Matrix . . . . .	98
5-9	Four-Way Mean Optimal Performance Matrix . . . . .	99
5-10	Four-Way Mean Optimal Border Effects Performance Matrix . . . . .	99
5-11	Four-Way Individual Optimal Performance Matrix . . . . .	100
5-12	Four-Way Individual Optimal Border Effects Performance Matrix . . . . .	100
5-13	Five-Way Mean Optimal Performance Matrix . . . . .	101
5-14	Five-Way Individual Optimal Performance Matrix . . . . .	101

# List of Tables

2.1	Haralick's Statistical Texture Features . . . . .	27
3.1	25 Benchmark VisTex Texture Images . . . . .	47
3.2	18 Benchmark Textural Image Pairs . . . . .	48
3.3	D4/D84 Validation Image Pair Results, Polynomial Kernel (degree 3) . . .	54
3.4	D5/D92 Validation Image Pair Results, Polynomial Kernel (degree 3) . . .	55
3.5	Basic Benchmark Results (100 training points per class) . . . . .	56
3.6	Basic Benchmark Results (50 training points per class) . . . . .	57
4.1	Nine Tier 1 Training Sets . . . . .	69
4.2	Nine Tier 1 Training Sets (Review) . . . . .	78
5.1	Tier 2 Colony-Wise Interior Texture Features . . . . .	88
5.2	Tier 2 Colony-Wise Border Features . . . . .	91
5.3	Full Colony Rating Scale . . . . .	91
5.4	Colony-Wise Interior Texture Feature Groups . . . . .	95
5.5	Colony-Wise Border Feature Groups . . . . .	95

[This page intentionally left blank.]

# Chapter 1

## Introduction

Stem cell research is one of the most promising and cutting-edge fields in the medical sciences. From the understanding of the core mechanisms of human growth and sustainment to the development of novel medical treatments, the potential benefits of this research are far reaching.

The hallmark of the stem cell that drives the current research frenzy is its unique ability to develop from a generic, undifferentiated state into a wide range of specialized human cells. For example, a single, unspecialized stem cell may specialize to become a blood cell, bone cell, brain cell or many of the numerous other human cells necessary to sustain life. While stem cell research is relatively young—beginning in 1998 with the first sustained culture of human embryonic stem cells—understanding of the biology of these cells and the medical treatments they can potentially provide has steadily progressed.

For example, by studying the process by which stem cells specialize, scientists can systematically and safely learn the specific genetic and molecular triggers that cause unspecialized cells in humans to differentiate into various types of specialized cells. Since abnormalities in this process of specialization and cell growth contribute to many severe diseases such as cancer and birth defects, an understanding of the molecular and genetic causes will help researchers treat or prevent these diseases in humans [33].

Additionally, the medical treatments that could potentially arise from stem cell research are numerous and powerful. One of the most ambitious treatments is transplantation therapy. Presently, only organs and tissues transplanted directly from other living or recently deceased human beings are used in this treatment, and the demand for human transplants exceeds the supply. With stem cell transplantation therapy, stem cells would be induced to specialize into the needed replacement tissues or organs providing a limitless supply of healthy transplants. Furthermore, stem cells could be used to treat conditions that even human transplantation cannot. Individual or groups of cells needed to cure such conditions as strokes, burns, Parkinson's, Alzheimer's and heart disease could be easily generated from stem cells [33].

Despite the many avenues stem cell research is taking today, medical researchers in the field all have the common need for high quality, undifferentiated lines of stem cell colonies on which to perform their research. While stem cell colonies can be readily cultured in the lab, researchers need to quickly and reliably determine which of the cultured colonies are sufficiently healthy for use in their experiments and therapies. The current state

of the art for this quality control process involves either direct visual inspection by an expert stem cell microscopist or the performance of invasive, destructive biological tests. The visual approach quickly becomes impractical considering the large number of colonies needed for research and therapeutic purposes. In the biological test approach, the resulting destruction of the inspected colonies is a significant drawback. Thus, this thesis is focused toward providing novel semi-automated, non-invasive, image-based computational tools to aid the medical researcher in the task of stem cell colony quality control.

However, that is not the only goal of this thesis. In fact, the method that we use and expand upon to approach the stem cell application is a notably more general problem, namely that of *textural image segmentation*. This is the problem of dividing, or segmenting, an image into various regions based on the texture of those regions. This general problem relates well to the stem cell application since, as we discuss in more detail later, stem cell colony texture is one of the most influential factors in distinguishing between areas of high and low quality within and among stem cell colonies. Thus, by studying solution methods to the general problem of textural image segmentation, we provide a strong basis for both our approach to the stem cell problem and to various other uses of the methods.

## 1.1 Medical Applications of Texture Classification

Throughout this thesis, as we encounter new topics and problems, we provide overviews of and references to the various solution approaches taken by past researchers. Thus, in this section, we do not intend to provide a thorough review of past work in the many fields on which we touch; we leave that effort to the thesis body itself. Instead, as a motivation for the usefulness and necessity of our work, we provide a few examples of the use of texture classification methods to various medical applications.

Surprisingly, when we consider what has previously been accomplished in creating non-invasive, image-based computational tools for stem cell colony analysis, we do not find an established body of literature. However, the use of texture-based analysis can be found in other medical fields, most notable toward cancer research.

For example, Rubegni *et. al.* use textural features to distinguish between harmless skin lesions and cancerous melanomas. In fact, of all the various features considered in their study, the textural contrast feature shows the most discriminative power [43]. Handels *et. al.* also consider the problem of melanoma classification using texture features. They propose the use of co-occurrence, Fourier and fractal textural features with a neural network classifier to achieve a classification performance of 97.7% [16]. Gerger *et. al.* also use co-occurrence texture features, along with color and gray level features, both to segment skin lesions from the surrounding skin area and to diagnose the segmented regions as benign or malignant. They employ the CART and nearest neighbor classifiers for these tasks [14].

The similarities between melanoma and stem cell colony classification do not end with texture analysis. As mentioned above, in Chapter 5 we consider measures of stem cell colony border quality, and we find similar work among melanoma researchers. For example, Grana *et. al.* propose a novel method of quantifying border sharpness in melanoma

images using gradient measures [15]. Lee outlines various methods for the analysis of border irregularity, proposes a new approach and applies his approach to the task of diagnosing cancerous melanomas [28]. While we do not explicitly use these approaches, they suggest the applicability of previous research to the problem of stem cell colony segmentation and categorization.

In addition to these applications to melanoma detection, textural segmentation has been used as a step in the process of classifying colon cancer images [38] and in the successful discrimination between malignant and benign breast cancer images [48].

Overall, we feel that the wide-ranging usefulness of texture-based image analysis by researchers in various medical fields suggests that an equally fruitful landscape awaits in the realm of stem cell research.

## 1.2 Thesis Overview

To provide a brief glimpse of our specific approaches to both the general problem of textural image segmentation and to the stem cell application, we summarize here the content of each chapter of this thesis, outlining the methods we employ, adapt or introduce in each.

### Chapter 2: Machine Learning for Texture Analysis

Chapter 2 serves as a technical literature review of the various approaches to the problem of textural image segmentation. We also use it to present in detail the specific methods we choose to employ in this thesis. We pose the textural image segmentation problem in the context of the machine learning classification paradigm and its two steps of feature representation and classification. We next discuss various common methods of representing the textural characteristics of images, focussing on the specific method that we employ, the wavelet energy feature. We then turn to the classification problem by outlining a few of the more popular non-parametric methods as motivation for the discussion of the support vector machine classifier that we find superior. We motivate the development of the support vector machine by appealing to its theoretical underpinnings before describing its formulation as an optimization problem and discussing its unique features. Finally, we detail the parametric wavelet-based Kullback-Leibler distance classifier of Vetterli and Do that we adapt for the task of textural image segmentation.

### Chapter 3: Benchmark Method Analysis

In Chapter 3 we apply the methods developed in Chapter 2 to the task of textural segmentation of benchmark images. We formalize the training and test algorithms of both the non-parametric support vector machine and parametric Kullback-Leibler distance method, apply the algorithms to 18 pairs of benchmark images and report the performance of both methods in segmenting the 18 pairs into the two textural fields comprising each. The excellent performance of both methods on the benchmark images gives us confidence in applying the methodology to the stem cell application. Finally, we comment

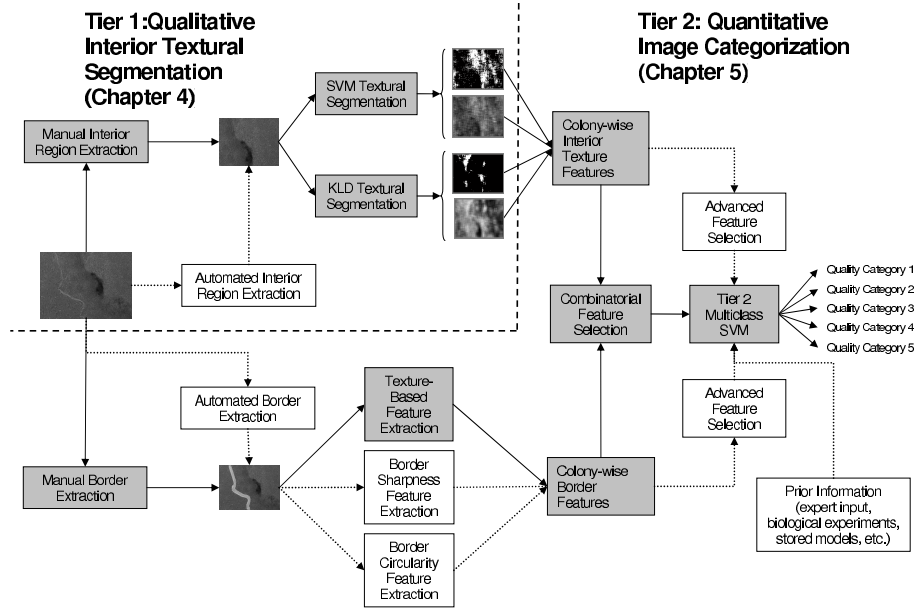


Figure 1-1: Hierarchical Two Tiered Stem Cell Analysis Paradigm

on the applicability and relative merits of non-parametric and parametric classifiers for the task of textural image segmentation.

#### Chapter 4: Stem Cell Application: Qualitative Interior Textural Segmentation

In Chapter 4, we motivate and introduce the stem cell application that is the major focus of this thesis. We discuss the medical aspects of the problem and find that the textural characteristics of stem cell colonies have a strong influence on their overall quality and suitability for research. This suggests that the application of our textural image segmentation methods of Chapters 2 and 3 would constitute a contribution to the field of stem cell research. We frame this contribution as a two-tiered, hierarchical methodology that creates both qualitative and quantitative measures of stem cell quality. Figure 1-1 diagrams this hierarchical paradigm showing work accomplished in this thesis with shaded blocks and solid lines. Future work is shown as unshaded blocks with dotted lines. We consider the Tier 1 analysis in Chapter 4 and the Tier 2 work in Chapter 5.

The Tier 1 analysis creates qualitative graphical representations of the interior regions of stem cell colonies to aid researchers in pinpointing areas of good and bad quality in their colonies. This is basically an application of the procedures of Chapter 3 applied to real images of stem cell colonies. We propose practical uses for each of the graphical outputs and discuss the effects of the various parameter settings that need to be specified in the process.

#### Chapter 5: Stem Cell Application: Quantitative Image Categorization

The Tier 2 work in Chapter 5 expands on the Tier 1 analysis by extracting colony-wise features from the Tier 1 outputs and introducing additional colony-wise features based on the quality of the border of each colony. It then uses a multiclass support vector machine



to categorize each full stem cell colony into one of several quality categories based on the colony-wise features. We report effective categorization rates using these colony-wise features.

## Chapter 6: Summary, Conclusions and Future Work

The purpose of Chapter 6 is to draw the thesis to a close by making unifying conclusions regarding the methods we have introduced for the problem of textural image segmentation and the application and extension of those methods to create analysis tools for the stem cell research community. We also suggest avenues for future work and applications of the methodology that we have considered.

### 1.3 Thesis Contributions

Despite the abundance of work already accomplished in the general field of textural image segmentation, this thesis provides two new contributions.

- We demonstrate the applicability of using the non-parametric support vector machine classifier with wavelet energy features to the textural segmentation of images. We report a high success rate of 92.3% over a diverse set of benchmark textural images.
- We adapt a parametric wavelet-based texture classification method originally proposed for the task of content-based image retrieval to the task of textural image segmentation. We report a high classification rate of 91.8% over a diverse set of benchmark textural images.

Additionally, we make three novel advancements to the field of stem cell research.

- We introduce a two-tiered, hierarchical paradigm for the creation of a standard, automated, non-invasive stem cell quality control tool. Our implementation of this paradigm represents, to the best of our knowledge, the first known attempt at creating a routine, non-invasive method for measuring stem cell quality under conditions favorable to their growth.
- As the first tier of this new paradigm, we perform textural image segmentation on stem cell colony images to create a rich set of graphical aids to assist medical researchers in determining the quality of their colonies.
- Employing inputs derived from the first tier, we introduce in the second tier a highly accurate method for making quantitative categorizations of stem cell colonies based on overall colony quality. Across a wide range of tests on real stem cell images, we attain an average performance of 80.0%.

[This page intentionally left blank.]

## Chapter 2

# Machine Learning for Texture Analysis

As described in Chapter 1, one of the main thrusts of this thesis is to introduce both qualitative graphical and quantitative numerical tools to aid stem cell researchers in the important task of characterizing the quality and health of their stem cell colonies. Based on the various biological properties of stem cells that we discuss in Chapter 4, we find that we can accurately pose this problem as one of *textural image segmentation*. Thus, before focusing exclusively on the stem cell application, we should consider the theoretical foundations and technical details of this textural image segmentation problem. That is the purpose of this chapter.

Since the problem of textural image segmentation is actually a subset of the larger and more general field of *machine learning classification*, we begin in Section 2.1 by describing in generic terms the major aspects and terminology of this broad field. We will see in this discussion that we can partition the problem into the two steps of *feature representation* and *classification*. When we then turn our focus to textural image segmentation, we use these two steps to guide our presentation.

Specifically, we delve into the details of textural feature representation in Section 2.2. We motivate and describe the wavelet-based features that we employ in this thesis. Sections 2.3 and 2.4 then focus on the classification problem by outlining the non-parametric support vector machine and parametric Kullback-Leibler distance classifiers that we employ and whose performance we compare in Chapter 3.

## 2.1 The Machine Learning Classification Paradigm

In the simplest terms, the field of machine learning attempts to figure out (learn) the processes of nature through methods coded into and implemented by a computer (machine)—hence the intuitive name “machine learning.” However, this definition lacks a certain concreteness. Thus, it is our goal in this section to formalize the machine learning problem. While machine learning techniques are well-suited to a large class of data *description* and *prediction* problems, we focus specifically on the problem of *classification*. We find a useful context for the classification problem by posing it in the “Black Box” manner of



Figure 2-1: The Basic “Black Box”

Section 2.1.1. We follow with descriptions of the two vital parts of the machine learning classification problem: feature representation and classification method.

### 2.1.1 The “Black Box” Illustration

Figure 2-1 shows the basic black box description of machine learning classification. An *input object* enters the box and is transformed into some output that subsequently exits the box. The input object can be anything that needs to be classified, such as an image, an email or a sonar reading. We interchangeably refer to the input object as the *input point* or simply *input*. If we restrict the output to two possible values,  $+1$  and  $-1$ , representing the *class*, or *label*, to which the input is to be classified, we arrive at the *binary classification problem*<sup>1</sup>.

Given a collection of input objects whose associated output classifications are fully known a priori, it is our goal to learn the internal mechanism of the black box that yields proper classifications. Once this mechanism is learned, we can predict the classification of future inputs for which we do not know the correct class a priori.

The process of learning the mechanism of the black box can be broken into the two steps of *feature extraction* and *classification* as in Figure 2-2. The feature extraction step involves representing the input object in a descriptive manner that is both recognizable by a computer system (i.e., as a numerical vector) and able to capture the key aspects of the object that will lead to an accurate classification. The classification step refers to the process of actually classifying the input points—via their feature representations—into the two classes.

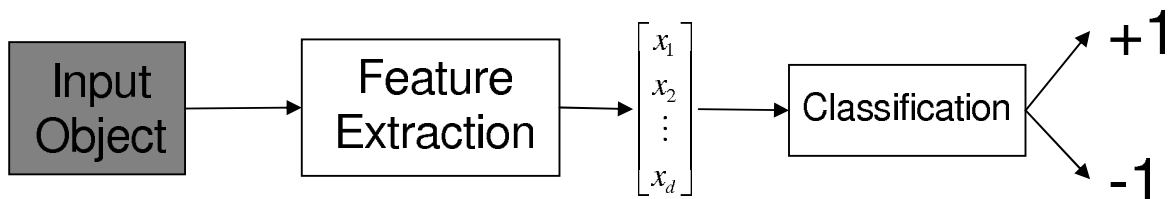


Figure 2-2: Partitioned “Black Box”

This partition into the two separate steps of feature representation and classification is the standard procedure in machine learning applications. Such an architecture allows feature representations and classification methods to be designed independently by experts in the individual fields. It also allows for the analysis of numerous different pairings

<sup>1</sup>We discuss the extension of the binary problem to the multiclass case in Appendix A

of feature representations and classification methods. There have also been attempts to unify the two steps by designing classifiers that are tailored to specific feature representations. We describe one such unifying method, the Kullback-Leibler distance classifier, in Section 2.4. However, we begin by considering the two steps in isolation.

We complete this section with an overview of some general terminology and theory of the feature representation and classification steps. While this treatment of machine learning classification theory is in no way complete, it will outline the key ideas and define the terms that will be used throughout the remainder of the thesis. More detailed work on machine learning theory and application can be found in the machine learning and statistical learning theory literature [19] [20].

## 2.1.2 Feature Representation

In contrast to the generic effort of designing classification methods, the feature representation task is driven by the specific application under consideration. Thus, it is difficult to discuss feature representation in a general manner. In fact, we have only one piece of standard terminology to introduce in this section. We let  $\mathbf{x}_i \in \mathbb{R}^d$  represent the  $d$ -dimensional real-valued *feature vector* that represents the  $i$ th object we are attempting to classify. For our stem cell analysis, this object is the texture of the stem cell images. However, we defer the more specific descriptions of textural feature representations to Section 2.2.

## 2.1.3 Classification

As mentioned above, we can speak generally of the classification task without regard to the specific application. We simply assume that we have been provided a set of  $d$ -dimensional feature vectors from the application-specific feature representation step; additionally, each feature vector has associated with it some known or unknown *label*,  $y \in \{+1, -1\}$ , representing the class into which the object falls.

Binary classification methods attempt to separate two classes of input points drawn from some probability distributions,  $p(\mathbf{x}|y = +1)$  and  $p(\mathbf{x}|y = -1)$ . Given a *training set* of  $N$  input points,  $(\mathbf{x}_i, y_i)$  for  $i = 1, \dots, N$ , whose true classifications ( $y_i$  values) are fully known, we attempt to design a function that both classifies correctly on that training set—called *empirical* performance—and on a separate *test set* not used in training the classification function—called the *generalization* performance<sup>2</sup>. If the probability distributions  $p(\mathbf{x}|y = +1)$  and  $p(\mathbf{x}|y = -1)$  happen to be known explicitly or can be accurately estimated, a *parametric model* is preferable in order to exploit this knowledge. For example, if we know that  $p(\mathbf{x}|y)$  follow Gaussian distributions with known mean and covariance, then the parametric linear discriminant analysis method is optimal [20]. The *Kullback-Leibler distance* (KLD) method presented later in the thesis is a parametric method. However, if we do not explicitly know or do not wish to make assumptions about the distributions  $p(\mathbf{x}|y)$ , we must rely on other *non-parametric models* that do not

---

<sup>2</sup>Technically, generalization error refers to the *expected* error, not test error. However, test error is an unbiased estimate of generalization error, and we use the terms interchangeably.

make use of underlying probabilistic distributions. The *support vector machine* (SVM) employed in this thesis falls into this non-parametric scheme. We assume for the remainder of this thesis that we do not know the true underlying distribution of the data, and we either must use a non-parametric classifier or estimate the distribution.

The end result of the various classification methods is the creation of a *classification function*  $h(\mathbf{x}) : \mathbb{R}^d \rightarrow \mathbb{R}$  that maps the input feature vector  $\mathbf{x}$  to the real number line. Implicit to most algorithms is a threshold that determines the cutoff value along the real number line between the two classes in consideration. Generally, this threshold is the origin so that  $f(\mathbf{x}) = \text{sign}(h(\mathbf{x}))$  returns the +1 or -1 classification of the input point  $\mathbf{x}$ . The output of  $f(\mathbf{x})$  returns the binary machine learning decision, while the output of  $h(\mathbf{x})$  relative to the threshold gives us some idea of how confident we can be in our binary decision.

We quantify how well our classifiers  $h(\mathbf{x})$  and  $f(\mathbf{x})$  perform via a measure of the difference between the outputs from the classifier over a set of data and what the true outputs should be over that set of data. This measure is called the *error*. We encounter two types of error. We define *empirical error*  $\hat{\varepsilon}_N$  as [20],

$$\hat{\varepsilon}_N = \frac{1}{N} \sum_{i=1}^N \text{Loss}(h(\mathbf{x}_i), y_i) \quad (2.1)$$

based on our  $N$  training points, and *generalization error*  $\varepsilon$  as [20],

$$\varepsilon = \mathbb{E}_{(\mathbf{x}, y) \sim p(\mathbf{x}|y)} [\text{Loss}(h(\mathbf{x}_i), y_i)] \quad (2.2)$$

based on the unknown distribution  $p(\mathbf{x}|y)$ . The *loss function*  $\text{Loss}(h(\mathbf{x}_i), y_i)$  is some measure of how our prediction at data point  $\mathbf{x}_i$  differs from the true value  $y_i$ . If this loss function is defined relative to the  $h(\mathbf{x})$  classifier as in the expressions above, we get some continuous measure of error. A common continuous loss function, and the one used for least squares regression, is the *squared error loss*,

$$\text{Loss}(h(\mathbf{x}_i), y_i) = (y_i - h(\mathbf{x}_i))^2 \quad (2.3)$$

The loss can also be defined relative to the binary  $f(\mathbf{x})$  classifier to get a discrete measure of error. The *0/1 loss*,

$$\text{Loss}(f(\mathbf{x}_i), y_i) = |y_i - f(\mathbf{x}_i)| \quad (2.4)$$

simply returns a 0 or 1 depending on whether our predicted classification  $f(\mathbf{x}_i)$  matches the true classification  $y_i$ . This 0/1 loss is the standard metric used in the field of machine learning classification. Thus, further reference to loss functions implies the use of the 0/1 loss unless otherwise specified.

The goal of any classifier is to minimize error. Since our specific goal is to predict the classes of a new set of data points, minimizing generalization error is the more appropriate objective since it tells us how well our classifier is *expected* to perform on a new data set sampled from the underlying distribution. However, since we do not know this underlying distribution, the only error measure we can actually calculate is the empirical error on the training set. Thus, most classification methods create classifiers that minimize empirical

error. Note that these classifiers minimize only a function of the training data with no guarantees about performance on a new test set<sup>3</sup>. This class of algorithms follows the *empirical risk minimization* (ERM) principle since the algorithms minimize empirical error, or risk [54]. We outline two such methods, the neural network and  $k$ -nearest neighbor classifiers, in Section 2.3.1.

However, if we can design a classification function via an algorithm that takes into account both empirical *and* generalization error, we hope that we would arrive at a classifier with better generalization performance<sup>4</sup>. Algorithms that create such classifiers follow the *structural risk minimization* (SRM) principle of Vapnik and Chervonenkis [54]. The support vector machine implements this strategy and is the focus of Section 2.3.2.

While we have only scratched the surface of statistical and machine learning theory, a rich and varied subject awaits the interested reader [20] [54].

## 2.2 Texture Feature Representation

With the necessary machine learning classification theory and terminology at our disposal, we can begin to focus on the more specific problem of textural image segmentation that we will apply to the stem cell application. That is, we will discuss how to segment an image into various regions based on the textural properties of those regions. We begin in this section by detailing the feature representation step as it applies to image texture. We turn to the classification step in Sections 2.3 and 2.4.

To fully develop our approach to textural feature representation, we begin by outlining the standard image processing terminology and considering how specifically to define texture. We then discuss some common textural feature paradigms and their associated feature sets before concluding with a description of the specific feature representation that we have adopted for our analysis, the wavelet energy feature.

### 2.2.1 Image Processing Terminology

Since we have implemented our analysis in MATLAB, our terminology follows closely with that used in the MATLAB environment. Let  $I$  be an  $N_x \times N_y$  pixel digital image represented by an  $N_x \times N_y$  matrix with grayscale values drawn from the set  $G = \{1, 2, \dots, N_g\}$ . If the image is originally in RGB format, it is first converted to grayscale via the MATLAB conversion function *rgb2gray*. Individual pixels  $s$  are indexed by the ordered pair  $(x, y)$  where  $x \in \{1, 2, \dots, N_x\}$  and  $y \in \{1, 2, \dots, N_y\}$ . We use the *image coordinate system* where the  $(1, 1)$  and  $(N_x, N_y)$  positions in the image matrix represent the upper left and lower right corners of the image, respectively. Based on this convention,  $I(x, y)$  represent the grayscale value of the  $(x, y)$  image pixel. We represent a subset of a full image as a *neighborhood*,  $\mathcal{N}_s$ , of (generally square) size  $M \times M$  pixels, centered around pixel  $s$ .

---

<sup>3</sup>All is not lost, however, as we assume the test set data is drawn from the same distribution as the training data preserving some hope of satisfactory generalization performance.

<sup>4</sup>While we can measure generalization ability of ERM classifiers via cross validation techniques *external* to the classifiers, the incorporation of generalization error by SRM classifiers is *internal* to the algorithm.

Images are generally classified in two different ways. *Image segmentation* refers to the process of identifying regions with similar characteristics within an image that might contain various fields of differing characteristics [23]. For example, a black and white image may be segmented into a black region and white region. This segmentation is done by analyzing each pixel in an image according to the classification scheme (feature representation and classification algorithm) specified by the user and identifying regions with similar characteristics. *Image classification* refers to the process of determining whether or not an entire image contains a certain characteristic. For example, this could involve classifying a radar image as containing an enemy aircraft or not. This task is accomplished by analyzing the whole image according to the classification scheme specified by the user. We alternatively call this *image categorization* or *image-wise classification*.

## 2.2.2 Defining Texture

The need to classify images into various categories based on their content is ubiquitous. From X-ray machines at the airports to spy photographs, even our national security benefits from good image analysis. Many characteristics such as color, shape and texture may be extracted to aid in the image’s classification [56]. Image classification is indeed a rich field with many avenues on which to proceed. However, as we will see later in the stem cell application, textural features are central to determining stem cell quality. As this stem cell application is a major focus of this thesis, we limit our work to the textural analysis of images.

Image texture has a particularly unwieldy definition, and differences exist among researchers as to the best definition. Jain and Tuceryan reference six different definitions in their overview of texture analysis [52]. They summarize these by defining texture as “a function of the spatial variation in pixel intensities” [52]. A more straightforward and practical approach defines texture as characterized by “the gray-scale value for a given pixel and the gray-level pattern in the neighborhood surrounding the pixel” [25].

Despite the differences in definitions, the commonality in all texture definitions is that an image’s texture is defined on a local image window extracted from the full image, rather than being defined on the entire image or on each individual pixel [56]. In the image processing terminology, texture is considered a *neighborhood operator*. Other common neighborhood operations include convolution, dilation and median filtering [1].

Two basic methods exist for performing analysis with neighborhood operators: the distinct block and sliding neighborhood methods. In the distinct block method, the full image is divided into non-overlapping windows. The neighborhood operator is then applied to each window and the resulting output assigned to *all* pixels in the analyzed window [1]. This method has been employed in the texture analysis literature, commonly for the task of image-wise classification for content-based image retrieval systems [11] [34] [56]. The sliding neighborhood method instead defines a window around an individual pixel, applies the neighborhood operator to that window and assigns the resulting output to the individual pixel itself [1]. Using this method, we can speak concretely of assigning neighborhood features to specific pixels, as in Figure 2-3. Numerous applications in the field of texture analysis, particularly for the task of image segmentation, have used the sliding neighborhood method [25] [30] [39].



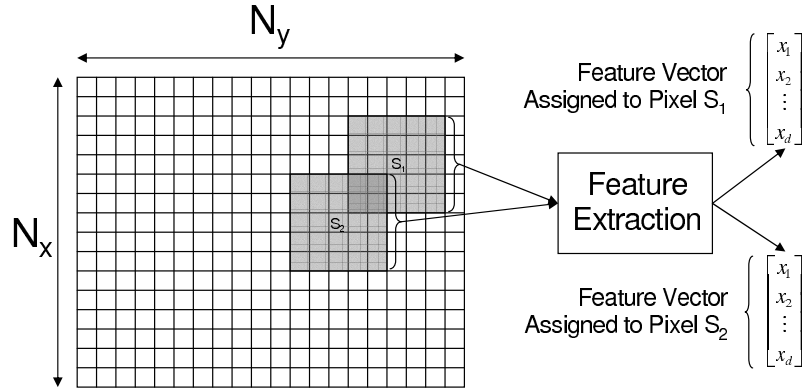


Figure 2-3: Sliding Neighborhood Feature Extraction

For the texture analysis in this thesis, we choose to use the sliding neighborhood method for two reasons. First, since machine learning techniques in general are data intensive, the sliding neighborhood technique allows us to extract more training and test samples from an image. For example, in a  $128 \times 128$  pixel image, we can extract 12,321 sliding neighborhood windows of size  $17 \times 17$  compared with only 49 distinct block, non-overlapping windows. Secondly, for the stem cell application described in Chapter 4, we find that stem cell researchers must be able to accurately pinpoint regions of good and bad quality in their stem cell colonies. By using the sliding neighborhood method, we can classify each pixel in the stem cell colony images as good or bad, thus creating higher resolution segmentations of the images. If we were to use the non-overlapping distinct block method instead, all pixels within a given window block would be assigned the same textural representation, and we would lose resolution.

One concern with using the sliding neighborhood method is that the features assigned to neighboring pixels could be correlated due to the overlap of the windows used to extract the features. Using such neighboring pixels in the training or test sets would yield sets whose samples would not necessarily be independent and identically distributed (IID). While classification methods theoretically require IID samples, we have not found any work in the literature suggesting a way around this problem. Thus, we follow previous work done in this field and continue to use the sliding neighborhood method for image segmentation [25] [30] [39].

A compromise approach might involve analyzing only those pixels at a given interval rather than every pixel in the image. For instance, in the stem cell application of Chapter 4, we classify only every fifth pixel in each stem cell colony image. While the motivation for this approach was originally to speed up implementation, we suspect that it may also reduce the dependencies among pixel classifications. However, a more thorough analysis of the effects of the dependence among pixels in the performance of machine learning classification methods is an avenue for future research.

## 2.2.3 Common Feature Paradigms and Feature Sets

Once we have extracted a window around a specific pixel, we need to analyze that window to calculate its textural feature representation. Given the lack of a clear definition of texture, the number of approaches to textural feature extraction is truly staggering, and it is not our intent to provide an exhaustive survey of these methods. It is difficult even to group all the available methods into clear similarity categories. Methods invariably overlap among the categories, and each researcher fashions his work from a slightly different perspective. However, we attempt to highlight here the most common textural feature representations by grouping them into four major paradigms: statistical, structural, model-based and signal processing methods. More in-depth surveys of the methods and further references can be found in the literature [22] [39] [40] [44] [52] [56].

### Statistical Paradigm

The statistical paradigm describes texture in terms of the statistical distribution of the grayscale pixel values within an image. Haralick’s *co-occurrence matrix* approach is the most common of these methods, and we outline his method briefly here to provide insight into this class of feature representations.

Let  $s_1 = (x_1, y_1)$  and  $s_2 = (x_2, y_2)$  be two pixels in the image, and let  $s_1$  and  $s_2$  be separated by a displacement of  $\mathbf{d} = (d_x, d_y)$  pixels so that

$$s_2 = (x_2, y_2) = (x_1 + d_x, y_1 + d_y) = s_1 + \mathbf{d} \quad (2.5)$$

For a fixed displacement  $\mathbf{d}$ , statistical methods assume that the probability that  $s_1$  and  $s_2$  take on grayscale values of  $i$  and  $j$ , respectively, is governed by the joint probability mass function (PMF)  $P(i, j; \mathbf{d})$ . We may equivalently reference the separation displacement between  $s_1$  and  $s_2$  by an absolute distance  $d$  and angle  $\theta$  relative to the horizontal axis. The PMF of the spatial grayscale values becomes in this case  $P(i, j; d, \theta)$  [18] [53].

Since we do not know the true grayscale distribution  $P(i, j; d, \theta)$ , we must estimate it from empirical data. Haralick accomplishes this with the so-called co-occurrence matrices. For an image  $I$  of size  $N_x \times N_y$  with the set of distinct grayscale values  $G = \{1, 2, \dots, N_g\}$ , Haralick’s method creates symmetric co-occurrence matrices  $\hat{P}(i, j; d, \theta)$  with  $i, j \in G$  specifying grayscale values,  $\theta \in \{0^\circ, 45^\circ, 90^\circ, 135^\circ\}$  defining an angular direction and  $d$  representing the user-defined pixel distance. The  $(i, j)$  entry of  $\hat{P}(i, j; d, \theta)$  holds the total number of pixel pairs in the image, normalized by the total number of pixels in the image, with grayscale values  $i$  and  $j$  such that the two pixels in the pairs lie  $d$  pixels apart in the angular direction  $\theta$  [18]. Thus, for any specified  $d$  value, the method produces four co-occurrence matrices, one for each of the four  $\theta$  values specified above. The value of  $d$  specifies the size of the neighborhood over which it is feasible to estimate the PMF of the grayscale distribution. The resulting co-occurrence matrices serve as an estimate of the true grayscale distribution of the image.

From these co-occurrence matrices, Haralick defines 14 textural features. Table 2.1 shows the four most common features [18]. In the table,  $\sigma_x$ ,  $\mu_x$ ,  $\sigma_y$ ,  $\mu_y$  are the standard deviations and means of the marginal distributions  $\hat{P}(i; d, \theta) = \sum_j \hat{P}(i, j; d, \theta)$  and

Textural Feature Name	Formula
Angular Second Moment	$\sum_i \sum_j \left( \hat{P}(i, j; d, \theta) \right)^2$
Contrast	$\sum_{n=0}^{N_g-1} n^2 \left\{ \sum_i \sum_{j   i-j =n} \hat{P}(i, j; d, \theta) \right\}$
Correlation	$\frac{\sum_i \sum_j (ij) \hat{P}(i, j; d, \theta) - \mu_x \mu_y}{\sigma_x \sigma_y}$
Entropy	$-\sum_i \sum_j \hat{P}(i, j; d, \theta) \log \left( \hat{P}(i, j; d, \theta) \right)$

Table 2.1: Haralick’s Statistical Texture Features

$\hat{P}(j; d, \theta) = \sum_i \hat{P}(i, j; d, \theta)$ , respectively.

In an attempt to speed up the implementation of Haralick’s method by replacing the double summations in the co-occurrence features of Table 2.1 with single summations, Unser proposes using *sum and difference histograms* to estimate the joint grayscale distribution. These sum and difference histograms allow for the exact computation of nine of Haralick’s 14 textural features and the estimation of the remaining five features [53].

Another statistical method is Galloway’s *gray level run length* approach. The basic idea of this approach is that texture can be characterized by first identifying strings of adjacent pixels having the same grayscale value. After calculating the length of these strings for various grayscale values and spatial orientations, five textural features can be calculated [13].

## Structural Paradigm

Where the statistical methods analyze the distribution of individual pixel grayscale values, the structural paradigm instead measures the spatial distribution of blocks of pixels called *textural primitives*. The underlying assumption of structural methods is that the image texture is indeed composed of these distinct textural primitives, rather than simply exhibiting a continuous distribution of grayscale values. The structural approach consists of two steps [52]:

1. Identify and extract the textural primitives
2. Infer the placement rule that governs the distribution of the primitives across the image

In general, specific methods of the structural paradigm do not receive as much attention as those of other paradigms. The main reason for this is that many textures do not actually satisfy the assumption of the presence of repeating primitives [40] and those that do are too regular to be of much real-world interest [30]. For example, the texture on the left in Figure 2-4 is composed of regularly-placed textural primitives and a structural feature representation would be useful for such a texture. In contrast, the more irregular texture on the right is not well-suited for the structural paradigm since it is not composed of clear primitives.

Details on various structural methods can be found in the literature [17] [52]. In fact, Haralick even describes methods that combine the statistical and structural paradigms [17].

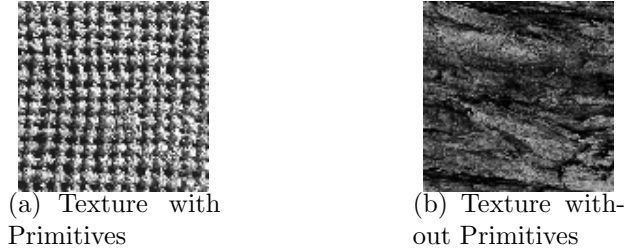


Figure 2-4: Illustration of Textural Primitives

One benefit of the structural paradigm is that if accurate primitives can be identified, texture synthesis reduces to the relatively simple task of replacing the primitives according to the inferred placement rule [52].

### Model-Based Paradigm

The model-based paradigm assumes an image texture model, fits that model to the image being analyzed and uses the model parameters as the textural features of the image [39]. One of the most prevalent model-based approaches involves extracting the multi-resolution autoregressive (AR) features of Mao and Jain. In this method, multiple neighborhoods of various sizes are defined about a pixel, the AR model is fit to those neighborhoods and the model parameters are estimated. The feature representation for the pixel consists simply of all the resulting model parameters [30].

Another more complex model-based method involves modelling an image as a Markov random field (MRF). This approach basically involves defining two Markov processes. The label process  $\{L_s, s \in \mathcal{N}\}$  defines the pixel class label within a neighborhood  $\mathcal{N}$  around the pixel  $s$ . The intensity process  $\{Y_s, s \in \mathcal{N}\}$  governs the distribution of grayscale values about  $s$  within  $\mathcal{N}$  [7]. A thorough discussion of this rich model-based approach is beyond the scope of this work. However, further details on the MRF and other model-based approaches can be found in the literature [7] [40].

### Signal Processing Paradigm

Signal processing, or filtering, methods depart from the previous spatially-oriented paradigms by attempting to incorporate features from the frequency domain of the textural image. These methods generally follow the same two basic steps. First, a filter is applied to the original image to produce a frequency-domain representation of the image. Then the energies across the frequency subbands are calculated and used to represent the texture of the image. Figure 2-5 illustrates this general process [39].

Randen and Husøy present a thorough overview and comparison of various signal processing approaches, and we defer to their work for details on the many signal processing methods available [39]. However, as a motivation for the wavelet energy feature set employed in this thesis, we discuss below two other signal processing methods that give insight into the signal processing paradigm.

The classic approach to extracting frequency information from a signal is via the

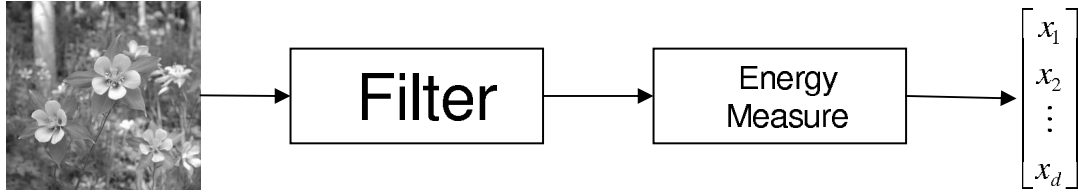


Figure 2-5: Signal Processing Feature Representation

Fourier transform. The *Discrete Fourier Transform* (DFT) of an  $N_x \times N_y$  image  $I$  is defined as [1],

$$F_I(p, q) = \sum_{x=1}^{N_x} \sum_{y=1}^{N_y} I(x, y) e^{-j(2\pi/N_x)px} e^{-j(2\pi/N_y)qy} \quad (2.6)$$

for  $p = 1, 2, \dots, N_x$  and  $q = 1, 2, \dots, N_y$  where  $j$  is the imaginary number  $\sqrt{-1}$ . The Fourier coefficients  $F_I(p, q)$ , or some energy measures extracted from them can then be used as textural features.

The main drawback of the Fourier approach is that the resulting frequency information reflects global characteristics of the image [52]. We note this by observing that the summations in Equation 2.6 are taken over the entire image, thus blurring any spatial variation of the texture. So while we gain frequency information by applying the DFT, we lose spatial information. It would be useful if we could obtain both frequency and spatial information.

One means of retaining spatial information in the frequency domain is to use the *window Fourier transform* (WFT), also called the short-time Fourier transform. This method isolates a portion of the original signal and performs a Fourier transform on that spatially isolated window. By repeating this process over various windows across the entire signal, the method extracts frequency information from the Fourier analysis and maintains spatial information due to the windowing. The width of the window determines the spatial resolution of the analysis. For a one-dimensional signal  $f(x)$ , this window Fourier transform is [52]

$$F_w(u, \xi) = \int_{-\infty}^{\infty} f(x) w(x - \xi) e^{-j2\pi ux} dx \quad (2.7)$$

where  $u$  and  $\xi$  represent the frequency and spatial components of the transformed signal  $F_w(u, \xi)$ . Equation 2.7 becomes the *Gabor transform* when the window function  $w(\cdot)$  is Gaussian [52]. Applying similar ideas to a two-dimensional image yields the two-dimensional Gabor filter method. For example, the impulse response of an even-symmetric (i.e., real part only) two-dimensional Gabor filter with a  $0^\circ$  orientation to the horizontal axis is [23],

$$h(x, y) = \exp \left\{ -\frac{1}{2} \left[ \frac{x^2}{\sigma_x^2} + \frac{y^2}{\sigma_y^2} \right] \right\} \cos(2\pi u_0 x) \quad (2.8)$$

where  $u_0$  is the frequency along the horizontal axis and  $\sigma_x$  and  $\sigma_y$  govern the Gaussian envelope (window) that modulates the frequency sinusoid. Different orientations can be

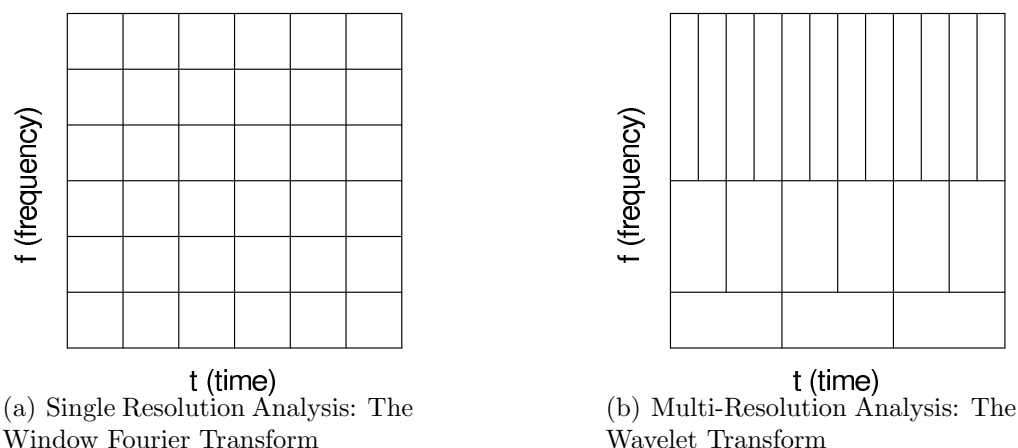


Figure 2-6: Single Resolution Analysis vs. Multi-Resolution Analysis

analyzed via axis rotation of the  $x$  and  $y$  coordinates [23].

Gabor filters have enjoyed much attention and success in the texture classification field [3] [23] [29]. For instance, Jain and Farrokhnia report good performance of the Gabor filter for the task of unsupervised texture segmentation. They extract textural features for each pixel  $s$  by first passing an image through Gabor filters with various orientations and frequencies. They then pass the filtered images through a nonlinearity function and calculate an energy measure—based on the mean absolute deviation—about a local window centered around  $s$ , yielding the textural features for the pixel  $s$  [23].

While quite successful in practice, the Gabor filter has an important drawback. Once a specific window with a specific width is chosen, that window is applied across the entire signal. The result is a single resolution analysis of the image. A multi-resolution analysis, where the image is analyzed over windows of varying widths, would provide a more detailed and accurate analysis of the signal. This is precisely what the *wavelet transform* achieves. We can visualize the difference between the single-resolution approach of the window Fourier transform and the multi-resolution approach of the wavelet transform for the one-dimensional case in Figure 2-6 [9]. The window Fourier transform uses a fixed window width for all frequencies in the image. However, in the wavelet approach, higher frequency regions are analyzed over smaller windows in order to more accurately localize these fast-changing regions. Conversely, larger windows are needed to adequately capture information in the slowly-varying lower frequency regions.

Despite the good performance shown by many of the methods discussed above, we find wavelet analysis to be the most versatile and advanced of the texture analysis methods. Its capability of capturing both the frequency and spatial content of image texture in a multi-resolution framework draws our attention to it for further analysis.

## 2.2.4 Wavelet Energy Features

While wavelet analysis has been used for image compression, signal denoising and edge detection [50], we focus on its application to texture analysis. We present in this section only the basic ideas and terminology of wavelet theory necessary to understand the anal-

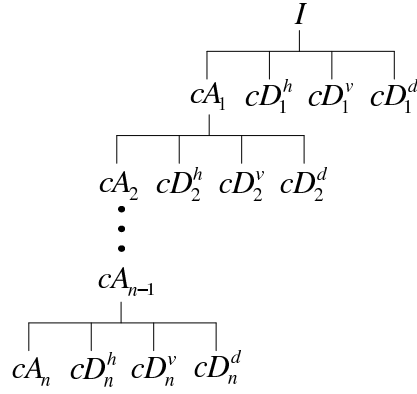


Figure 2-7:  $n$ -level Wavelet Pyramid

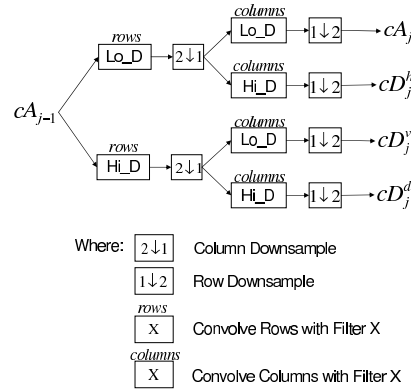


Figure 2-8: Two-Dimensional DWT Schematic

ysis done in this thesis. A thorough treatment of the topic can be found in the text by Strang and Nguyen [50].

The  $n$ -level two-dimensional *discrete wavelet transform* (DWT) iteratively extracts low and high frequency information from an image. At the first level, it decomposes the original image  $I$  into one set of *approximation* coefficients, denoted  $cA_1$ , and three sets of *detail* coefficients, denoted  $cD_1^{(h)}$ ,  $cD_1^{(v)}$  and  $cD_1^{(d)}$ , where  $h$ ,  $v$  and  $d$  refer to the horizontal, vertical and diagonal directions, respectively. At the next iteration, the first-level approximation output,  $cA_1$ , is further decomposed into its approximation and detail coefficients  $cA_2$ ,  $cD_2^{(h)}$ ,  $cD_2^{(v)}$  and  $cD_2^{(d)}$ . The process continues to the  $n$ th level of decomposition, forming the *wavelet pyramid* structure shown in Figure 2-7 [31].

Figure 2-8 shows a schematic of the DWT process at the  $j$ th level of decomposition [31]. Note that for the first decomposition,  $cA_{j-1}$  is replaced with the original image  $I$ .

From the schematic, we note that the approximation coefficients result from the application of two low-pass filters. Thus, the approximation coefficients hold low frequency information, and we can alternatively denote  $cA_j$  as  $LL_j$ . Similarly, the detail coefficients result from the use of either one high-pass and one low-pass filter or two high-pass filters. Thus, they hold the high frequency information, and we can denote  $cD_j^{(h)}$ ,  $cD_j^{(v)}$  and  $cD_j^{(d)}$  as  $LH_j$ ,  $HL_j$  and  $HH_j$ , respectively. This naming convention is used in the

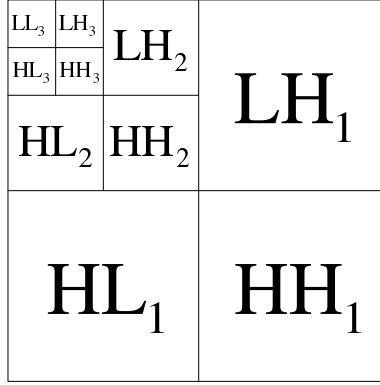


Figure 2-9: 3-level Wavelet Pyramid

alternative visual representation of the DWT pyramid structure shown in Figure 2-9 for  $n = 3$  [9]. While the standard wavelet pyramid method shown here only decomposes the approximation, or LL coefficients, it is also possible to iteratively decompose the detail coefficients as well. The resulting decomposition is called the *wavelet packet* structure.

Each set of coefficients of an  $n$ -level wavelet decomposition represents an image *sub-band*. When using the wavelet coefficients to characterize texture, it is common practice to disregard the  $cA_n$ , or  $LL_n$ , subband as most interesting textural information has been lost due to the iterative low-pass filtering at that subband [8]. Thus, an  $n$ -level wavelet pyramid decomposition will yield a coefficient set for  $3n = B$  subbands, and we denote the  $b$ th subband coefficient set as  $cD[b]$  for  $b = 1, 2, \dots, B$ . The optimal choice for  $n$  depends on the specific application, and we discuss that choice in Chapter 3 when we apply our technique to benchmark textural images.

Along with the decomposition level, the type of wavelet used in the decomposition will affect the resulting wavelet coefficients, and hence the resulting classification performance. Common wavelets and wavelet families available in the MATLAB Wavelet Toolbox include the Haar, Daubechies, Biorthogonal, Coiflets, Symlets, Morlet, Mexican Hat and Meyer wavelets [31]. It is not our intent to discuss all the possible wavelet choices. We settle on the Daubechies 4 wavelet for two reasons. First, it is the wavelet used in the KLD classification method presented in Section 2.4, and we wish to be consistent in the choice of wavelet. Secondly, Randen and Husøy present textural image segmentation results that show generally better performance of the Daubechies 4 wavelet over other wavelets from the Daubechies family [39].

After filtering the textural image and extracting the wavelet coefficients at each sub-band, we need to then create the features that will actually represent the texture. Laws first suggested that energy measures of the filtered image should be used as these textural features [27]. If we have  $N_b$  wavelet coefficients at subband  $b$ , for  $b = 1, 2, \dots, B$ , then the *energy* at that subband is defined as [8],

$$E_b = \frac{1}{N_b} \sum_{i=1}^{N_b} (cD[b]_i)^2 \quad (2.9)$$



where  $cD[b]_i$  is the  $i$ th wavelet coefficient at the  $b$ th subband. For our analysis in Chapter 3 and in the stem cell application, we use the highly-related variance of the wavelet coefficients at each subband as our energy measure,

$$E_b = \frac{1}{N_b} \sum_{i=1}^{N_b} (cD[b]_i - \mu_b)^2 \quad (2.10)$$

where

$$\mu_b = \frac{1}{N_b} \sum_{i=1}^{N_b} cD[b]_i \quad (2.11)$$

is the mean value of the coefficients at the  $b$ th subband. Similarly, the *mean deviation*, or *absolute mean*, is defined as [8],

$$MD_b = \frac{1}{N_b} \sum_{i=1}^{N_b} |cD[b]_i| \quad (2.12)$$

Applying Equations 2.10 and 2.12 across all  $B$  subbands in the wavelet decomposition of an image yields the wavelet energy signature of that image. Thus, the vector,

$$\mathbf{x} = [E_1, E_2, \dots, E_B, MD_1, MD_2, \dots, MD_B] \quad (2.13)$$

is the resulting wavelet energy textural feature representation.

Wavelet energy features from the standard wavelet pyramid structure have been applied successfully by many researchers. Smith and Chang use this approach for the task of image classification and achieve a success rate of 92.14% [49]. Kociołek *et. al.* also perform image classification with wavelet energies with a high success rate [12].

Laine and Fan first showed that wavelet energy features from the wavelet packet structure could be useful for texture classification [26]. Chang and Kuo also use energies in the wavelet packet structure and report that the use of the packet structure as opposed to the standard pyramid structure is better suited for textures with energies concentrated in the lower frequencies [6]. Ma and Manjunath compare the use of energy features between the wavelet packet and pyramid structures [29]. They conclude that the wavelet pyramid structure performs just as well as the wavelet packet structure and does so with a smaller feature vector yielding a lower computational complexity.

Moving beyond the strict use of wavelet energies, Wouwer *et. al.* explore a rich set of wavelet histogram features [8]. Vetterli and Do then estimate a probabilistic distribution on the actual wavelet coefficients for the task of content-based image retrieval [11].

Given the various choices available for textural feature representation using wavelet analysis, we choose to employ energy measures from the standard wavelet pyramid structure. The use of wavelet subband energies has a rich theoretical and practical history and is relatively straightforward to implement. We use the standard wavelet pyramid structure since it's use has shown promising results and requires less computational effort than the wavelet packets structure. Additionally, we find the pyramid structure more useful for comparing the non-parametric SVM performance with the parametric KLD performance

since the KLD method naturally employs the wavelet pyramid structure.

## 2.3 Non-Parametric Classification: The Support Vector Machine

Now that we have a sophisticated means of representing the textural characteristics of images in general, and specifically for the stem cell images we will consider in Chapters 4 and 5, we turn to finding an equally sophisticated method to classify those textural representations.

As discussed in Section 2.1.3, we can partition classification methods into two groups based on the risk minimization principle they employ. Classifiers that implement the empirical risk minimization (ERM) principle simply minimize error on the training set. In contrast, classifiers following the structural risk minimization (SRM) principle attempt to minimize both training error and generalization error. Since the support vector machine implements this more advanced SRM principle, we choose to use it in our work. To support this decision, we discuss the SVM's pleasing theoretical foundation and strong performance on various applications below. First, however, to provide context for our choice, we comment briefly on two of the more popular ERM-based classification methods.

### 2.3.1 Common Non-Parametric Classification Methods

The number of ERM-based classification methods available today is truly immense, and we limit our comments to two of the most popular and well-known of these: neural networks and  $k$ -nearest neighbor.

The basic *neural network* architecture is shown in Figure 2-10 [19]. The input to the network is the standard  $d$ -dimensional feature representation. The two outputs represent the two classes of the binary classification problem. The wrinkle in the network is the presence of the  $m$  nodes of the *hidden layer*. The hidden layer allows for interactions among the input features in a manner similar to the basis function expansion found in regression. While the neural network in the figure contains only one hidden layer, it is possible to design networks with multiple hidden layers. In the figure,  $w_{ji}$  represents the *weight* assigned to the arc from input feature  $i$  to hidden layer node  $j$ . Similarly,  $w_{kj}$  represents the weight of the arc from hidden layer node  $j$  to output class  $k$ . Not shown in the figure, but essential to the neural network classifier, are nonlinear *activation functions* that govern the interaction between the inputs, the hidden layer and the outputs [19].

Thus, in a neural network, the output classification is related to the input point via the hidden layers and the specified activation functions. Training a neural network involves using training data to estimate the arc weights and the form of the activation functions. The so-called *backpropagation algorithm* is the standard method of training [19]. While this is a very elementary description of the neural network, it provides the basic ideas behind the method. Further details can be found in the literature [19] [20].

Advantages of the neural network classifier include its ability to learn the form of the nonlinear activation functions from the training data, the flexibility of the hidden layer architecture to model many complex problems and its strong empirical performance in

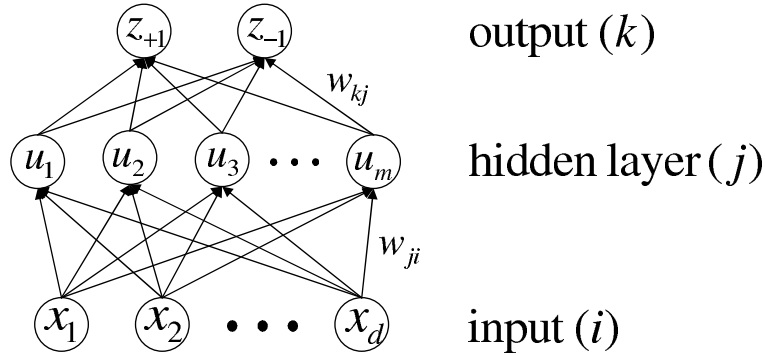


Figure 2-10: Neural Network Architecture

many applications [19]. The main difficulty of the method involves *regularization*, or controlling the complexity of the classifier. If the model contains too many hidden layers or too many nodes within each layer, it will be overly complex and tend to overfit the training data at the expense of generalization performance [19].

The  $k$ -nearest neighbor algorithm is a much simpler classifier than the neural network; it has only one parameter that needs to be specified, namely  $k$ , and has no explicit training step. Given a test point  $\mathbf{x}$ , the algorithm finds the  $k$  nearest training points to the test point. The test point is then assigned the label of the class most represented by those  $k$  nearest neighbors. The Euclidean metric is a standard choice for defining the distance between the test and training points [19].

The  $k$ -nearest neighbor classifier has the advantage of simplicity since only one parameter must be specified. Additionally, it is easy to implement and has performed well in many applications. A drawback of the method is that its decision boundary is difficult to specify analytically and is not generally smooth.

### 2.3.2 The Support Vector Machine

While the ERM-based classification methods discussed above often perform quite well in practice, we feel that the SRM-based support vector machine represents a general improvement over these methods. In fact, in many applications it has shown superior performance to various other classifiers. To fully discuss why this is the case, we must first give some attention to the theory and unique formulation that are the SVM's hallmark.

#### Theoretical Basis for the SVM

Implementing the SRM principle involves minimizing not only the empirical error but also the generalization error. It can be shown that for a classifier  $f$ , parameterized by  $\alpha$ , for some  $\eta$ ,  $0 \leq \eta \leq 1$ , the following bound on generalization error holds [5]:

$$R(\alpha) \leq R_{\text{emp}}(\alpha) + \sqrt{\left( \frac{v(\log(2N/v) + 1) - \log(\eta/4)}{N} \right)}, \text{ with probability } 1 - \eta \quad (2.14)$$

In Equation 2.14,  $R(\alpha)$  is the generalization error,  $R_{\text{emp}}(\alpha)$  is the empirical error based on a training set of  $N$  data points and  $v$  is the *Vapnik Chervonenkis dimension* (VC-dimension) of the family of classifiers  $\mathcal{F}$  that contains  $f$ . The VC-dimension is a measure of the complexity of the family of classifiers  $\mathcal{F}$ , and thus of the specific classifier  $f$ . We see that the generalization error is bounded by the sum of the empirical error and a complexity term. This bound tells us that if we can minimize both the empirical error and VC-dimension of the family of classifiers from which our classifier is drawn, we are guaranteed to generalize well.

We have one step remaining to tie the support vector machine to the SRM principle. Consider  $N$  data points from two classes that are constrained to lie within a hypersphere of radius  $\varphi$ . Vladimir Vapnik shows that a certain type of classifier, namely *maximally separating hyperplanes*, applied to this set of data points has a VC-dimension bounded as [54],

$$v \leq \min\left(\frac{\varphi^2}{\Delta}, N\right) + 1 \quad (2.15)$$

where  $\Delta$  is a measure of the separation achieved between the two classes. Maximizing this separation measure  $\Delta$  can yield a smaller VC-dimension, which in turn yields a tighter upper bound on the generalization error. The support vector machine creates just such a maximally separating hyperplane.

## SVM Formulation

To derive the formulation of the SVM, consider two sets of training data points, each point described by its  $d$ -dimensional input vector, with the points distributed among the  $+1$  and  $-1$  classes. Support vector machines attempt to create a maximally separating hyperplane by separating the two classes with a  $d$ -dimensional hyperplane  $\mathbf{w} \cdot \mathbf{x} + b = 0$  where  $\mathbf{w} \in \mathfrak{R}^d$ ,  $b \in \mathfrak{R}$ . If a set of points can be separated without error by such a hyperplane, the set is termed *linearly separable*. Generally many different hyperplanes will separate a linearly separable set of training data and achieve zero empirical error. The ERM principle would not distinguish among such hyperplanes. However, the SRM principle defines the *optimal*, or maximally separating hyperplane as the one that maximizes *margin* where the margin is defined as the distance from the nearest point of either class to the hyperplane. Figure 2-11 illustrates this concept of maximizing margin in  $\mathfrak{R}^2$  ( $d = 2$ ). By finding a hyperplane that achieves both zero empirical error and a large separation between classes, we expect to have low generalization error according to the SRM principle.

We should make an important note here regarding the link between the SRM principle and the support vector machine classifier. Chris Burges makes clear that there is no *rigorous* connection between the SVM and the SRM principle [5]. That is, the SRM bounds in Equations 2.14 and 2.15 are generally not tight for a given implementation of the SVM. Despite this, the concept of simultaneously achieving low empirical error and maximizing margin as implemented by the SVM has a strong foundation in the SRM principle and represents a significant improvement over methods that simply minimize empirical error.

In practice, the support vector machine finds the maximally separating hyperplane by solving a quadratic optimization problem. First, let  $y_i = 1$  for all points  $\mathbf{x}_i$  in class

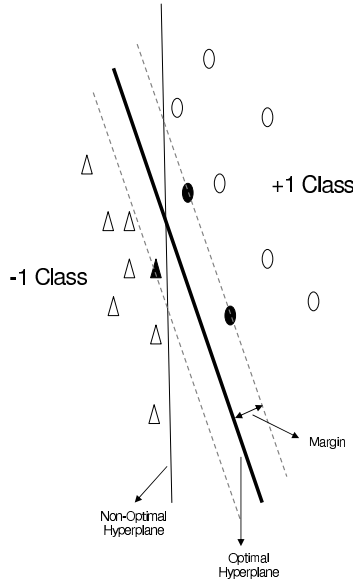


Figure 2-11: Separating Hyperplanes in  $\mathfrak{R}^2$

$+1$  and  $y_i = -1$  for all points  $\mathbf{x}_i$  in class  $-1$ . Assuming that the training data is indeed linearly separable, we define a hyperplane by the pair  $(\mathbf{w}, b)$  such that [35],

$$\mathbf{w} \cdot \mathbf{x}_i + b \geq 1 \quad \forall i \text{ such that } y_i = +1 \quad (2.16)$$

$$\mathbf{w} \cdot \mathbf{x}_i + b \leq -1 \quad \forall i \text{ such that } y_i = -1 \quad (2.17)$$

where  $\mathbf{w} \in \mathfrak{R}^d$  and  $b \in \mathfrak{R}$  is a scalar bias term. We can write the above expressions more compactly as [35],

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 \quad \forall i = 1, 2, \dots, N \quad (2.18)$$

This expression represents the constraint that all training points must be correctly classified by the hyperplane.

The distance from a point  $\mathbf{x}_i$  to the hyperplane is  $\frac{|\mathbf{w} \cdot \mathbf{x}_i + b|}{\|\mathbf{w}\|}$ . If we impose the normalization  $\min_{i=1 \dots N} |\mathbf{w} \cdot \mathbf{x}_i + b| = 1$ , the distance from the hyperplane to the nearest point of either class, or the margin, is simply  $\frac{1}{\|\mathbf{w}\|}$ . Since we want to maximize this margin while ensuring that all points are correctly classified according to Equation 2.18, we can pose the following optimization problem [35]:

$$\begin{aligned} & \text{maximize}_{\mathbf{w}, b} && \frac{1}{\|\mathbf{w}\|} \\ & \text{s.t.} && y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 \quad \forall i = 1, 2, \dots, N \end{aligned}$$

Noting that maximizing  $\frac{1}{\|\mathbf{w}\|}$  is equivalent to minimizing  $\frac{1}{2}\|\mathbf{w}\|^2$ , we have the following

equivalent quadratic optimization formulation [35]:

$$\begin{aligned} & \text{minimize}_{\mathbf{w}, b} && \frac{1}{2} \|\mathbf{w}\|^2 \\ & \text{s.t.} && y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 \quad \forall i = 1, 2, \dots, N \end{aligned}$$

Solving this problem returns the optimal pair  $(\mathbf{w}, b)$  from which we create the classification functions  $h(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b$  and  $f(\mathbf{x}) = \text{sign}(\mathbf{w} \cdot \mathbf{x} + b)$  for a new data point  $\mathbf{x}$  [35].

We call the formulation above the *linear primal formulation* and note that it is only useful for data that is linearly separable in its original  $d$ -dimensional space. Using duality theory, we can find an equivalent *linear dual formulation* that is often easier to solve than the primal. Furthermore, by allowing some training points to be misclassified (violating Equation 2.18), we can create the *soft margin primal formulation* and *soft margin dual formulation*. Finally, by projecting the data into a higher-dimensional space, we arrive at the most flexible and powerful of the SVM formulations, the *nonlinear soft margin dual formulation*. We outline these more advanced formulations in Appendix A. Further details can also be found in the excellent tutorial by Burges [5].

## SVM Features and Terminology

As mentioned above, the most powerful of the SVM formulations is the *nonlinear soft margin dual formulation*. This formulation also serves as a useful vehicle to discuss much of the unique terminology and features of the support vector machine. We formulate it as [5],

$$\text{maximize}_{\{\lambda_1, \lambda_2, \dots, \lambda_N\}} \sum_{i=1}^N \lambda_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \lambda_i \lambda_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) \quad (2.19)$$

$$\text{s.t.} \quad \sum_{i=1}^N \lambda_i y_i = 0 \quad (2.20)$$

$$\lambda_i \leq C \quad \forall i = 1, 2, \dots, N \quad (2.21)$$

$$\lambda_i \geq 0 \quad \forall i = 1, 2, \dots, N \quad (2.22)$$

where  $\lambda_i$  is the *Lagrange multiplier* associated with the training point  $\mathbf{x}_i$ ,  $K(\cdot, \cdot)$  is a *kernel function* and  $C$  is a *cost penalty*. Solving this quadratic optimization problem returns the  $\lambda_i$  values that define the maximally separating hyperplane (See Equations 2.25, 2.26 and 2.27).

We first note that the optimization problem above scales with the the number of data points  $N$  rather than the dimensionality  $d$  of the input points. Thus, support vector machines, unlike many other classification methods, can handle data with high-dimensional feature vectors.

Furthermore, the  $i$ th Lagrange multiplier  $\lambda_i$  tells us how influential the  $i$ th training point is in defining the separating hyperplane. If  $\lambda_i = 0$ , then the input point  $\mathbf{x}_i$  has no influence in characterizing the hyperplane, and we can disregard such points without

consequence. Conversely, those points  $\mathbf{x}_i$  for which  $\lambda_i > 0$  are called *support vectors*, and we define the set  $S = \{\mathbf{x}_i : \lambda_i > 0\}$  as the *support vector set*. The separating hyperplane is fully characterized by only those points in  $S$ . The cardinality of  $S$  is generally less (and never more) than the number of input points  $N$ ; thus, the SVM actually scales with only a subset of the original input points.

The kernel function  $K(\mathbf{x}_i, \mathbf{x}_j)$  is a function in the original  $d$ -dimensional input space that calculates the inner product of the two input vectors  $\mathbf{x}_i$  and  $\mathbf{x}_j$  in some higher-dimensional *feature space*. The use of the kernel function allows us to find the optimal hyperplane in the high-dimensional feature space and map that hyperplane back into the original input space as a non-linear separator. Two common kernel functions are the *polynomial kernel* of degree  $p$ ,

$$K(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i \cdot \mathbf{x}_j + 1)^p \quad (2.23)$$

and the *radial basis function kernel* with width  $\sigma$ ,

$$K(\mathbf{x}_i, \mathbf{x}_j) = e^{-\|\mathbf{x}_i - \mathbf{x}_j\|^2 / 2\sigma^2} \quad (2.24)$$

Finally, the cost penalty  $C$  controls the tradeoff between empirical error and margin. A large  $C$  will sacrifice some margin width for better empirical performance (less misclassified training points), while a small  $C$  will attempt to create a larger margin at the expense of more misclassified training points.

Once we have solved for the optimal multipliers  $\{\lambda_1, \lambda_2, \dots, \lambda_N\}$  we can create the classification functions  $h(\mathbf{x})$  and  $f(\mathbf{x})$  for a new input point  $\mathbf{x}$  as [5],

$$h(\mathbf{x}) = \sum_{i \in S} \lambda_i y_i K(\mathbf{x}_i, \mathbf{x}) + b \quad (2.25)$$

$$f(\mathbf{x}) = \text{sign} \left( \sum_{i \in S} \lambda_i y_i K(\mathbf{x}_i, \mathbf{x}) + b \right) \quad (2.26)$$

where

$$b = y_i - \sum_{j=1}^N y_j \lambda_j K(\mathbf{x}_j, \mathbf{x}_i), \text{ for } i \text{ such that } \mathbf{x}_i \in S \quad (2.27)$$

Geometrically,  $h(\mathbf{x})$  returns the signed Euclidean distance from the test point  $\mathbf{x}$  to the optimal hyperplane, and  $f(\mathbf{x})$  returns the binary  $+1$  or  $-1$  classification of  $\mathbf{x}$ . We interpret  $h(\mathbf{x})$  as the confidence we have in our binary decision. Larger positive/negative values of  $h(\mathbf{x})$  imply higher confidence in our corresponding  $+1/-1$  decision.

## SVM Performance

In terms of performance, a number of unique and useful features make the support vector machine more attractive than other classification methods. Perhaps the most beneficial of these properties is the fact that the SVM attempts to minimize not only empirical error but also the more appropriate generalization error by maximizing margin. This suggests

that it will perform better on a test data set than other methods, such as neural network and nearest neighbor classifiers, that only minimize training error.

In further contrast to the neural network, the SVM is naturally *regularized* in two senses. First, by adjusting the cost penalty  $C$ , we can easily control its complexity and limit overfitting. A proper setting of  $C$  allows for the misclassification of some training points in exchange for a larger margin and better generalization performance. Secondly, Ryan Rifkin shows that the standard SVM formulation can actually be derived directly from the deep and well-known concepts of regularization theory [42]. A full discussion of this approach is outside the scope of our work; however, it gives further evidence of the theoretical foundation of the SVM.

The concise and flexible formulation of the SVM as a quadratic optimization problem also yields many beneficial features. For example, we can choose to work with either the primal or dual version of the problem, whichever is easiest. The convexity of both formulations guarantees a globally optimal solution. This convexity property, along with duality theory, also allows for the derivation of Edgar Osuna's *active set algorithm*, a fast solution method for the dual problem [35]. The kernel function allows us to find a linear separator in a high dimensional feature space without actually mapping the input points into that space. Thus, while the resulting decision boundary may be highly nonlinear in the original space, its linearity in the feature space results in a concise classification function that is easily expressed in the original space. Overall, few other classifiers are as flexible or exhaustive in their formulation properties.

As mentioned above, the size of the SVM scales with a generally small subset of the number of training points rather than with the dimensionality of the inputs. This allows the SVM to efficiently solve problems with high dimensional inputs.

Finally, the support vector machine method has repeatedly shown superior performance to other popular classification methods in various real-world applications. Osuna *et. al.* report an improved rate of face detection in images from 94.6% to 97.1% over a neural network classifier [36]. For the task of digit recognition in the US Postal Service digit database, the SVM yields a test error rate of 4.2% compared to error rates of 6.7% for the classic radial basis function network classifier, 5.9% for a nearest neighbor method and 5.9% for a two-layer perceptron (neural network) [45]. For the task of texture classification, the SVM has been shown to improve performance over a conventional neural network by approximately 5.0% [25].

The twin aspects of solid theory and good empirical performance across numerous and various applications leads us to choose the SVM as the machine learning classifier employed in this thesis.

## 2.4 Parametric Classification: The Kullback-Leibler Distance Classifier

Up to this point, we have considered the machine learning classification task in two separate and unrelated steps, feature representation and classification, in the manner of Figure 2-2. This setup is useful for two main reasons. First, splitting up the steps



allows researchers to focus on that part of the problem for which they are particularly suited. Computer scientists and statisticians can theorize and formulate classification algorithms in a very general setting. Simultaneously, application-driven scientists can derive powerful feature representations based on their knowledge of the specific problem domain. By then combining the best feature representation with the best classification method, the abilities of two expert communities are combined quite easily. The other major benefit of splitting up the machine learning tasks is that the various available feature representations can be paired with the many classification algorithms to produce a vast amount of experimental work that can be performed. Given  $n$  feature representations for a problem and  $m$  classification methods,  $n \cdot m$  different experiments can be run and analyzed to find the optimal pairing of feature representation and classification method for a given problem.

Despite the benefits outlined above, we might also find it useful to tailor the two tasks to each other. For example, if we suspect that the data which we are attempting to classify follows a certain probability distribution, we would want to design a parametric classification method that takes that knowledge into account. This is precisely the goal of the Kullback-Leibler distance (KLD) classifier used in this thesis. This classifier is an adaptation of a method proposed by Martin Vetterli and Minh Do for the task of content-based image retrieval (a specific application of the image-wise classification problem described in Section 2.2.1) [11]; we adapt it for the task of image segmentation as described in Section 2.2.1.

### 2.4.1 The Generalized Gaussian Distribution for Wavelet Coefficient Modelling

Vetterli and Do make the assumption that the wavelet coefficients at each subband extracted from a textural image follow a generalized Gaussian distribution (GGD),

$$p(x; \alpha, \beta) = \frac{\beta}{2\alpha\Gamma(1/\beta)} e^{-(|x|/\alpha)^\beta} \quad (2.28)$$

where

$$\Gamma(z) = \int_0^\infty e^{-t} t^{z-1} dt \quad (2.29)$$

is the Gamma function defined on  $z > 0$ . The two GGD parameters  $\alpha$  and  $\beta$  control the width and decreasing rate of the distribution's peak. The GGD reduces to the Laplacian and Gaussian distributions for  $\beta = 1$  and  $\beta = 2$ , respectively [11]. In fact, as  $\beta \rightarrow \infty$ , the GGD approaches the uniform distribution. Figure 2-12 shows the GGD for these three parameter settings. The bottom line is that the GGD is a very general, unimodal and symmetric distribution able to model a large range of data through the estimation of its two parameters.

The applicability of the GGD for modelling wavelet coefficients has been demonstrated by showing that estimated GGD's fit well the actual wavelet coefficient histograms [8]. Vetterli and Do further justify this by performing texture synthesis using the estimated GGD's to generate synthesized sets of wavelet coefficients which are used to create an

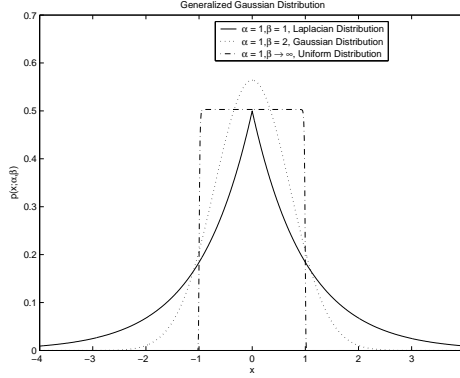


Figure 2-12: Generalized Gaussian Distribution

estimated textural image via the inverse wavelet transform. The visual similarity between the original and synthesized images demonstrates the accuracy of the GGD model of wavelet coefficients [11].

As discussed in Section 2.2.4, at each level of wavelet decomposition we extract three sets of coefficients, each representing a different subband. Thus, for an  $n$ -level decomposition, we have coefficients for  $3n = B$  subbands. By modelling each set of coefficients as a GGD and estimating the two parameters for each ( $\alpha^b$  and  $\beta^b$  for  $b = 1, 2, \dots, B$ ), we can extract  $2B$  parameters for each image.

## 2.4.2 The Kullback-Leibler Distance

After extracting the GGD parameters, Vetterli and Do turn to designing a classification method that takes advantage of the parametric feature representation. They conclude that a test point should be classified to the class whose wavelet coefficient distribution is closest to the distribution of the test point as measured by the *Kullback-Leibler distance* metric. While Vetterli and Do use their KLD classification method for the task of image-wise classification, we have adapted it for image segmentation and present the ideas in that vein below.

Let  $p_k^b(x; \alpha_k^b, \beta_k^b)$ ,  $k = \{+1, -1\}$ ,  $b = 1, 2, \dots, B$ , represent the GGD of the wavelet coefficients of the  $k$ th class at the  $b$ th subband. Let  $p_t^b(x; \alpha_t^b, \beta_t^b)$  be the corresponding GGD of a test point which we are trying to classify. Vetterli and Do show that we should choose the class that satisfies the KLD decision rule [11],

$$k_{\text{opt}} = \arg \min_{k \in \{+1, -1\}} D(p_t^b(x; \alpha_t^b, \beta_t^b) \| p_k^b(x; \alpha_k^b, \beta_k^b)) \quad (2.30)$$

where

$$D(p_t^b(x; \alpha_t^b, \beta_t^b) \| p_k^b(x; \alpha_k^b, \beta_k^b)) = \int p_t^b(x; \alpha_t^b, \beta_t^b) \log \frac{p_t^b(x; \alpha_t^b, \beta_t^b)}{p_k^b(x; \alpha_k^b, \beta_k^b)} dx \quad (2.31)$$

is the Kullback-Leibler distance operator at the  $b$ th subband.

Vetterli and Do show that for a GGD, Equation 2.31 can be simplified to the closed

form expression [11],

$$D(p_t^b(x; \alpha_t^b, \beta_t^b) \| p_k^b(x; \alpha_k^b, \beta_k^b)) = \log \left( \frac{\beta_t^b \alpha_k^b \Gamma(1/\beta_k^b)}{\beta_k^b \alpha_t^b \Gamma(1/\beta_t^b)} \right) + \left( \frac{\alpha_t^b}{\alpha_k^b} \right)^{\beta_k^b} \frac{\Gamma((\beta_k^b + 1)/\beta_t^b)}{\Gamma(1/\beta_t^b)} - \frac{1}{\beta_t^b} \quad (2.32)$$

From Equation 2.32, we note that we can calculate the Kullback-Leibler distance between a test point and the  $k$ th class using only the estimated GGD parameters of the  $k$ th class and the test pixel<sup>5</sup>.

Furthermore, under the assumption that the wavelet coefficient distributions are independent across the  $B$  subbands of the decomposed image, Vetterli and Do show that Equation 2.30 expands to [11],

$$k_{\text{opt}} = \arg \min_{k \in \{+1, -1\}} \sum_{b=1}^B D(p_t^b(x; \alpha_t^b, \beta_t^b) \| p_k^b(x; \alpha_k^b, \beta_k^b)) \quad (2.33)$$

That is, with the independence assumption, we can simply sum up the KLD's across all subbands.

For notational consistency, we pose the KLD decision rule of Equation 2.33 in the manner of the SVM decision rules of Section 2.3.2. If  $\mathbf{x}$  is some test point, then

$$\text{KLD}(\mathbf{x}, +1) = \sum_{b=1}^B D(p_t^b(x; \alpha_t^b, \beta_t^b) \| p_{+1}^b(x; \alpha_{+1}^b, \beta_{+1}^b)) \quad (2.34)$$

and

$$\text{KLD}(\mathbf{x}, -1) = \sum_{b=1}^B D(p_t^b(x; \alpha_t^b, \beta_t^b) \| p_{-1}^b(x; \alpha_{-1}^b, \beta_{-1}^b)) \quad (2.35)$$

are the Kullback-Leibler distances between the test points's wavelet coefficient distribution and the +1 and -1 classes' wavelet coefficient distributions, respectively. Then we write Equation 2.33 as the KLD classification function,

$$f(\mathbf{x}) = \text{sign}(\text{KLD}(\mathbf{x}, -1) - \text{KLD}(\mathbf{x}, +1)) \quad (2.36)$$

Additionally, we introduce a KLD confidence measure,

$$h(\mathbf{x}) = \text{KLD}(\mathbf{x}, -1) - \text{KLD}(\mathbf{x}, +1) \quad (2.37)$$

With these concepts in hand, we are prepared to explore and compare the performance of the non-parametric SVM classification scheme using the wavelet energy features and the parametric KLD classification scheme using the generalized Gaussian distribution features and Kullback-Leibler distance classifier. We perform this analysis first on benchmark textural images in Chapter 3 to show the usefulness of the methods. We then turn to applying the methods to the stem cell application of Chapters 4 and 5.

---

<sup>5</sup>We discuss in Chapter 3 how specifically we calculate the parameters  $\alpha_k$ ,  $\beta_k$ ,  $\alpha_t$  and  $\beta_t$  that represent the  $k$ th class and each test pixel.

[This page intentionally left blank.]

# Chapter 3

## Benchmark Analysis

Having discussed the theoretical background and mathematical framework of the textural image segmentation problem in Chapter 2, we turn in this chapter to the application of those concepts in practice. We demonstrate the applicability of the non-parametric support vector machine (SVM) classifier for the task of textural image segmentation using the wavelet energy feature representation. To the best of our knowledge, this combination of the SVM and wavelet energy features has not been previously studied, and our analysis presented here constitutes a useful addition to the field of texture classification. We also explore the adaptation of Vetterli and Do's parametric Kullback-Leibler distance (KLD) method for the task of textural image segmentation. We limit our analysis to binary textural image segmentation—the segmentation of images containing only two texture fields—since that is the form of the stem cell application presented later.

We use this chapter to explicitly describe the algorithmic implementation of the two classification methods. We report results of the algorithms on the VisTex set of benchmark textural images and compare the SVM and KLD approaches based on empirical performance on these images. We consider two main performance measures in this analysis:

1. Basic error rates on various test images
2. Test error rates for different amounts of training data

The overall goal of this chapter is two-fold. We first wish to establish the usefulness of the SVM and KLD textural image segmentation methods before applying them to the stem cell problem in Chapters 4 and 5. Secondly, we hope to elucidate some of the differences between the two approaches and comment on the applicability of non-parametric and parametric classification methods in general to the problem of textural image segmentation.

### 3.1 Problem Description

We pose the problem of *binary textural image segmentation* to analyze the performance of the SVM and KLD classification schemes. Specifically, given an image with two textural fields, we attempt to segment the image into those two fields. We consider images of the

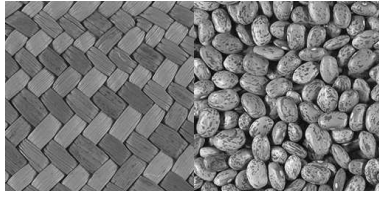


Figure 3-1: Example Two-Texture Composite Test Image

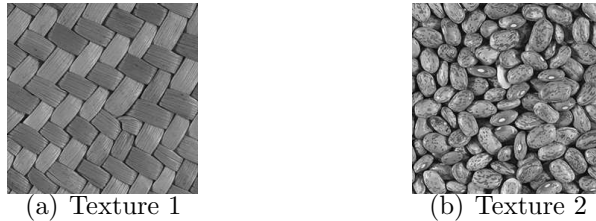


Figure 3-2: Example Training Images

form shown in Figure 3-1 where the two textural fields lie side-by-side. We refer to these two-texture images as the *composite test images*, or simply *test images*.

For each test image, we train the SVM and KLD classifiers using two *training images*—one for each of the two textures in the composite test image—each containing only a single textural field. Figure 3-2 shows the training images that are associated with the test image in Figure 3-1.

### 3.1.1 Benchmark Textural Images

We have chosen to use the VisTex database from the MIT Media Lab to serve as a standard set of benchmark textural images on which to apply our SVM and KLD segmentation algorithms [55]. This database contains a rich set of texture images. Each image contains only a single textural field that we can partition for use in creating training and test images such as those shown in Figures 3-1 and 3-2. While the database holds over 100 texture images, we employ only 25 of those images for our analysis. We have chosen images containing generally uniform textural characteristics across the entire image. Table 3.1 lists these 25 images according to their original file names in the VisTex database and the shorter names used in this thesis.

### 3.1.2 Image Preprocessing

We begin our analysis by performing two image preprocessing steps. Since the original VisTex images are in RGB color format, we first convert the RGB images to grayscale via the MATLAB function `rgb2gray`. We then scale the original 256 grayscale intensity levels to the interval  $[-1\ 1]$  for computational stability.

<b>VisTex File Name</b>	<b>Thesis Name</b>
Bark.0001.ppm	Bark1
Bark.0003.ppm	Bark3
Bark.0011.ppm	Bark11
Brick.0000.ppm	Brick0
Brick.0002.ppm	Brick2
Brick.0004.ppm	Brick4
WheresWaldo.0001.ppm	WheresWaldo1
Fabric.0000.ppm	Fabric0
Fabric.0004.ppm	Fabric4
Fabric.0009.ppm	Fabric9
Fabric.0015.ppm	Fabric15
Flowers.0002.ppm	Flowers2
Food.0000.ppm	Food0
Food.0005.ppm	Food5
Food.0006.ppm	Food6
Grass.0001.ppm	Grass1
Leaves.0003.ppm	Leaves3
Leaves.0008.ppm	Leaves8
Leaves.0010.ppm	Leaves10
Metal.0003.ppm	Metal3
Sand.0003.ppm	Sand3
Stone.0004.ppm	Stone4
Terrain.0005.ppm	Terrain5
Water.0000.ppm	Water0
Wood.0001.ppm	Wood1

Table 3.1: 25 Benchmark VisTex Texture Images

Thesis Names	Thesis Designation
Brick0 – WheresWaldo1	P1
Leaves3 – Flowers2	P2
Leaves3 – Food6	P3
Fabric9 – Water0	P4
Water0 – Bark1	P5
Leaves10 – Sand3	P6
Food5 – Grass1	P7
Brick2 – Terrain5	P8
Terrain5 – Food0	P9
Wood1 – Fabric15	P10
Fabric4 – Flowers2	P11
Fabric9 – Sand3	P12
Brick0 – Fabric9	P13
Fabric0 – Leaves8	P14
Stone4 – Brick4	P15
Metal3 – Bark11	P16
Fabric0 – Bark3	P17
WheresWaldo1 – Leaves8	P18

Table 3.2: 18 Benchmark Textural Image Pairs

### 3.1.3 Training and Test Images

We create the composite test images by considering various pairs of textures drawn from the 25 images in Table 3.1. Table 3.2 lists the 18 pairs that we have randomly chosen to analyze. These pairs are referenced by the thesis names of the two images in the pair and according to the pairwise designations  $P_i$ ,  $i = 1, 2, \dots, 18$ , used to reference each pair throughout the remainder of this chapter.

To create the two training images for each pair, we extract  $256 \times 256$  pixel subimages from each of the two original  $512 \times 512$  images in the pair. To create the composite test image, we extract  $128 \times 128$  subimages from each of the original images in the pair and place them side-by-side. This results in a composite test image of size  $128 \times 256$  pixels. The subimages extracted to create the training and test images do not overlap in the original image and thus constitute statistically independent training and test data.

We show each of the 18 composite test images corresponding to the 18 image pairs of Table 3.2 in Figure 3-3.

By training the SVM and KLD classifiers on the two training images and testing on the composite test image for each pair, we can calculate empirical measures of the performance of the two methods across a variety of textures.



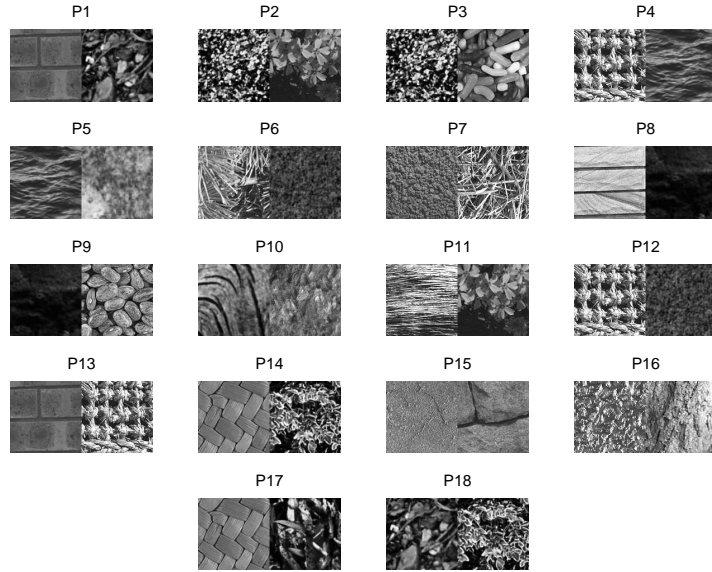


Figure 3-3: 18 Two-Texture Composite Test Images

## 3.2 Training and Test Algorithms

With the 18 benchmark image pairs of Table 3.2 at our disposal, we turn to presenting the specific algorithms that we use to analyze those textural images. We describe these training and test algorithms for both the support vector machine and Kullback-Leibler distance classification methods. Furthermore, we highlight the various parameters that need to be set to fully specify the algorithms, and we discuss our method of tuning these parameters. With slight modifications and specifications which will be noted later, the algorithms presented here are also those used in the stem cell application.

### 3.2.1 Support Vector Machine Method

The *SVM training algorithm* consists of the following steps as depicted in Figure 3-4:

1. Randomly extract  $N$   $M \times M$  windows (training points) from each training image
2. Send these  $N$  windows from each class to the  $n$ -level wavelet energy feature extractor
3. The feature extractor returns the  $N$  wavelet energy feature vectors corresponding to the  $N$  extracted windows for each class
4. Send these  $N$  feature vectors from each class to the support vector machine
5. The support vector machine solves the quadratic optimization problem described in Equations 2.19 through 2.22 in Section 2.3.2
6. The support vector machine returns the classification functions  $h(\mathbf{x})$  and  $f(\mathbf{x})$ —described as Equations 2.25 and 2.26 in Section 2.3.2—used to classify a new input point  $\mathbf{x}$

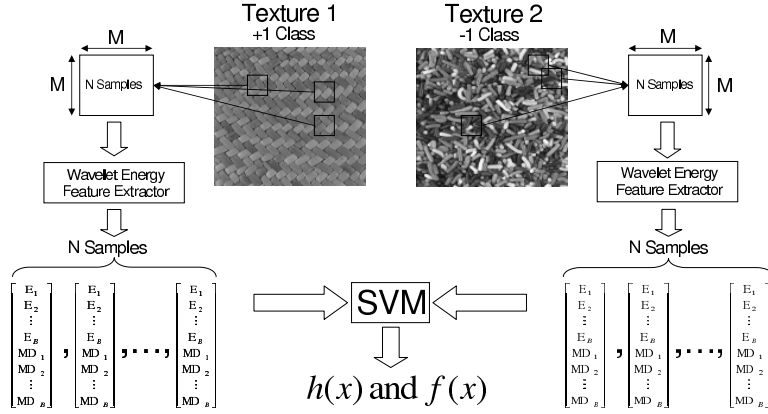


Figure 3-4: Support Vector Machine Training Algorithm

The *SVM test algorithm* is depicted in Figure 3-5. It consists of the following steps:

1. Extract all  $N_{test}$   $M \times M$  windows (test points) centered about each pixel (except those within  $\frac{M-1}{2}$  pixels of the image border to avoid exceeding the image boundaries) of the composite test image
2. Send these windows to the  $n$ -level wavelet energy feature extractor
3. The feature extractor returns the wavelet energy feature vectors corresponding to the extracted windows
4. Evaluate  $h(\mathbf{x}_i)$  and  $f(\mathbf{x}_i)$  (from the training algorithm) for each test point  $\mathbf{x}_i$
5. Calculate the percentage of misclassified test points; that is, points for which  $f(\mathbf{x}_i) \neq y_i$  where  $y_i$  is the true class of the test point  $\mathbf{x}_i$
6. Plot a *binary reclassified image* where pixels  $\mathbf{x}_i$  for which  $f(\mathbf{x}_i) = +1$  are plotted white and pixels  $\mathbf{x}_i$  for which  $f(\mathbf{x}_i) = -1$  are plotted black

As shown in Figure 3-5, we quantify the error rate of the classifier by the percentage of misclassified pixels in the composite test image. We extend this idea to include more interesting performance measures, such as those incorporating the confidence outputs  $h(\mathbf{x})$ , in the stem cell application in Chapter 4.

For our implementation of the SVM classification method, we have used Thorsten Joachims' *SVM<sup>light</sup>* solver implemented in C [24]. To interface this solver with our MATLAB code, we have used Anton Schwaighofer's interface code, version 0.92 [46].

### 3.2.2 Kullback-Leibler Distance Method

As shown in Figure 3-6, the *KLD training algorithm* consists of the following steps:

1. Randomly extract  $N$   $M \times M$  windows (training points) from each training image
2. Perform an  $n$ -level wavelet decomposition on these  $N$  windows from each class

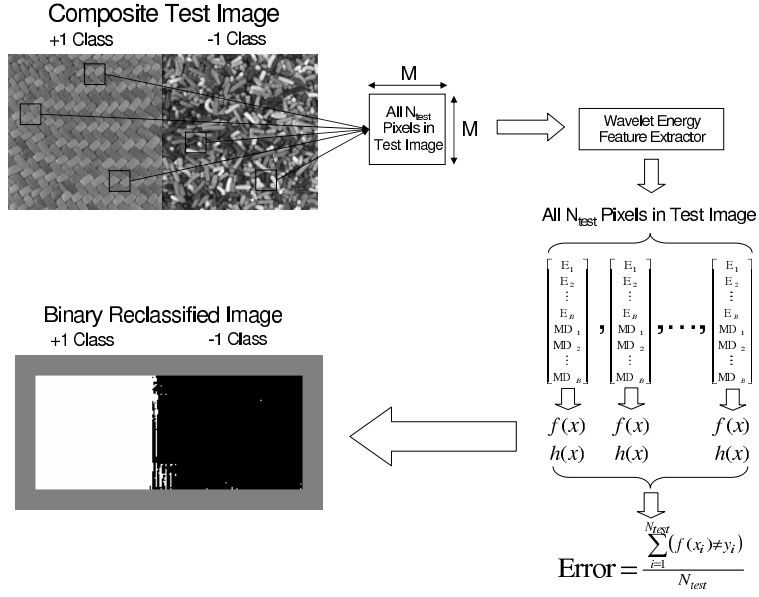


Figure 3-5: Support Vector Machine Test Algorithm

3. The wavelet decomposition yields  $3n = B$  sets of wavelet coefficients for each training point
4. Estimate the generalized Gaussian distribution (GGD) parameters  $\alpha_i^b$  and  $\beta_i^b$ ,  $b = 1, 2, \dots, B$ , for the  $B$  sets of coefficients for each training point  $\mathbf{x}_i$ <sup>1</sup>
5. Average the  $N$  GGD parameters at each of the  $B$  subbands to yield the trained KLD parameters,

$$\alpha_k^b = \frac{1}{N} \sum_{i=1}^N \alpha_i^b \quad (3.1)$$

$$\beta_k^b = \frac{1}{N} \sum_{i=1}^N \beta_i^b \quad (3.2)$$

for each class  $k \in \{+1, -1\}$

Figure 3-7 shows the *KLD test algorithm* consisting of the following steps:

1. Extract all  $N_{test}$   $M \times M$  windows (test points) centered about each pixel (except those within  $\frac{M-1}{2}$  pixels of the image border to avoid exceeding the image boundaries) of the composite test image
2. Perform an  $n$ -level wavelet decomposition on these windows
3. The wavelet decomposition yields  $3n = B$  sets of wavelet coefficients for each test point

<sup>1</sup>Vetterli and Do use a combination of the method of moments and method of maximum likelihood to numerically estimate these GGD parameters.

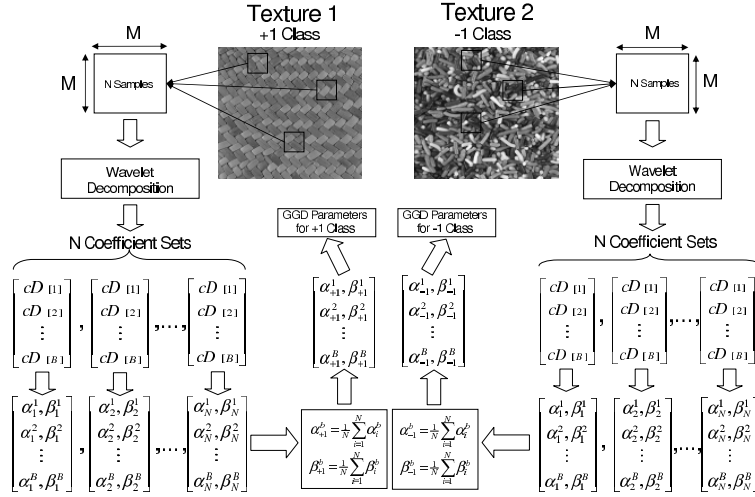


Figure 3-6: Kullback-Leibler Distance Training Algorithm

4. Estimate the GGD parameters  $\alpha_i^b$  and  $\beta_i^b$ ,  $b = 1, 2, \dots, B$ , for the  $B$  sets of coefficients for each test point  $\mathbf{x}_i$
5. Calculate  $\text{KLD}(\mathbf{x}_i, +1)$  and  $\text{KLD}(\mathbf{x}_i, -1)$  as defined in Equations 2.34 and 2.35 in Section 2.4.2, using the trained KLD parameters for each class from the training algorithm, for each test point  $\mathbf{x}_i$

6. Evaluate

$$h(\mathbf{x}_i) = \text{KLD}(\mathbf{x}_i, -1) - \text{KLD}(\mathbf{x}_i, +1) \quad (3.3)$$

and

$$f(\mathbf{x}_i) = \text{sign}(\text{KLD}(\mathbf{x}_i, -1) - \text{KLD}(\mathbf{x}_i, +1)) \quad (3.4)$$

for each test point  $\mathbf{x}_i$

7. Calculate the percentage of misclassified test points; that is, points for which  $f(\mathbf{x}_i) \neq y_i$  where  $y_i$  is the true class of the test point  $\mathbf{x}_i$
8. Plot a *binary reclassified image* where pixels  $\mathbf{x}_i$  for which  $f(\mathbf{x}_i) = +1$  are plotted white and pixels  $\mathbf{x}_i$  for which  $f(\mathbf{x}_i) = -1$  are plotted black

For our implementation of the KLD classification method, we have modified the MATLAB code written by Minh Do for the original task of textural image classification for our task of textural image segmentation [11].

### 3.2.3 Parameter Tuning

A quick study of the training and test algorithms above suggests the presence of various parameters that we need to tune in order to fully specify the algorithms. We first note common parameter choices in past work in order to limit the number of settings we consider for each parameter. We then choose the specific settings for each parameter by experimenting on two independent *validation image pairs*. These pairs, drawn from the

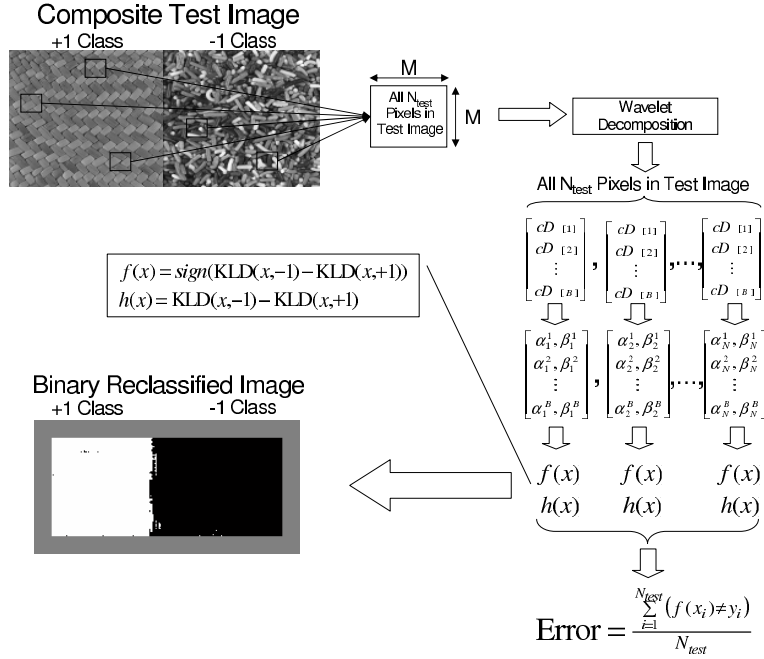


Figure 3-7: Kullback-Leibler Distance Test Algorithm

well-known Brodatz texture database, are shown in Figure 3-8 [4]. We employ the SVM algorithm in the parameter tuning process and apply the resulting optimal settings to both the SVM and KLD methods in the benchmark analysis.

The parameters that need tuning include the size  $M$  of the square  $M \times M$  local windows on which we define the textural features, the decomposition level  $n$  of the wavelet analysis and the kernel type and cost parameter  $C$  of the SVM classifier. We do not choose the number  $N$  of training points (windows) per class in this validation process. Instead, it is one of the main goals of this chapter to explore the performance effects of training the SVM and KLD classifiers on different amounts of training data; thus, we experiment on the benchmark test data with various  $N$  values in Section 3.3.2 to glean

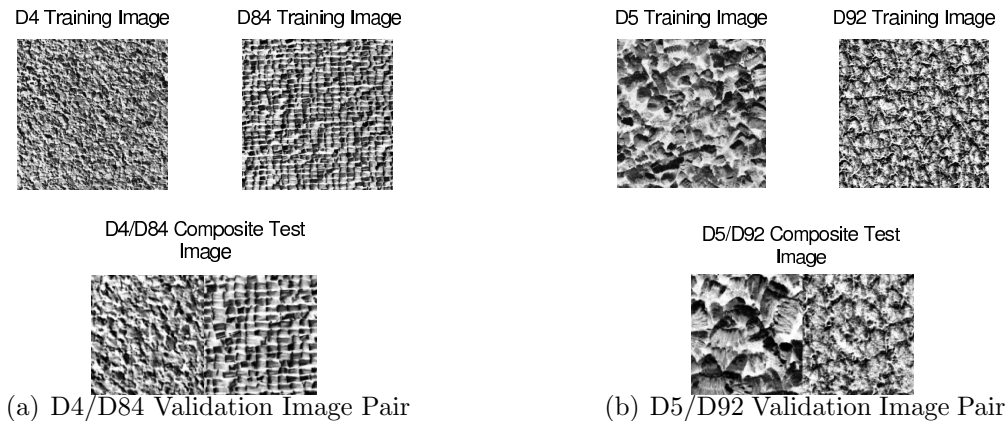


Figure 3-8: Brodatz Validation Image Pairs

Parameter Settings	Error Rate
$M = 35, n = 2$	3.81%
$M = 35, n = 3$	13.1%
$M = 65, n = 2$	3.41%
$M = 65, n = 3$	11.1%

Table 3.3: D4/D84 Validation Image Pair Results, Polynomial Kernel (degree 3)

this information. Additionally, as mentioned in Section 2.2.4, we have fixed the type of wavelet used throughout the analysis to the Daubechies 4 wavelet.

The choices of the window size  $M$  and wavelet decomposition level  $n$  are related. As shown in Figure 2-9 in Section 2.2.4, as the wavelet decomposition proceeds, the wavelet transform is applied to subimages of decreasing size. Since we do not want to apply the transform to subimages that are too small to capture the textural qualities of the full image, we must tailor the decomposition level to the window size. Window sizes used in the literature range from  $17 \times 17$  [25] to  $128 \times 128$  [11]. Wavelet decomposition levels of 2 and 3 are commonly found in the literature [11] [39] [56]. We experiment on the validation images with window sizes of  $35 \times 35$  and  $65 \times 65$  using 2- and 3-level wavelet decompositions.

The choice of window size also affects the ability of the classifier to distinguish the border between adjacent texture classes in the composite test image. We find that smaller windows better distinguish the border at the expense of performance on the interior textural regions. Larger windows achieve almost perfect classification on the interior textural regions but suffer in the border region. One solution to this disparity is to employ an adaptive scheme where a large window is used in the interior and a smaller window at the border. When we implement this method, however, we find that the additional computational burden of considering multiple window sizes overrides the small gains in classification performance. Thus, continuing to use the overall error rate across the entire composite test image with a fixed window size seems the most natural way to account equally for both interior and border errors.

To choose the best kernel and its associated parameters, we must rely solely on experimental results since we did not find any published work employing the SVM with wavelet energy features for textural image segmentation that could guide our selections. We experiment with the polynomial kernel (Equation 2.23 in Section 2.3.2) of degrees 3 and 5. We employ the default value of  $C$  determined automatically by the SVM<sup>light</sup> solver<sup>2</sup>.

Table 3.3 shows the error rates on the D4/D84 validation image pair using a polynomial kernel of degree 3. Table 3.4 shows the same results on the D5/D92 validation image pair. Using a polynomial kernel of degree 5 did not significantly reduce the error rates.

We see clearly that the 3-level decomposition suffers for both window sizes and both validation pairs. For the 2-level decomposition on the D4/D84 pair, the  $35 \times 35$  window

---

<sup>2</sup>Joachims uses the equation  $C = \left( \frac{1}{N} \sum_{i=1}^N \sqrt{K(\mathbf{x}_i, \mathbf{x}_i)} \right)^{-2}$ , where  $\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\}$  is the set of training data, to calculate the default  $C$  value.

Parameter Settings	Error Rate
$M = 35, n = 2$	8.1%
$M = 35, n = 3$	10.85%
$M = 65, n = 2$	9.1%
$M = 65, n = 3$	14.1%

Table 3.4: D5/D92 Validation Image Pair Results, Polynomial Kernel (degree 3)

performs only 0.4% worse than the  $65 \times 65$  window. However, for the D5/D92 pair, we see a full percentage point *improvement* using the  $35 \times 35$  window over the  $65 \times 65$  window. Thus, we choose to use the 2-level decomposition on  $35 \times 35$  windows for the full SVM and KLD benchmark analyses. For the SVM analysis, we use the polynomial kernel of degree 5. Although the degree 5 polynomial kernel does not show significant improvement over the degree 3 kernel for the validation images, its use does not add much additional computational cost and leaves open the possibility that it might improve performance on the full set of 18 benchmark image pairs.

To summarize, all results presented in Section 3.3 use the following parameter settings:

- Daubechies 4 wavelet
- 2-level wavelet decomposition
- $35 \times 35$  windows
- Polynomial kernel of degree 5 (for the SVM method)
- SVM<sup>light</sup> default  $C$  value (for the SVM method)

### 3.3 Benchmark Results

Now that the SVM and KLD algorithms are fully specified, we can report the various results we have achieved applying those algorithms to the 18 benchmark image pairs. The first and most important set of results reports the low error rates we have achieved with our new SVM classification scheme and our adaptation of the KLD scheme. This success gives us confidence in applying these methods to the stem cell application to follow. We also explore how the amount of training data influences the performance of the SVM and KLD methods and discuss the statistical reasoning behind this result.

#### 3.3.1 Basic Results

To establish the basic performance of the SVM and KLD classification schemes, we test all 18 composite test image pairs using two different training set sizes. Since the use of the support vector machine with wavelet energy features represents a new contribution to the field, we can only conjecture that this new method will perform well based on the excellent performance of the SVM in numerous other applications and of the wavelet

Test Image Pair	SVM Error Rate	KLD Error Rate
P1	5.1%	6.6%
P2	3.0%	3.4%
P3	3.1%	3.7%
P4	1.5%	4.7%
P5	25.4%	12.1%
P6	6.2%	7.2%
P7	7.7%	2.4%
P8	4.9%	7.7%
P9	4.2%	7.5%
P10	2.7%	6.3%
P11	5.1%	4.0%
P12	2.0%	6.9%
P13	1.7%	6.7%
P14	3.0%	2.2%
P15	3.3%	1.5%
P16	51.5%	51.2%
P17	4.8%	6.6%
P18	2.9%	6.5%
Average	7.67%	8.18%

Table 3.5: Basic Benchmark Results (100 training points per class)

energy features in the various texture classification studies outlined in Section 2.2.4. Additionally, we conjecture that the adapted KLD method will perform well based on its excellent performance for the original task of image-wise classification, as demonstrated by Vetterli and Do [11].

### 100 Training Points Per Class

Table 3.5 shows the error rates on each image pair and an average error rate over all the pairs using 100 training points per class for both the SVM and KLD methods.

Overall, we observe good performance for both the SVM and KLD methods. With the exception of pairs P5 and P16, which are difficult to segment for both methods, no error rate exceeds 7.7%. A paired  $t$ -test of the equivalence of the mean performance for the SVM and KLD methods over the sample of 18 image pairs (7.67% and 8.18% respectively) shows that the difference in means is not significant at the 0.05 level. Nevertheless, the SVM method achieves a lower error rate than the KLD method on 12 of the 18 pairs.

### 50 Training Points Per Class

Table 3.6 shows the error rates on each image pair and an average error rate over all the pairs using 50 training points per class for both the SVM and KLD methods.



Test Image Pair	SVM Error Rate	KLD Error Rate
P1	5.1%	6.5%
P2	2.9%	3.4%
P3	2.8%	3.7%
P4	1.6%	4.8%
P5	25.9%	12.6%
P6	5.5%	7.1%
P7	10.7%	3.2%
P8	4.2%	7.6%
P9	4.0%	7.5%
P10	3.8%	6.4%
P11	5.0%	4.3%
P12	1.7%	6.9%
P13	2.4%	6.6%
P14	2.2%	2.6%
P15	4.1%	1.6%
P16	51.3%	51.1%
P17	19.4%	6.1%
P18	3.1%	6.7%
Average	8.65%	8.26%

Table 3.6: Basic Benchmark Results (50 training points per class)

We note here some performance effects of using only 50 training points per class compared with using 100 training points as above. We see an average performance drop of 0.98% for the SVM method and 0.08% for the KLD method. We note informally that the SVM method suffers more from limited training data than does the KLD method. We hope to explore this observation more in Section 3.3.2 when we explicitly consider training set size effects.

Using only 50 training points per class also results in the KLD method outperforming the SVM method as measured by the average performance over all 18 image pairs. However, a paired  $t$ -test shows that this difference is not significant at the 0.05 level. Additionally, as with 100 training points per class, the SVM method yields a lower error rate than the KLD method on 12 of the 18 pairs.

## Conclusions

Overall, given sufficient training data (i.e., 100 training points per class) both the SVM and KLD segmentation schemes perform well on 17 of the 18 image pairs, with pair P16 as the exception. The average error rates over all 18 pairs are 7.67% and 8.18% for the SVM and KLD methods, respectively. If we exclude pair P16, the error rates drop to 5.1% and 5.6%. Paired  $t$ -tests show that the average SVM and KLD performances over all 18 image pairs are not statistically different using both 100 and 50 training points per class.

Overall, we are confident in the viability of both the SVM and KLD wavelet-based methods for the basic task of binary textural image segmentation. As discussed in Section 2.2.3, wavelet analysis is a sophisticated multiresolution image processing tool that can accurately capture both the spatial and frequency characteristics of a textural image. This improves upon the many feature representations, such as Haralick’s co-occurrence method and the various structural and model-based techniques, that consider only spatial characteristics. Additionally, within the frequency domain, the adaptive windowing technique of wavelet analysis accurately extracts both the high and low frequency components of the image. This represents an improvement over the single-resolution Gabor filter method.

We outlined in Section 2.3.2 the many features of the support vector machine classifier that have made it generally superior to other classification methods. Additionally, Vetterli and Do have shown the usefulness of the parametric Kullback-Leibler distance approach for modelling the textural characteristics of images [11].

These benefits of our chosen feature representation and classification methods are also manifested in empirical results. While we did not find in the literature any studies of the performance of other classification schemes using VisTex image pairs, our error rates of 5.1% and 5.6% using the wavelet-based SVM and KLD methods compare favorably with error rates reported across different image pairs using various other classification methods and feature representations. For example, Randen and Husøy report average error rates of 5.4%, 3.3% and 7.8% using autoregressive (AR), co-occurrence and Gabor filter features, respectively, with the LVQ classifier<sup>3</sup> on three image pairs drawn from the

---

<sup>3</sup>The learning vector quantization (LVQ) classifier, proposed by Kohonen in 1989, is a prototype-based classifier. The algorithm clusters the training data into a number of classes and represents those training

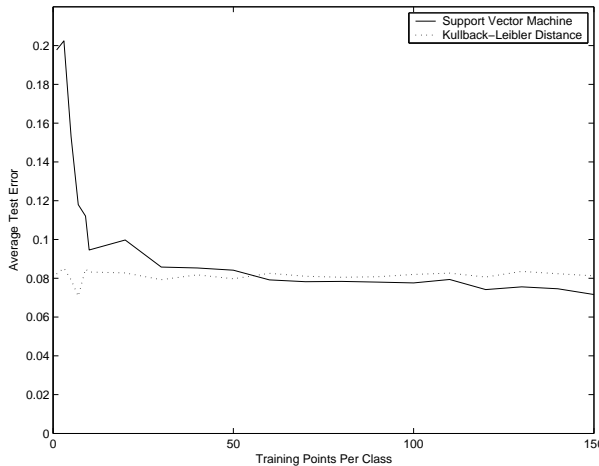


Figure 3-9: Training Set Size Effects

Brodatz texture database [39].

### 3.3.2 Training Set Size Results

We observed informally above that the SVM method suffers more than the KLD method when trained on a small data set. However, with sufficient training data, we observed that the SVM method outperforms the KLD method in terms of the average error rate across all 18 image pairs. While this difference in performance is not statistically significant, we would like to explore these potential training set size effects in a bit more detail.

For this analysis, we simply train the SVM and KLD methods using training sets of various sizes ranging from 1 to 150 points per class. We then calculate the average error rate of each method over all 18 image pairs for each training set size. Figure 3-9 shows the resulting average error rates as a function of training set size.

The plot confirms our initial observations from Section 3.3.1. We see that given only a few training points per class, the SVM method suffers significantly more than the KLD method. However, with more training points it quickly improves, eventually surpassing the KLD classifier at about 60 training points per class. The addition of even more training points does not significantly reduce the error rates for either method; however, the SVM method shows consistently better performance than the KLD method in the presence of a large amount of training data.

These results make sense in the statistical context of the SVM and KLD classifiers. The parametric KLD method makes the assumption that the wavelet coefficients at each subband follow a probabilistic distribution. Given only one training sample, the KLD method can estimate the parameters of that distribution and produce a complete and fully specified model of the coefficient distribution. The non-parametric SVM method, on the other hand, does not have the luxury (or curse) of making such a probabilistic assumption. The absolute mean and variance extracted at each subband yield some indication of the coefficient distribution; however, they do not specify this distribution as clusters with prototype points. New inputs are then classified to the class of the nearest prototype [20].

completely as the full model in the KLD approach. Thus, more training data is needed to achieve a viable model of the coefficient distribution.

We conclude that the SVM method is sensitive to the amount of training data, up to about 60 training points per class. The KLD method shows consistent performance regardless of the amount of available training data. Thus, if only a few training points per class are available, the KLD method is the recommended approach. However, if there is ample training data, the SVM method is generally superior.

### 3.3.3 Conclusions

In considering the various results of the benchmark analysis, we first observe that the support vector machine using wavelet energy features shows sufficiently low error rates on a wide range of textural images to conclude that it is a useful method of textural image segmentation. In addition, we have shown that Vetterli and Do's Kullback-Leibler distance classifier can be well-adapted to the task of image segmentation based on its good performance over a wide range of textural images.

In comparing the SVM and KLD methods, we do not find a statistically significant difference in their mean performances over the 18 benchmark image pairs using either 100 or 50 training points per class. We do observe that the SVM method suffers more than the KLD method with only a small amount of training data. However, with more than 50 training points per class, both methods show excellent performance with the SVM method consistently having a slightly lower error rate over the 18 image pairs.

The observations above regarding the differences between the SVM and KLD methods reflect common differences between non-parametric and parametric methods. If accurate parameter estimation can be achieved, then parametric methods can perform quite well on only a few training points since their assumption of a probabilistic distribution can fill the gaps left by the incomplete data. However, as in the KLD method here, numerical parameter estimation can take some time if closed-form expressions do not exist. In contrast, non-parametric methods are fully dependent on data to specify the model and thus do not perform as well with small training sets. However, with sufficient training data, they can equal and exceed parametric methods. While non-parametric methods may not be extremely fast, if they eliminate the need for time-consuming numerical parameter estimation, they can significantly reduce computational time as well.

# Chapter 4

## Stem Cell Application: Qualitative Interior Textural Segmentation

Now that we have laid the theoretical and practical foundations for the use of both the non-parametric support vector machine classifier with wavelet energy features and the parametric Kullback-Leibler distance classifier to characterize texture, we turn to creating a computer aid based on these methods for the characterization and categorization of stem cell colonies. This chapter discusses the medical background of stem cells, outlines our two-tiered approach to the problem and explores the textural segmentation problem that forms the first-tier analysis. This analysis yields both qualitative stand-alone results that are intended as graphical aids to stem cell researchers and the means of creating a set of image-wise features that will be used for the second-tier image categorization analysis of Chapter 5.

### 4.1 Medical and Biological Motivation

Before delving into the technical aspects of the problem, we use this section to first shed some light on the medical and biological issues involved. We also outline the specific contributions and scope of the research we will present, making clear the specific intent and potential of the work<sup>1</sup>.

#### 4.1.1 Stem Cell Background

The successful culture of human embryonic stem cells at the University of Wisconsin in 1998 opened the door for serious research into the potential medical benefits of these unique cells. These medical benefits derive from three special properties of stem cells not shared by normal human cells [33]:

1. Stem cells can divide and renew themselves for long periods of time

---

<sup>1</sup>This work has been highly motivated by Dr. Paul Sammak of the Pittsburgh Development Center of the Magee-Womens Research Institute at the University of Pittsburgh. He has provided all the stem cell images used in our work and has given us the expert medical insight needed to tackle this problem.

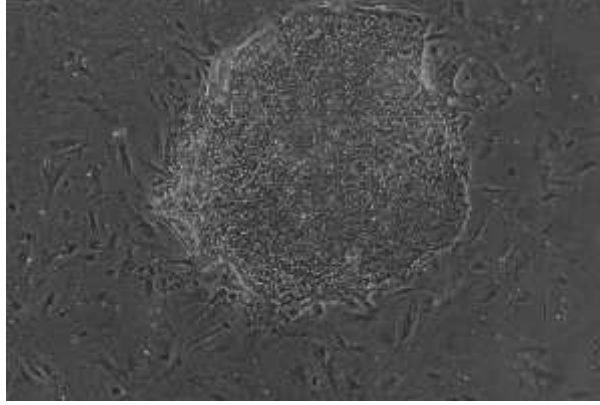


Figure 4-1: Typical Stem Cell Colony

2. Stem cells are unspecialized
3. Stem cells can be induced to take on the specialized function of any normal human cell

Ideally, these three properties would allow stem cells to remain *undifferentiated* for a long period of time and then, when needed, be induced to *differentiate* into the type of human cell necessary for a specific medical treatment. For example, transplantation therapy could be given a patient suffering from type 1 diabetes, where the patient's insulin-creating cells in the pancreas have been destroyed by the body's immune system. Available stem cells would be induced to differentiate into insulin-producing cells that would then be injected into the patient's pancreas, curing the disease [33]. Another application would involve testing exploratory drugs safely on differentiated stem cells rather than on live human subjects [33].

The first step in realizing these medical benefits involves successfully creating and maintaining healthy, undifferentiated stem cells in a laboratory environment. This process, known as *cell culture*, begins by transferring the approximately 30 inner cells of a three- to five-day-old embryo onto a culture dish where these cells begin to proliferate. As they proliferate, the stem cells become clustered into *stem cell colonies*. Then, as these colonies begin to crowd the culture dish, healthy regions of the colonies are transplanted to other dishes for further proliferation. This process of *subculturing* is continued for six or more months until a healthy, undifferentiated, *stem cell line* is created and becomes available for research purposes [33].

Figure 4-1 shows a typical stem cell colony. The clear structure in the middle of the image is the colony itself, with the individual stem cells making up its interior. Surrounding the colony are feeder cells and other culture fluids.

In order for stem cells to be viable for research, they must remain genetically normal and undifferentiated throughout the culturing process. To ensure this normality, the stem cell colonies undergo a continuous quality control process where they are monitored and tested for signs of degradation and differentiation. It is this monitoring process that our research advances. We discuss below the current state of the art of this quality control process as motivation for the contributions of our research.

### 4.1.2 Research Contributions

The core step in the stem cell quality control process is the determination of each individual colony's health. Only those colonies deemed sufficiently healthy are maintained in culture. Presently, two approaches exist for the determination of colony health, each with significant drawbacks.

The first method involves visual inspection by an experienced stem cell microscopist. While such an expert can quickly recognize a good colony, a working laboratory might have up to 12,000 colonies in culture that need to be inspected every two days. Furthermore, in future therapeutic uses of stem cells, a ten-fold increase in the number of colonies requiring inspection would be expected. With so many colonies to be inspected, it quickly becomes infeasible for researchers to directly inspect all of their colonies. Additionally, visual analysis naturally introduces the biases of individual researchers into the quality control process.

The second approach to colony health determination involves the use of invasive biological tests that either destroy the colonies or render them useless for research purposes. For example, tests such as immunocytochemistry, gene arrays and PCR (tests for mRNA characteristics of stem cells) are destructive tests that do not allow for continued colony growth. Even a non-destructive test such as flow cytometry is very harsh and can result in the death of a majority of the stem cells in a colony. At present, there is no routine, non-invasive method for measuring stem cell quality under conditions favorable to their growth.

The drawbacks of these two current methods suggest that a standardized, non-invasive, automated approach to stem cell colony inspection would be a useful contribution to the stem cell research community. Toward that end, we propose a semi-automated, computer-implemented approach to providing stem cell researchers with standard qualitative and quantitative measures of stem cell quality. We do so in a non-invasive manner by analyzing digital images of stem cell colonies based on specific visual characteristics. We outline our procedure in more detail in Section 4.1.4 below. However, since our work is based on visual characteristics, we first discuss the specific visual criteria of stem cell colony quality.

### 4.1.3 Stem Cell Visual Quality Criteria

Focusing on the *visual* characteristics of stem cell colony health, we learn that there are three criteria of a good stem cell colony:

1. Textural homogeneity
2. Textural tightness
3. Border sharpness and circularity

Though the first two criteria are independent factors, together they define the *textural* characteristics of a good stem cell colony. *Homogeneity* refers to the uniformity of the textural qualities across the colony. This factor is important because heterogeneous areas of the colony may not be suitable for subculturing as these regions are showing signs

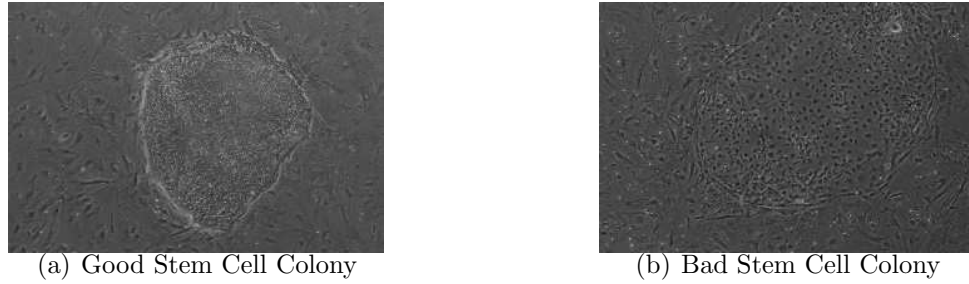


Figure 4-2: Good and Bad Stem Cell Colonies

of cellular differentiation. However, this criterion alone does not discriminate among the various types of uniform texture. For example, a colony could exhibit uniformly loose texture and still satisfy the homogeneity criterion. The second criterion of *textural tightness* dictates the specific type of texture that constitutes a good colony; namely, a good colony consists of small cells, densely packed together to reveal a tight texture. Jointly, these first two criteria state that a good stem cell colony exhibits a homogeneous tight texture throughout the colony.

The third criterion is based on the *border* characteristics of a stem cell colony. A good stem cell colony exhibits sharply defined, highly circular borders. Moving radially out from the centroid of a good colony should reveal a sharp gradient at the colony border, while traversing rotationally along the border should expose few inflection points or deviations from circularity.

Figure 4-2 shows examples of a good and bad stem cell colony. Notice that the good colony on the left exhibits a tight, homogeneous texture with a clear, generally circular border. In contrast, the bad colony on right has a rather heterogeneous textural interior consisting of very loose, broad cells. In addition, its border, while generally circular, is not sharp.

#### 4.1.4 Research Scope: A Two-Tiered Approach

As noted in Section 4.1.2 above, our research yields a semi-automated approach to providing stem cell researchers with both qualitative and quantitative measures of stem cell quality. We partition our work into two tiers based both on the type of output produced by each tier and the specific visual colony quality criteria employed to create that output. Tier 1 focuses on the first two *textural* criteria and yields texture-based qualitative graphical aids aimed at reducing the time spent by stem cell researchers in visually examining their colonies. Tier 2 then combines the texture analysis from Tier 1 with its own *border* criterion analysis to provide standard quantitative measures of colony quality.

To better motivate the usefulness of these two tools, we must first understand more fully what the medical researchers are looking for when judging their colonies. As a stem cell colony grows, some of the cells within the colony interior may differentiate while other cells will remain healthy and undifferentiated. The researchers must be able to identify the healthy regions of such a colony in order to extract cells from those regions for further subculturing. Visually examining each colony for these healthy regions can be a time



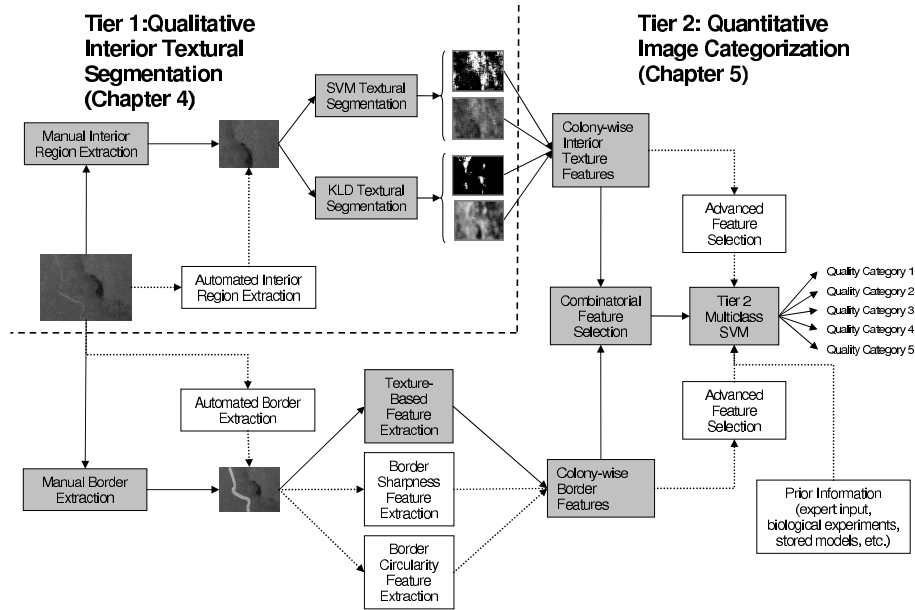


Figure 4-3: Two Tiered Stem Cell Analysis Methodology

consuming process for researchers without any guide as to where to focus their attention. A useful time-saving aid would involve providing a graphical representation that segments each colony into good and bad regions based on the interior colony texture; this visual output would highlight the areas of the colony that deserve more visual attention. This is precisely what the Tier 1, or *qualitative interior textural segmentation* analysis of this chapter achieves.

Additionally, a numerical rating that characterizes the overall health of the entire colony would be beneficial in order to highlight those colonies that show potential for further subculturing and those that clearly do not. This standard output would also mitigate against individual researcher bias in judging the quality of a set of stem cell colonies. This Tier 2, or *quantitative image categorization* output is the focus of Chapter 5.

Note that while Tier 1 focuses on the qualitative analysis of interior texture quality, Tier 2 uses interior textural quality features—derived from the graphical Tier 1 outputs—and a new set of border quality features to categorize the entire stem cell colony in terms of its overall health and level of differentiation. This represents a hierarchical, two-tiered approach where the Tier 2 analysis uses and builds upon the Tier 1 outputs. Figure 4-3 illustrates the architecture of this two-tiered method and shows the interaction between the two tiers. The shaded blocks connected with solid lines represent the work accomplished in this thesis. The unshaded blocks connected with dotted lines illustrate areas of future work. More details on these future additions are outlined in Section 6.1.

While the main goal of our research is to aid medical researchers in the task of visual colony quality control, we also propose an interesting future biological application of this research. Stem cell researchers perform biological experiments on their colonies to learn about the effects of certain treatments and procedures. To quantify their results, it would be useful to have a standard means of measuring the success or failure of the different treatments and procedures performed on the colonies. For example, researchers would take

a baseline measure of stem cell health before subjecting the colony to various experiments. They would then take new measures to determine whether the experiments improved or degraded the health of the colony. The Tier 2 ideas presented in Chapter 5 could be extended to create the standard, quantifiable measure by which researchers would judge the results of their biological experiments.

## 4.2 Interior Textural Segmentation Methodology

In order to begin the implementation of the two-tiered approach proposed in Section 4.1.4, we must first consider the many algorithmic details of the SVM and KLD methods we have chosen to use. The most important of these details is the actual data itself. For our work, this data consists of 77 stem cell colony digital images. The images themselves warrant some attention, as do the preprocessing steps we perform to set up the Tier 1 analysis. We follow this discussion by describing the training data we use to train the SVM and KLD classifiers. We conclude this section by specifying the various other parameter settings of the two classifiers. Overall, the ideas presented here build on those of Chapter 3, and we reference that chapter for more detailed explanations of the SVM and KLD classification methods that we employ.

### 4.2.1 Description of Images

For our analysis, we have been provided 77 images of stem cell colonies of varying quality. The colonies are approved National Institutes of Health (NIH) human embryonic stem cells, line HSF-6 from the University of San Francisco. The images were acquired on a Nikon TMS inverted tissue culture microscope with a 10x phase contrast objective and a Nikon D100 digital camera. Each image is in RGB format of size  $1000 \times 1504$  pixels. Figure 4-2 shows grayscale versions of two of these 77 images.

The stem cell colonies are clearly defined in some images. However, in others, it is not as easy to determine where the colony of interest is located. Some images contain multiple colonies whose boundaries are often overlapping and melding. Others contain colonies that are cut off by the image boundaries. However, following the guidance of an expert in the stem cell research community, we are able to identify the true colony of interest in each image for use in our analysis.

Another potential problem with the raw images is their sheer size. As described in Chapter 3, textural image segmentation requires the analysis of the texture surrounding *each* pixel in the image to be segmented. Since there are about 1.5 million pixels in a  $1000 \times 1504$  image, even our segmentation of a subset of the pixels—namely, the interior region of the colony—requires many intensive computations. We discuss ways to speed up the process in Section 4.2.4 below.

### 4.2.2 Image Preprocessing

We perform the same image preprocessing steps outlined in Section 3.1.2 on each of the 77 stem cell images. Specifically, we convert each RGB image to grayscale via the MATLAB

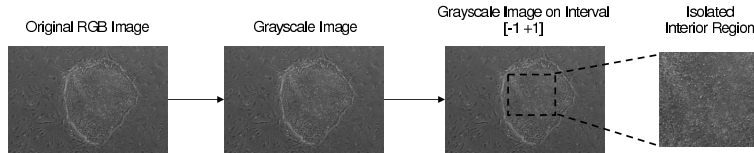


Figure 4-4: Tier 1 Image Preprocessing

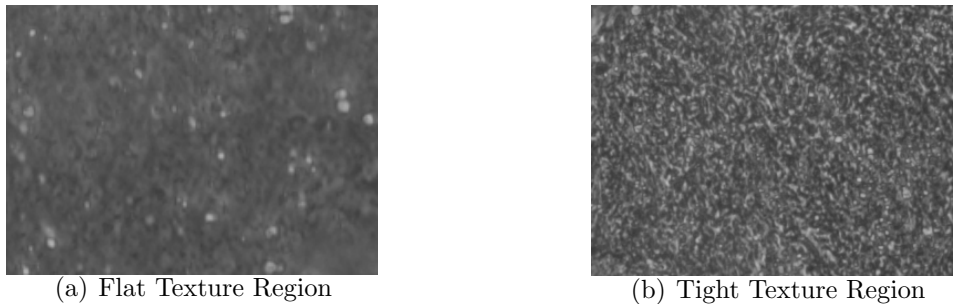


Figure 4-5: Two Types of Good Stem Cell Texture

function `rgb2gray` and scale the resulting 256 grayscale intensity levels to the interval  $[-1\ 1]$  for computational stability.

Additionally, since the Tier 1 analysis performs textural segmentation strictly on the interior regions of the stem cell colonies, we must extract these interior regions from each colony. We do so by identifying the largest inscribed rectangle lying clearly and fully within the colony interior, as illustrated in Figure 4-4. We define this rectangular subimage as the interior region used in the Tier 1 analysis. For the remainder of this chapter, general references to the stem cell colony images refer specifically to this interior textural region.

### 4.2.3 Training Data Extraction

Because our goal in the Tier 1 analysis is to segment each stem cell colony into healthy and poor regions based on the colony texture, we need to provide training data representing good and bad texture to the SVM and KLD classification methods. This task is complicated by the fact that there are two types of good texture. Similarly, there exist two types of bad texture.

*Flat texture* defines healthy regions of the colony where the individual stem cells are packed together so tightly as to be individually unrecognizable, giving the regions a flat appearance. The *tight texture* regions also signify healthy portions of the colony where individual cells are recognizable, yielding a tight textural quality. Figure 4-5 shows examples of these two types of good texture.

The two types of bad texture we encounter are the *black texture* and *loose texture* regions, shown in Figure 4-6. The black texture regions occur when the individual stem cells pile up on top of each other and appear as a black splotch under the microscope and in the resulting images. The individual cells in the loose texture regions are not tightly packed together and are generally broad, giving the region its loose textural quality.

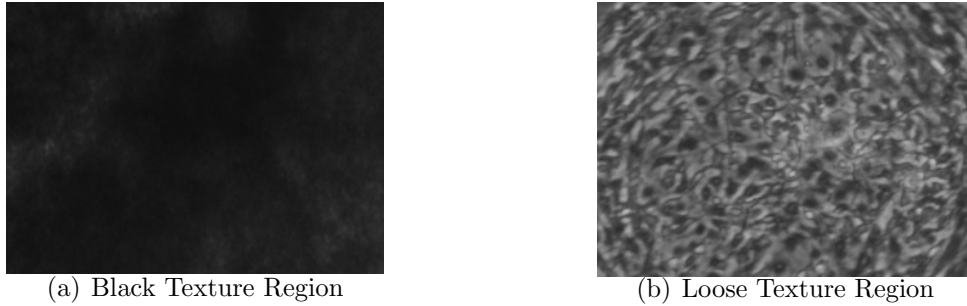


Figure 4-6: Two Types of Bad Stem Cell Texture

In Chapter 3 we created the training sets by randomly extracting  $N$  windows from each of the two textural classes under consideration for each run. This random selection was necessary because in each run we were comparing a different combination of textures. For our stem cell application, however, we are always comparing good and bad stem cell texture. Thus, instead of randomly extracting training data for each run, we create standard training sets representing each of the four types of texture defined above; we use these training sets for all the runs in the Tier 1 analysis. We outline this approach in the following bullets:

1. Identify four different colonies from among the 77 available colonies each containing sufficiently large regions (approximately  $300 \times 300$  pixels) within them that exhibit the *flat texture* depicted in Figure 4-5(a).
2. Randomly extract 50 windows of size  $55 \times 55$  from each of the four different regions identified in Step 1, yielding  $50 \cdot 4 = 200$  windows of size  $55 \times 55$  exhibiting flat texture.
3. Perform a wavelet decomposition on each of these 200 flat texture windows.
  - For use with the SVM classifier, take the mean deviation and variance of the wavelet coefficients at each subband for each of the 200 wavelet decompositions, as described in Section 2.2.4. This yields a wavelet energy feature vector for each of the 200 flat texture windows that can be used to train the SVM classifier.
  - For use with the KLD classifier, estimate the generalized Gaussian distribution parameters of the wavelet coefficients at each subband for each of the 200 wavelet decompositions. This yields a set of GGD parameters for each of the 200 flat texture windows that can be used to train the KLD classifier.
4. Define the *flat texture training set* to consist of these 200 wavelet energy feature vectors and GGD parameter sets.
5. Repeat Steps 1 through 4 using regions exhibiting the *tight texture* depicted in Figure 4-5(b). This results in a *tight texture training set* consisting of 200 wavelet

Training Set Number	Paired Texture Types
1	flat texture + black texture
2	flat texture + loose texture
3	flat texture + composite bad
4	tight texture + black texture
5	tight texture + loose texture
6	tight texture + composite bad
7	composite good + black texture
8	composite good + loose texture
9	composite good + composite bad

Table 4.1: Nine Tier 1 Training Sets

energy feature vectors and GGD parameter sets calculated from the tight texture regions.

6. Create a *composite good texture training set* by combining 100 random samples from the flat texture training set of Step 4 and 100 random samples from the tight texture training set of Step 5.
7. Steps 1 through 6 result in the creation of three good texture training sets, each with 200 training points.
8. Repeat Steps 1 through 7 replacing the two types of good texture with the two types of bad texture depicted in Figure 4-6. This results in three bad texture training sets: *black texture training set*, *loose texture training set* and *composite bad training set*, each with 200 training points.
9. Repeat Steps 1 through 8 using windows of size  $45 \times 45$  and  $35 \times 35$  to create the various training sets for these two additional window sizes.

For a given window size, pairing each good texture training set with each bad texture training set yields nine complete training set pairs. Throughout the remainder of this thesis, we reference these nine Tier 1 training sets numerically as shown in Table 4.1.

#### 4.2.4 SVM and KLD Algorithmic Settings

For the Tier 1 analysis runs, we consider the three window sizes  $55 \times 55$ ,  $45 \times 45$  and  $35 \times 35$ . Additionally, for the SVM method, we use a polynomial kernel of degree 3 and SVM<sup>light</sup>'s default  $C$  value. As in Chapter 3, we use the Daubechies 4 wavelet with a 2-level decomposition.

The SVM and KLD methods are computationally expensive due to the wavelet decomposition, KLD parameter estimation and SVM classification steps. While the training algorithms perform these steps on only 400 textural windows, having to train the SVM and KLD classifiers each time we run the test algorithms introduces unnecessary computations. However, since we have defined the nine training sets a priori, we can train

the SVM and KLD classifiers only once for each training set and save the parameters that define the trained classifier. Then, before each test run, we need only load the saved classifiers rather than retrain.

In the SVM and KLD test algorithms, performing the wavelet decomposition, KLD parameter estimation and SVM classification steps on even a small subset—specifically, the extracted interior region—of the approximately 1.5 million pixels in the  $1000 \times 1504$  pixel colony images is a significant computational expense. To speed up this classification process, we classify only every fifth pixel in both the vertical and horizontal directions of the interior regions of each of the 77 images; the remaining pixels are assigned to the class of their nearest classified neighbor. This results in a lower-resolution reclassified image, but the computational speedup is significant.

### 4.3 Interior Textural Segmentation Results

Having motivated the need for and the technical details of the qualitative analysis of stem cell colony health, we are prepared to discuss the actual Tier 1 outputs that result from the analysis. We describe the various runs we perform, the graphical outputs we can create from those runs and the benefits we expect the stem cell community to gain from them. We conclude with a discussion of the effects that different parameter choices—specifically, the choice of classification method, training set and window size—have on the resulting graphical outputs.

#### 4.3.1 Runs Performed

To provide a rich set of qualitative outputs, we perform a number of runs on each of the 77 colony images. These numerous Tier 1 runs also yield many interior textural quality features that will be used for the task of quantitative image categorization in the Tier 2 analysis of Chapter 5.

For these runs, we employ both the SVM and KLD methods to test—or, more accurately, reclassify—each of the 77 preprocessed stem cell colony images using each of the nine training sets with each of the three window sizes. This results in  $2 \cdot 9 \cdot 3 = 54$  analyzes for each of the 77 colonies. Note, however, that some of the 77 colonies have contributed training data toward the creation of the nine training sets described in Section 4.2.3. In order to keep the training and test data separate, we disregard the analysis of any colony where a training set containing data contributed by that colony is employed.

As described in Sections 3.2.1 and 3.2.2, each analysis calculates a binary classification  $f(\mathbf{x}_i)$  and confidence value  $h(\mathbf{x}_i)$  for each pixel  $\mathbf{x}_i$  in the stem cell colony image being reclassified. We create the graphical outputs described in Section 4.3.2 below using these values calculated across the colony images.

#### 4.3.2 Reclassified Image Outputs

The SVM and KLD binary classifications and confidence values calculated across each stem cell colony image allow us to produce various graphical outputs that will assist stem

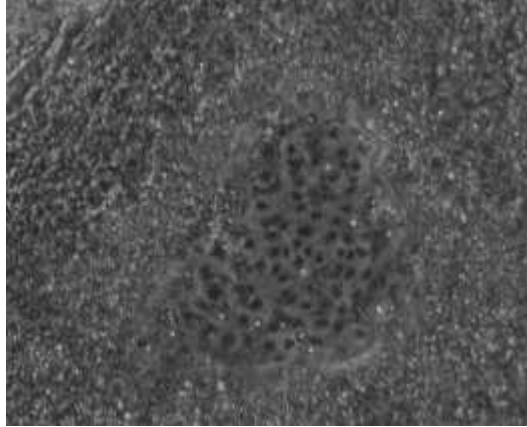


Figure 4-7: Tier 1 Graphical Outputs Example Colony

cell researchers in determining the health of their colonies. We propose four types of graphical outputs, described below.

These graphical outputs will naturally be affected by which of the two classification algorithms, nine training sets and three window sizes is used. We consider these parameter choice effects in Section 4.3.3. However, to illustrate the various Tier 1 graphical outputs that we can create, we consider the SVM and KLD analyzes of the colony shown in Figure 4-7 using training set 5 (tight texture + loose textured) with window size  $55 \times 55$ .

Examining the stem cell colony in Figure 4-7, we notice the loose texture in the center of the colony. This region of the colony is clearly differentiated. Immediately surrounding this center region is a ring of tight texture, followed by generally good, though not as tight, texture in the outer portion of the colony. We expect the various graphical outputs from our SVM and KLD segmentation algorithms to capture these features of this particular stem cell colony.

### Binary Reclassified Image

By plotting those pixels classified as good (pixels for which  $f(\mathbf{x}_i) = 1$ ) in white and those classified as bad (pixels for which  $f(\mathbf{x}_i) = -1$ ) in black, we can create a *binary reclassified image*. This output segments those areas of the colony interior that are healthy from those that are differentiated and unhealthy. Additionally, the amount of black, or poor, regions in the reclassified image provides a measure of the overall quality of the colony relative to all the other colonies being examined. Thus, the binary reclassified images serve as both a means of analyzing quality within an individual colony and across all the colonies under consideration.

Figure 4-8 shows the binary reclassified images of the stem cell colony shown in Figure 4-7 using both the SVM and KLD classification algorithms. The SVM and KLD methods both successfully segment the differentiated center region of the colony.

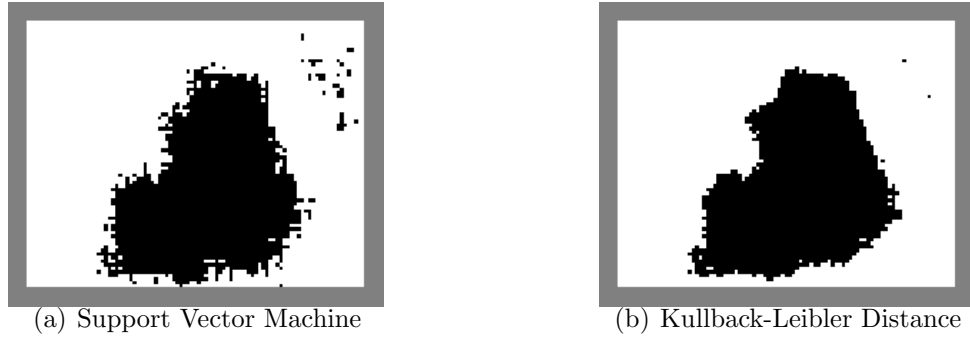


Figure 4-8: Binary Reclassified Images

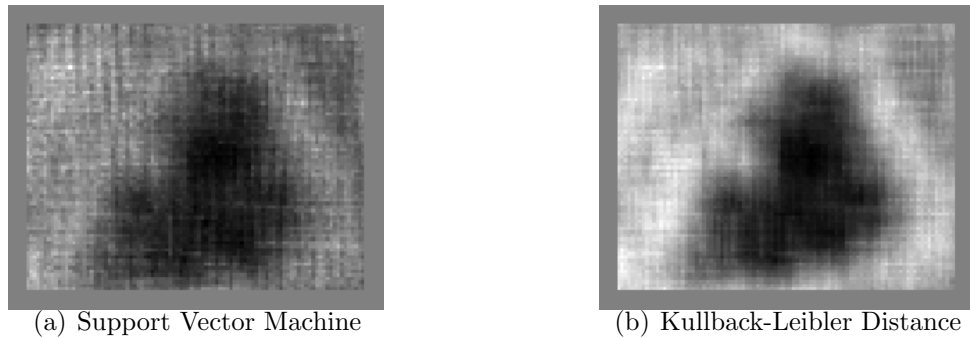


Figure 4-9: Locally Normalized Confidence Reclassified Images

### Locally Normalized Confidence Reclassified Image

Where the binary reclassified image displays the binary classifications  $f(\mathbf{x}_i)$  of each pixel, the *locally normalized confidence reclassified image* displays the confidence values  $h(\mathbf{x}_i)$ . While these confidence values are technically unbounded, for graphical display purposes we need to normalize them to the interval  $[0, 1]$ . We do this by examining the specific colony under consideration and assigning the lowest confidence in the image to 0, the highest confidence to 1 and the rest of the confidence values linearly to the  $[0, 1]$  interval. The resulting representation highlights the subtle differences in textural quality that exist across the colony. Since the normalization to the  $[0, 1]$  interval is local to the specific colony, locally normalized confidence reclassified images should not be used to compare textural quality among multiple different colonies.

Figure 4-9 shows this representation using both the SVM and KLD classification algorithms. As with the binary representation, these images clearly show the differentiated inner region. Additionally, the use of the more detailed confidence values highlights the difference in texture quality between the ring surrounding the center region and the outer portion of the colony. The lighter contrast of the ring compared to the outer region—shown in both images, though more clearly in the KLD representation—demonstrates the usefulness of using the confidence values to create a higher resolution graphical reclassification of the colony.



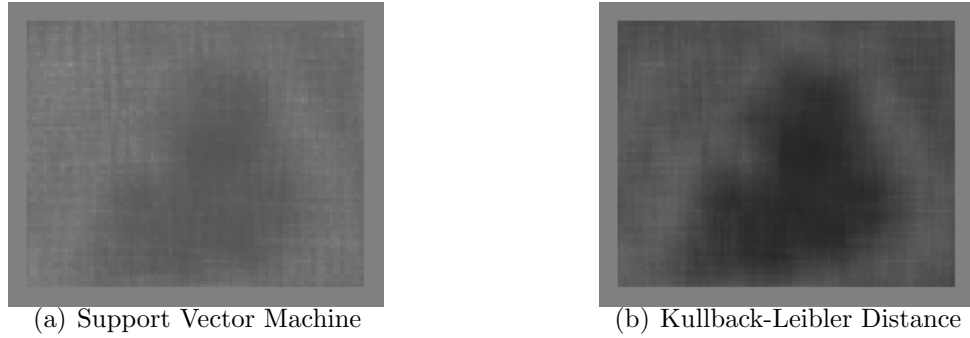


Figure 4-10: Globally Normalized Confidence Reclassified Images

### Globally Normalized Confidence Reclassified Image

A drawback of the locally normalized confidence reclassified images described above is their unsuitability in making comparisons of textural quality across multiple different colonies. To allow for this inter-colony comparison, we introduce *globally normalized confidence reclassified images*. For a given classification algorithm, training set and window size we find the lowest and highest confidence values across all 77 analyzed colonies. These global extreme points are assigned the values 0 and 1, respectively. Then the remaining confidence values in all the colonies are linearly scaled to the  $[0\ 1]$  interval. The result is a set of reclassified images that show less detail in each individual image but can be used to make textural quality comparisons among the different colonies.

Figure 4-10 shows the globally normalized reclassified representation using the SVM and KLD algorithms. Though we can still make out the textural variation within the colony, the global normalization smooths out the contrast. While not particularly useful when analyzing only one colony, this representation would be necessary for a comparison of the relative qualities of a collection of stem cell colonies.

### Quality Movie

The final graphical output we suggest consists of a series of images that shows the emergence of unhealthy regions of the stem cell colony. This *quality movie* is actually a simple dynamic representation of the locally normalized confidence reclassified image. The frames in the movie are created by imposing various thresholds (from 0 to 1) on the locally normalized confidence reclassified image; those confidence values falling above the threshold are plotted as white and those falling below as black. As the movie is played, the poorest regions of the colony will quickly appear in black, followed by regions of steadily improving quality.

Figure 4-11 shows 12 frames from a movie using the information from the KLD version of the locally normalized confidence reclassified image. The movie confirms our observations about the colony in Figure 4-7. The differentiated center region appears first, followed by the outer area; the ring of tight texture around the center region appears last, highlighting its good textural quality.

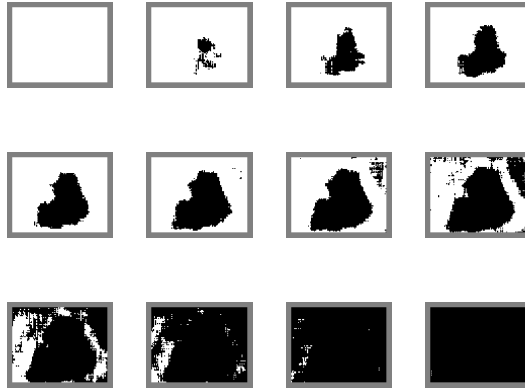


Figure 4-11: Quality Movie

### Two-Stage Qualitative Analysis

Since each of the four graphical outputs described above provides a different insight into the quality of a researcher's stem cell colonies, we propose a two-stage examination scheme to effectively use all the available information. The binary reclassified and globally normalized confidence reclassified representations of all the colonies under consideration should first be examined. Since these two representations allow for comparisons among different colonies, they can be used to identify the clearly healthy and unhealthy colonies. The healthy colonies can be immediately retained for further subculturing while the clearly unhealthy colonies can be disregarded. The remaining inconclusive colonies can then be analyzed further using their associated locally normalized confidence reclassified images and quality movies. This closer analysis will highlight the portions of these colonies that can be salvaged and used for subculturing or further biological experimentation.

### 4.3.3 Reclassified Image Comparisons

As mentioned above in Section 4.3.2, the choice of classification method, training set and window size in the Tier 1 analysis will affect the resulting graphical outputs. We consider the effects of these parameter choices by examining the graphical outputs obtained using various settings of the parameters on four prototypical stem cell colonies. The four colonies, shown and numerically labelled in Figure 4-12, represent a cross-section of the types of colonies stem cell researchers would likely encounter in their work. The colonies are labelled in order of decreasing quality with Colony 1 showing the highest overall quality and Colony 4 the lowest.

#### Classification Method Effects

To analyze the differences in the Tier 1 graphical outputs between the SVM and KLD methods, we consider the binary and locally normalized confidence reclassified representations of each of the four colonies in Figure 4-12. We obtain the outputs using training set 5 (tight texture + loose texture) and a  $55 \times 55$  window. Figures 4-13, 4-14, 4-15 and 4-16 show the outputs for each of the four colonies; the SVM method produces the top

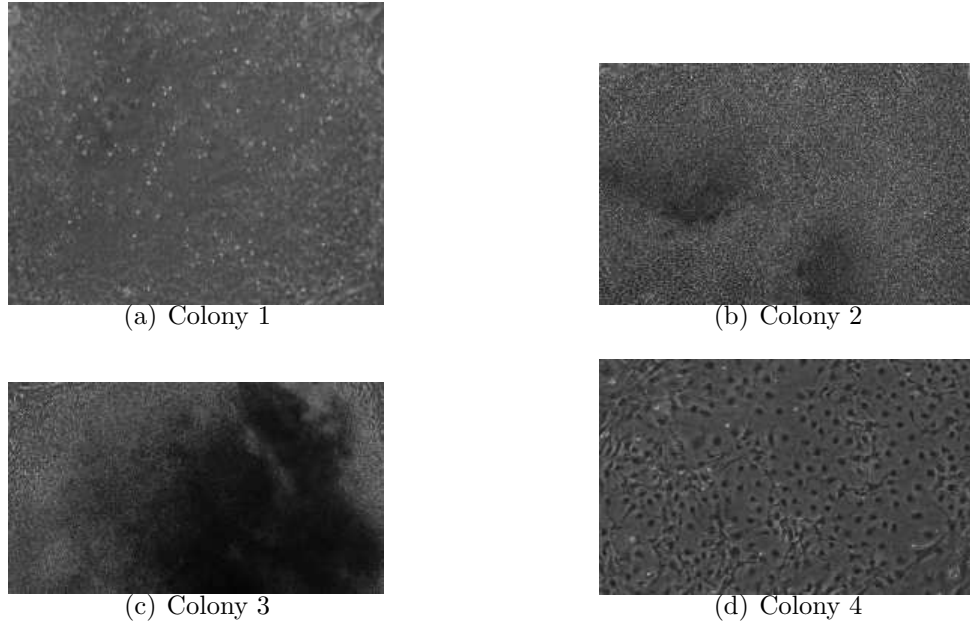


Figure 4-12: Stem Cell Colonies Used for Reclassified Image Comparison Study

two images of each figure, and the KLD method produces the bottom two.

Comparing the various SVM and KLD outputs across the four colonies, without making judgements as to the accuracy of those outputs, we notice little difference between the two methods. They both generally segment the colonies into the same good and bad regions. We do observe that the KLD method is more extreme in its segmentations; it produces clearer boundaries between good and bad regions in the binary images and shows a larger contrast in the grayscale values between good and bad regions in the confidence images. The SVM outputs tend to exhibit less-defined boundaries and a more even contrast in the binary and confidence reclassified images. So, while the two methods give commensurate qualitative Tier 1 outputs, the cautious reader might find the SVM method more agreeable.

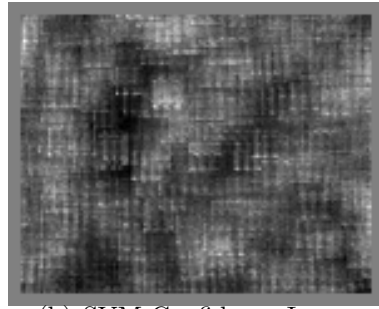
### Training Set Effects

In comparing the SVM and KLD outputs above, we neglected to discuss why Colony 1 was classified by both methods as showing almost completely poor texture quality when, in fact, it is the most healthy of the four colonies. We find that this is due to the use of training set 5 in the analysis that produced those reclassified images. This example highlights the potentially influential choice of Tier 1 training set. We consider this issue in more depth by examining the binary reclassified images for each of the four colonies using each of the nine training sets. We employ the SVM method and fix the window size at  $55 \times 55$ . Table 4.2 reviews the nine training sets according to their numerical labels and the types of good and bad texture they represent.

Figures 4-17, 4-18, 4-19 and 4-20 show the nine outputs for each of the four colonies. The original colony is shown at the top of each figure, and the SVM binary reclassified



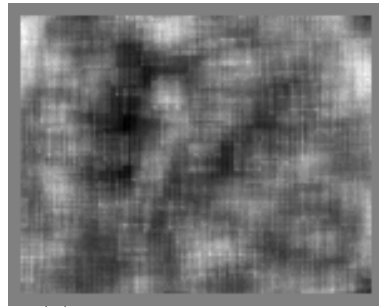
(a) SVM Binary Image



(b) SVM Confidence Image

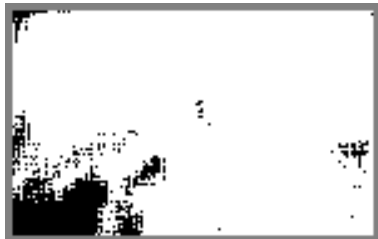


(c) KLD Binary Image

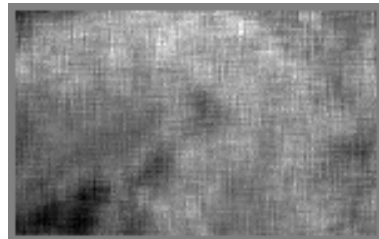


(d) KLD Confidence Image

Figure 4-13: Colony 1 Classification Method Effects



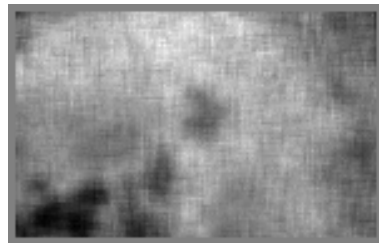
(a) SVM Binary Image



(b) SVM Confidence Image

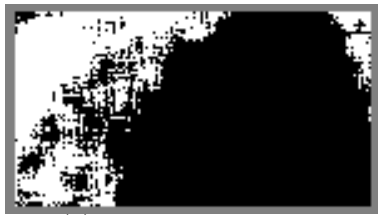


(c) KLD Binary Image

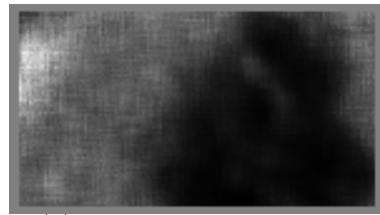


(d) KLD Confidence Image

Figure 4-14: Colony 2 Classification Method Effects



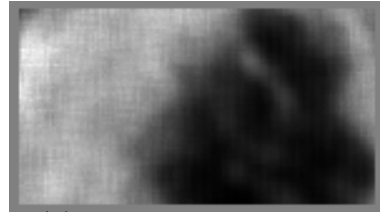
(a) SVM Binary Image



(b) SVM Confidence Image



(c) KLD Binary Image

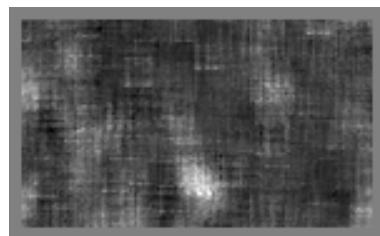


(d) KLD Confidence Image

Figure 4-15: Colony 3 Classification Method Effects



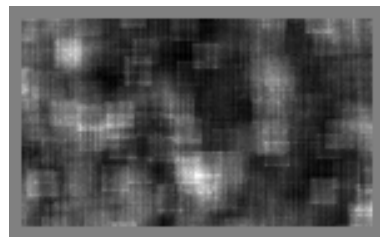
(a) SVM Binary Image



(b) SVM Confidence Image



(c) KLD Binary Image



(d) KLD Confidence Image

Figure 4-16: Colony 4 Classification Method Effects

Training Set Number	Paired Texture Types
1	flat texture + black texture
2	flat texture + loose texture
3	flat texture + composite bad
4	tight texture + black texture
5	tight texture + loose texture
6	tight texture + composite bad
7	composite good + black texture
8	composite good + loose texture
9	composite good + composite bad

Table 4.2: Nine Tier 1 Training Sets (Review)

outputs are arranged in increasing training set number from left to right and top to bottom. Thus, looking across the rows shows the effects of training on the different types of bad texture (black, loose or composite) for a fixed type of good texture; looking down the columns shows the effects of the type of good texture (flat, tight or composite) for a fixed bad texture.

In examining Colony 1 in Figure 4-17, we note that its homogeneous flat texture makes it a high quality colony. However, this flat homogeneity is also the cause of the wildly different binary reclassified representations of the colony. When using training sets 4, 5 and 6 (the middle row in the figure), the classifier cannot recognize the high quality of the flat-textured colony since it has only been trained to recognize tight texture as signifying colony quality. In contrast, when the classifier is trained to recognize flat regions as high quality, solely (top row) or in combination with the tight textured definition (bottom row), the resulting segmentations are more accurate.

The training set effects are not as clear for Colony 2 in Figure 4-18. Still, we do notice that the binary images in the first column do not segment as clearly the loose textured region in the lower left corner of the original colony. This is due to the fact that the classifier for these images has been trained using the black texture definition of poor quality rather than the loose texture definition. Thus, it has trouble clearly classifying this area of loose texture as a bad region.

In Figure 4-19 we see again what we would expect regarding the different binary images for the different training sets. Colony 3 shows an obvious dark portion in its center, signifying a poor quality region. In those binary images for which the black texture definition is employed (first and last columns), we see a clear classification of the dark central region as bad. However, for training sets 2 and 8, where the loose texture definition is used, the classifier cannot clearly distinguish the poor central black region.

The binary output for training set 5 presents an interesting case. Even though it is not trained to recognize black regions as bad, it still performs well in its segmentation of the center region. This is due to the fact that it is also trained to recognize tight texture as good. Since the texture surrounding the dark center is relatively tight, it correctly classifies this surrounding region and simply finds the center region less similar to tight

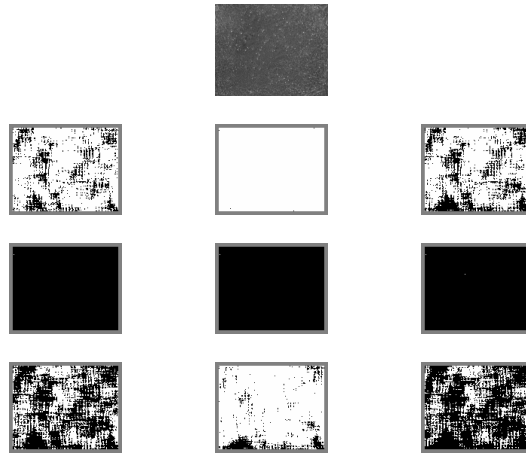


Figure 4-17: Colony 1 Training Set Effects

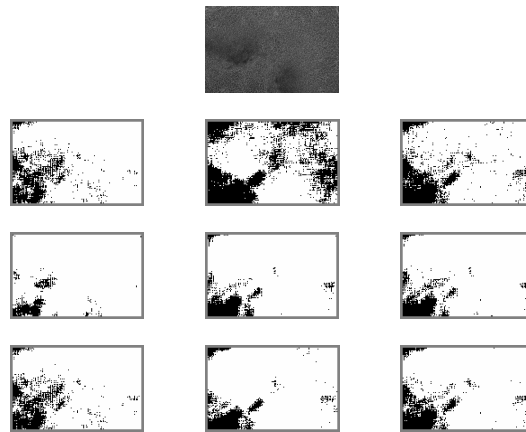


Figure 4-18: Colony 2 Training Set Effects

texture than to loose texture, thus correctly classifying it as bad.

All but one of the outputs for Colony 4 in Figure 4-20 classify the colony as almost completely of poor quality. The binary reclassified image using training set 2 (middle image in the top row) is the one exception. It is perhaps ironic that the SVM classifier used to create this reclassified image, trained to distinguish explicitly between flat and loose texture, would fail on a colony exhibiting extremely loose texture throughout. However, under further examination, the result makes sense for two reasons. First, since Colony 4 is the only one among the 77 colonies with such a loose texture, we do not include its extreme texture in the training data since it does not represent well the colony images at our disposal. Thus, it is rightfully difficult for the classifier to correctly segment the colony. Secondly, the windows used to capture the textural qualities of the colony are not large enough to recognize such a loose texture. Instead, these windows see only the flat regions among the widely-dispersed cells that create the colony's loose texture. Since the classifier is trained to recognize flat texture as good, it tends to incorrectly classify the very loose, bad texture as tight, good texture.

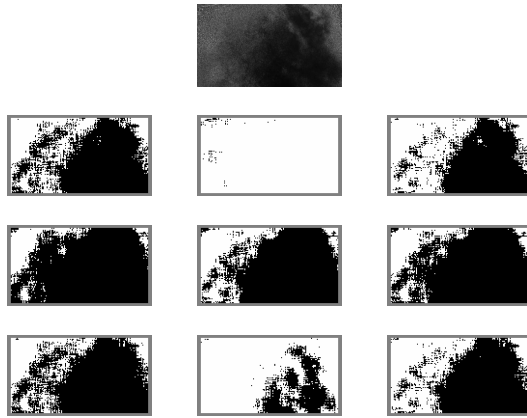


Figure 4-19: Colony 3 Training Set Effects

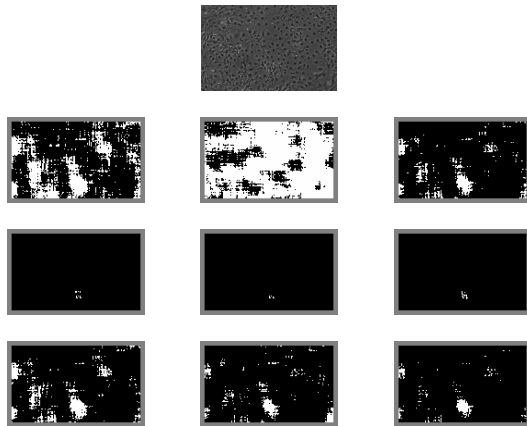


Figure 4-20: Colony 4 Training Set Effects



## Window Size Effects

Along with classification method and training set, the window size can conceivably affect the resulting stem cell colony segmentation. We analyze these window size effects by considering the SVM binary reclassified images for each of the four colonies using training set 5 (tight texture + loose texture). Figures 4-21, 4-22, 4-23 and 4-24 show the outputs for each colony; in each figure, the leftmost image uses a  $55 \times 55$  window, the middle image a  $45 \times 45$  window and the rightmost image a window of size  $35 \times 35$ .

The only common window size effect among the four colonies appears to be the increased speckling with the  $35 \times 35$  window. We attribute this to the increased locality of the analysis with this smaller window size; the larger windows naturally smooth out the local irregularities in the image. Overall though, we do not notice any dominant window size effects that would significantly influence our results.

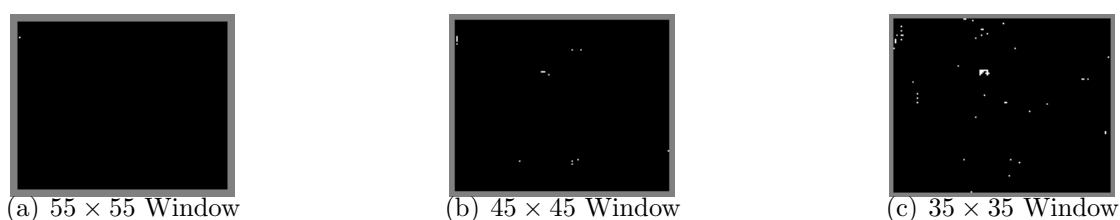


Figure 4-21: Colony 1 Window Size Effects

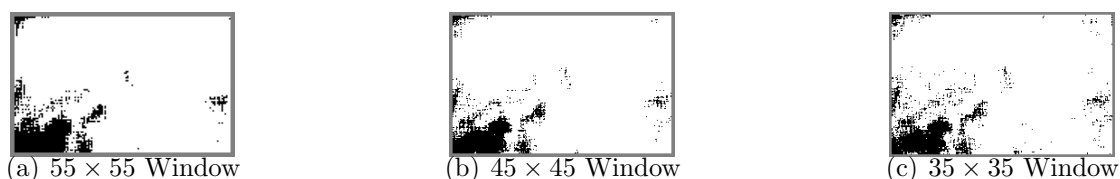


Figure 4-22: Colony 2 Window Size Effects

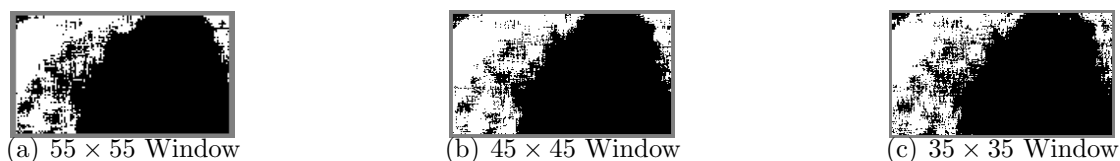


Figure 4-23: Colony 3 Window Size Effects

## Conclusions

The above analyses demonstrate some of the complexities of the stem cell classification problem. While the classification method and window size parameter choices do not significantly affect the resulting colony segmentations, the choice of training set can have

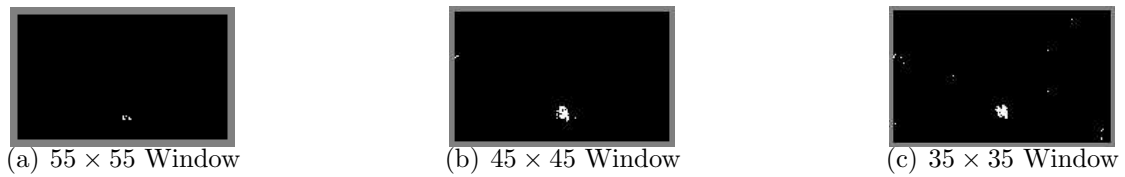


Figure 4-24: Colony 4 Window Size Effects

a legitimate effect. This is due to the difficulty in fully characterizing good and bad stem cell colony texture. The fact that there exist two types of both good and bad texture is evidence of this difficulty. Additionally, a wide range of textures is represented within each of these four types of texture. As it is not practical to train the classifiers on each manifestation of texture we might encounter, we trade textural specificity for practical applicability and train only on the four most common textures. We recommend the stem cell researchers using our work examine the outputs from all nine training sets to gain a full understanding of the quality of their colonies.

As we now transition to the quantitative stem cell categorization task of Chapter 5 we will encounter these difficulties again, and we consider methods to account for performance differences over the various parameter settings.

## Chapter 5

# Stem Cell Application: Quantitative Image Categorization

In Chapter 4 we outlined a two-tiered approach—diagrammed again here as Figure 5-1 below—to the problem of providing tools to aid stem cell researchers in the visual characterization of the health of their stem cell colonies. Then, based on the textural criteria of stem cell quality, we created various graphical representations that segmented each colony interior into good and bad regions using the SVM and KLD algorithms discussed in Chapters 2 and 3. Our hope was that these outputs would reduce the time spent by researchers in visually examining their colonies by providing a clear blueprint of the quality across the colonies and focusing their attention on those regions that might require closer inspection. This work constituted the Tier 1, or *qualitative interior textural segmentation* analysis.

While helpful to stem cell researchers, the Tier 1 analysis leaves open the opportunity for further work in two areas. First, the Tier 1 graphical outputs still require a subjective interpretation in order to make stem cell quality judgements. A *quantitative* measure of overall colony quality would thus be useful to mitigate against the natural subjectivity of individual researchers. Secondly, since the Tier 1 analysis focused strictly on *interior* textural quality, it naturally did not consider the third, or *border* criterion of stem cell quality. However, in implementing a quantitative analysis of overall colony quality, we should consider such border features.

The goal of this chapter is to report a successful method of implementing these two open tasks; this work constitutes the Tier 2, or *quantitative image categorization* analysis. We frame this analysis as a *multiclass*, or *multiway* categorization problem where we attempt to classify each full stem cell colony into one of several quality categories. Following the machine learning classification methodology presented in Section 2.1, we consider again the two steps of *feature representation* and *classification*, operating now at the full colony level rather than the pixel level of the benchmark and Tier 1 segmentation work.

We tackle the colony-wise feature representation problem in Section 5.1 by deriving *colony-wise interior texture features* from the Tier 1 outputs and introducing new *colony-wise border features*. This analysis yields the colony-wise feature representations  $\mathbf{x}_i$ ,  $i = 1, 2, \dots, 77$ , for each of the 77 stem cell colonies. We discuss in Section 5.2 the

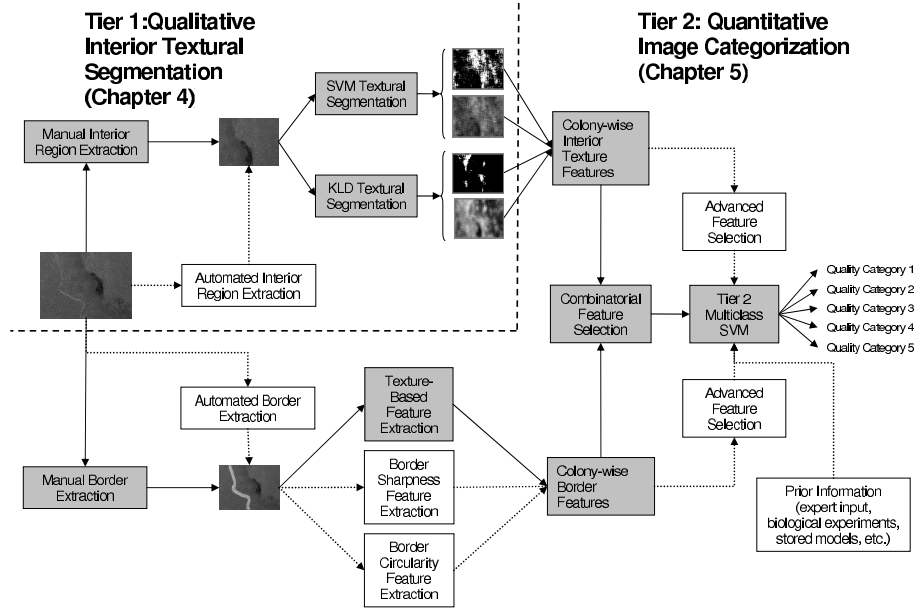


Figure 5-1: Two Tiered Stem Cell Analysis Methodology (Review)

associated multiclass labels  $y_i \in \{1, 2, \dots, 5\}$ ,  $i = 1, 2, \dots, 77$ , for each colony. Then, in Section 5.3 we describe the multiclass support vector machine classifier that trains and tests on the full colonies, and we outline various run setup procedures. In Section 5.4 we report the encouraging categorization rates we achieve with this methodology and give guidance for its practical implementation.

## 5.1 Colony-Wise Feature Extraction

Just as feature representation on the pixel level was the first step for the Tier 1 analysis, feature extraction on a colony level is the first Tier 2 task. Where the Tier 1 analysis worked at the level of local windows extracted about each pixel of the stem cell colony images, we now expand to the level of the entire colony in order to provide a quantitative measure of overall colony quality. These *colony-wise features* should capture the aggregate health of the colony.

We accomplish this by deriving features based on the criteria of stem cell colony quality discussed in Section 4.1.3. We recall the following three criteria:

1. Textural homogeneity
2. Textural tightness
3. Border sharpness and circularity

Since the first two criteria reference *interior textural quality* and the third focuses on *border quality*, we similarly partition the colony-wise features into two sets, those that characterize texture quality and those that consider border quality. We describe the specific features from each of these two sets in the two sections to follow.

### 5.1.1 Interior Texture Feature Extraction

We conjecture that we can capture the interior textural quality of a stem cell colony by considering the following two general measures:

1. The percent of the colony interior that is healthy and undifferentiated
2. The level of homogeneity of healthy regions across the colony

We propose four colony-wise features, called *mean features*, to capture the first characteristic and six features, called *variance features*, for the second.

All ten colony-wise interior texture features are easily calculated from the Tier 1 outputs of Chapter 4. Recall that the Tier 1 analysis performed a pixel-by-pixel classification of the interior region of each stem cell colony. Furthermore, each colony was analyzed using both the SVM and KLD classifiers with three different window sizes and nine different training sets. Thus, for a fixed Tier 1 window size and training set, we have binary classifications  $f_{SVM}(\mathbf{x}_i)$  and  $f_{KLD}(\mathbf{x}_i)$  and confidence values  $h_{SVM}(\mathbf{x}_i)$  and  $h_{KLD}(\mathbf{x}_i)$  for each pixel  $\mathbf{x}_i$  in the colony interior, depending on whether we employ the outputs from the SVM or KLD classifier. We derive the colony-wise interior texture features using these binary classification and confidence values. *Additionally, we derive these features assuming a fixed Tier 1 window size and training set.*

#### Mean Features

The first measure of interior texture quality listed above states that the percentage of healthy area in the colony interior reflects the relative quality of the colony itself. For example, a colony whose interior consists of a single high-quality region would be considered of high quality as a whole. A colony where only half of the interior area is healthy would itself be rated a mediocre colony.

Recall that the Tier 1 analysis provides a pixel-by-pixel classification of each colony interior. As a slight modification to the original approach of letting  $f(\mathbf{x}_i) = -1$  for poor quality pixels, we now set  $f(\mathbf{x}_i) = 0$  for such pixels. To then find the percentage of good pixels across the colony, we can simply calculate the colony average of the  $f(\mathbf{x}_i)$  values. This motivates the use of the mean operator to create colony-wise features. Depending on whether we use the SVM or KLD classification method, we have the following *SVM texture binary mean* and *KLD texture binary mean* features:

$$\mu_{f:SVM} = \frac{1}{N} \sum_{i=1}^N f_{SVM}(\mathbf{x}_i) \quad (5.1)$$

$$\mu_{f:KLD} = \frac{1}{N} \sum_{i=1}^N f_{KLD}(\mathbf{x}_i) \quad (5.2)$$

where  $N$  is the number of pixels in the interior of the stem cell colony being considered.

In Chapter 4, we found that we could create higher-resolution reclassified images by plotting the confidence values  $h(\mathbf{x}_i)$  for each pixel. Similarly, taking the average confidence values across each colony interior provides a higher-resolution colony-wise feature for the

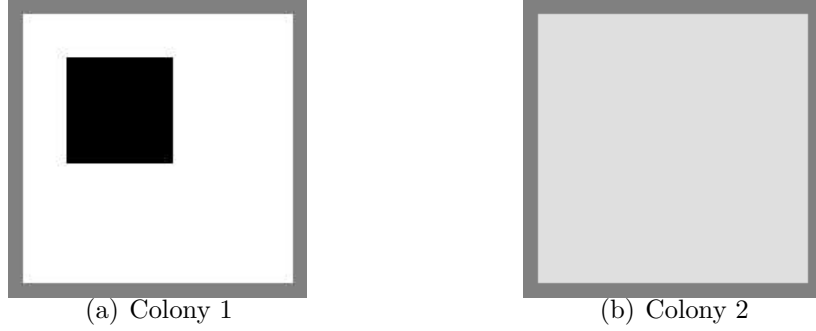


Figure 5-2: Confidence Reclassified Images of Two Toy Colonies

Tier 2 analysis. Thus, we have the following *SVM texture confidence mean* and *KLD texture confidence mean* features:

$$\mu_{h:SV M} = \frac{1}{N} \sum_{i=1}^N h_{SV M}(\mathbf{x}_i) \quad (5.3)$$

$$\mu_{h:KLD} = \frac{1}{N} \sum_{i=1}^N h_{KLD}(\mathbf{x}_i) \quad (5.4)$$

## Variance Features

The second measure of interior textural quality listed above is based on the homogeneity criterion of stem cell quality. That criterion states that a healthy colony exhibits textural homogeneity across the entire colony.

The four mean features in Equations 5.1 through 5.4 do not necessarily capture this homogeneity characteristic. For example, consider the confidence reclassified images of the two toy colonies in Figure 5-2. Outside of the obvious poor area in its interior, Colony 1 exhibits excellent quality. It returns a confidence mean value of 0.874. Colony 2 also has a confidence mean of 0.874 on account of the slightly degraded quality throughout. Thus, even though Colony 2 is clearly the superior colony due to its homogeneity, the confidence mean features cannot distinguish between the two.

However, we could use the variance operator to create features that capture the difference in the homogeneity of the two colonies and thus distinguish between them. Indeed, the variance in the confidence values across Colony 1 of 0.1101 is clearly distinguishable from the variance of 0 for Colony 2.

Using the above example as motivation, we introduce four variance-based features: *SVM texture binary variance*, *KLD texture binary variance*, *SVM texture confidence variance* and *KLD texture confidence variance*. Their respective mathematical expressions

are,

$$\sigma_{f:SV M}^2 = \frac{1}{N} \sum_{i=1}^N (f_{SV M}(\mathbf{x}_i) - \mu_{f:SV M})^2 \quad (5.5)$$

$$\sigma_{f:KLD}^2 = \frac{1}{N} \sum_{i=1}^N (f_{KLD}(\mathbf{x}_i) - \mu_{f:KLD})^2 \quad (5.6)$$

$$\sigma_{h:SV M}^2 = \frac{1}{N} \sum_{i=1}^N (h_{SV M}(\mathbf{x}_i) - \mu_{h:SV M})^2 \quad (5.7)$$

$$\sigma_{h:KLD}^2 = \frac{1}{N} \sum_{i=1}^N (h_{KLD}(\mathbf{x}_i) - \mu_{h:KLD})^2 \quad (5.8)$$

where  $N$  is the number of pixels in the colony interior being considered and the  $\mu$ . expressions are the mean features defined in Equations 5.1 through 5.4 above.

We introduce two additional outlier-based features to augment the variance features above in characterizing homogeneity. Specifically, we define new  $35 \times 35$  windows about each pixel in the colony under consideration. We then calculate the local confidence mean within each window. If the local mean value for a given pixel's window differs by more than two standard deviations from the confidence mean across the entire colony, we declare the pixel to be an outlier<sup>1</sup>. We define the *SV M texture outlier* and *KLD texture outlier* features to be the total number of outlier pixels across the colony using the SV M and KLD classifiers, respectively. Mathematically, we have,

$$\xi_{SV M} = \sum_{i=1}^N \delta \left( |\mu_{h:SV M}^{(i)} - \mu_{h:SV M}| > 2\sigma_{h:SV M} \right) \quad (5.9)$$

$$\xi_{KLD} = \sum_{i=1}^N \delta \left( |\mu_{h:KLD}^{(i)} - \mu_{h:KLD}| > 2\sigma_{h:KLD} \right) \quad (5.10)$$

where  $N$ ,  $\mu$ . and  $\sigma$ . are as above,  $\mu_{h:SV M}^{(i)}$  and  $\mu_{h:KLD}^{(i)}$  are the local means of the SV M and KLD confidence values within the  $35 \times 35$  window about pixel  $\mathbf{x}_i$ , and  $\delta(\cdot)$  is a function that returns a 1 if its argument is satisfied and a 0 otherwise.

## Feature Summary

Motivated by the textural criteria for stem cell colony quality and confirmed by an expert in the field of stem cell microscopy, we have proposed ten colony-wise features that characterize the overall health and quality of the interior region of a stem cell colony. Additionally, all ten features can be easily calculated from the outputs of the Tier 1 analysis

---

<sup>1</sup>The use of the mean statistic here is perhaps not the most appropriate approach for outlier detection since the local and colony means are themselves already skewed by outliers. A more robust measure, such as the median, would be more appropriate. However, we leave its implementation and analysis for future work.

Feature Name	Feature Symbol	Formula
SVM Texture Binary Mean	$\mu_{f:SVM}$	$\frac{1}{N} \sum_{i=1}^N f_{SVM}(\mathbf{x}_i)$
KLD Texture Binary Mean	$\mu_{f:KLD}$	$\frac{1}{N} \sum_{i=1}^N f_{KLD}(\mathbf{x}_i)$
SVM Texture Confidence Mean	$\mu_{h:SVM}$	$\frac{1}{N} \sum_{i=1}^N h_{SVM}(\mathbf{x}_i)$
KLD Texture Confidence Mean	$\mu_{h:KLD}$	$\frac{1}{N} \sum_{i=1}^N h_{KLD}(\mathbf{x}_i)$
SVM Texture Binary Variance	$\sigma_{f:SVM}^2$	$\frac{1}{N} \sum_{i=1}^N (f_{SVM}(\mathbf{x}_i) - \mu_{f:SVM})^2$
KLD Texture Binary Variance	$\sigma_{f:KLD}^2$	$\frac{1}{N} \sum_{i=1}^N (f_{KLD}(\mathbf{x}_i) - \mu_{f:KLD})^2$
SVM Texture Confidence Variance	$\sigma_{h:SVM}^2$	$\frac{1}{N} \sum_{i=1}^N (h_{SVM}(\mathbf{x}_i) - \mu_{h:SVM})^2$
KLD Texture Confidence Variance	$\sigma_{h:KLD}^2$	$\frac{1}{N} \sum_{i=1}^N (h_{KLD}(\mathbf{x}_i) - \mu_{h:KLD})^2$
SVM Texture Outlier	$\xi_{SVM}$	$\sum_{i=1}^N \delta \left(  \mu_{h:SVM}^{(i)} - \mu_{h:SVM}  > 2\sigma_{h:SVM} \right)$
KLD Texture Outlier	$\xi_{KLD}$	$\sum_{i=1}^N \delta \left(  \mu_{h:KLD}^{(i)} - \mu_{h:KLD}  > 2\sigma_{h:KLD} \right)$

Table 5.1: Tier 2 Colony-Wise Interior Texture Features

completed in Chapter 4. Thus, there is little additional computational cost in extracting these colony-wise features. All ten colony-wise interior texture features are summarized in Table 5.1 for a fixed Tier 1 window size and training set.

Relaxing the assumption of a fixed Tier 1 window size and training set, we find that we actually have  $3 \cdot 9 = 27$  different versions of the ten colony-wise interior texture features of Table 5.1. Each version depends on the specific window size and training set employed in the Tier 1 analysis to calculate the binary classifications  $f_{SVM}(\mathbf{x}_i)$  and  $f_{KLD}(\mathbf{x}_i)$  and confidence values  $h_{SVM}(\mathbf{x}_i)$  and  $h_{KLD}(\mathbf{x}_i)$  then used to derive the colony-wise features. Considering these 27 versions, we actually have  $10 \cdot (3 \cdot 9) = 270$  available features to characterize the interior texture of a stem cell colony. The astute reader will recognize that using all 270 features in training would likely yield a severely overfit and complex classifier. Thus, we discuss an approach to *feature selection* in Section 5.3.2 below to avoid these problems.

### 5.1.2 Border Feature Extraction

While we are able to utilize the various results from the Tier 1 analysis for the calculation of the colony-wise interior texture features, we do not have such a luxury for the creation of the colony-wise border features since the Tier 1 analysis did not consider border characteristics. Thus, we must first perform some preliminary steps in order to calculate the data that will then be used to create the actual border features.

We begin this process by expanding on and revising the border criterion of stem cell colony quality documented at the beginning of this section. We then describe how we extract information from the stem cell colony images to capture this revised criterion. Finally, we outline the actual border features that we calculate.



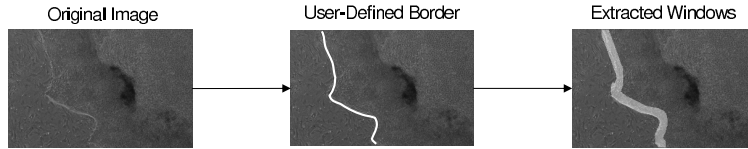


Figure 5-3: Border Window Extraction Process

## Revised Border Criterion

The original border criterion dictates that a good colony exhibits a sharp and circular border. While we consider a revised border criterion discussed below, methods do in fact exist for the analysis of border sharpness and circularity. For example, gradient methods can be used to quantify changes in intensity as the colony border is crossed in a radial direction from the colony centroid. A large gradient, reflecting a sharp intensity change from colony interior to exterior, would indicate a sharp border while a low gradient value would signal a less-defined border [15]. Measures of circularity include the circularity index, fractal-based methods and Fourier analysis [28]. Other novel techniques are discussed in the literature, often in the application to cancerous melanoma detection [15] [28].

Despite the availability of these methods, we choose instead to employ a slightly different manifestation of the border quality criterion that dovetails better with our work in texture analysis. The revised criterion states that *a good colony border will contain high quality texture immediately interior to the colony border*. This revision has a strong biological justification. Namely, we find that stem cell colonies generally show the first signs of degradation and differentiation in two specific regions: the center and border regions of the colony. Since this degradation is reflected in the texture of the stem cells in those regions, by analyzing texture in the border region we hope to capture the border quality of the colony<sup>2</sup>.

## Border Window Extraction

Implementing the revised border criterion involves analyzing texture quality in the border region in much the same manner as we have done in Chapter 4 for the interior region. Just as we defined windows around each pixel of the colony interior, calculated wavelet-based texture features and classified each pixel according to those features using the SVM and KLD methods, we now perform the same basic tasks on the border region. This process of border window extraction is illustrated in Figure 5-3 and described in more detail below.

The first step involves explicitly defining the colony border. This process of *border segmentation* can be performed either manually by an expert or in some automated fashion. Sophisticated automated border segmentation algorithms exist in the literature. For example, Pien *et. al.* employ the curve evolution method of Shah to the task of segmenting anatomic structures of the human brain in magnetic resonance images [37] [47]. However, as it is not the primary focus of our work to consider such automated methods, we leave their implementation to future work and employ the manual segmentation approach.

---

<sup>2</sup>Since the texture in the colony interior has already been incorporated into the colony-wise interior texture features of Section 5.1.1 above, we do not explicitly analyze the center region again here.

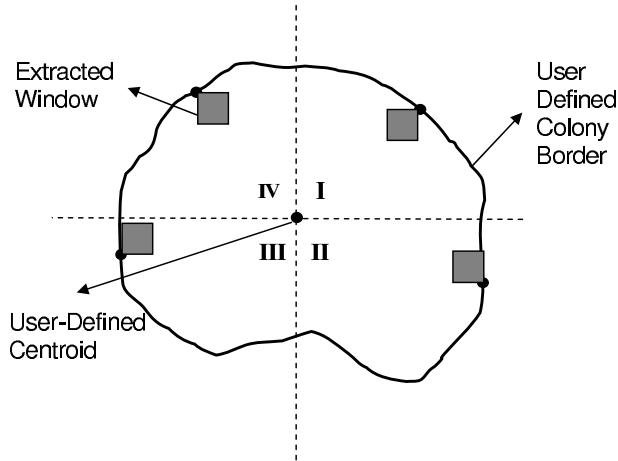


Figure 5-4: Border Window Orientation

With these defined borders, we then extract windows about each border pixel on which to perform the textural analysis that we use to characterize the overall border quality in accordance with the revised border criterion. To speed up implementation of this process, we employ an automated method requiring the user to specify only one parameter, the colony centroid. This centroid location partitions the image into four quadrants. A window is then placed about each border pixel in an orientation dictated by the quadrant in which the pixel lies. The four possible orientations are shown in Figure 5-4.

By following this process for each of the 77 colonies, we can extract the border windows on which to perform the wavelet-based texture analysis discussed in the previous chapters. Specifically, we extract border windows of size  $35 \times 35$ ,  $45 \times 45$  and  $55 \times 55$  and run these windows through the SVM and KLD algorithms of Section 3.2 using each of the nine training sets outlined in Section 4.2.3. This results in the various binary classifications  $f_{SVM}(\mathbf{x}_i)$  and  $f_{KLD}(\mathbf{x}_i)$  and confidence values  $h_{SVM}(\mathbf{x}_i)$  and  $h_{KLD}(\mathbf{x}_i)$  for a fixed window size and training set that characterize each border pixel  $\mathbf{x}_i$  of each colony.

### Colony-Wise Border Features

Consider for now that we have fixed the window size and training set used in the border window analysis. Then, using the resulting binary classifications and confidence values for each border pixel, we calculate eight colony-wise border features analogous to the first eight colony-wise interior texture features. That is, we take the mean and variance of the binary and confidence values across each colony's border pixels using both the SVM and KLD methods. These features are summarized in Table 5.2.

As with the colony-wise interior texture features, if we relax the assumption of a fixed window size and training set, we actually have  $8 \cdot (3 \cdot 9) = 216$  available colony-wise border features to characterize the border of a stem cell colony. Again, as with the colony-wise interior texture features, we discuss below how to select only a subset of these 216 features to avoid an overfit and complex model.

Feature Name	Feature Symbol	Formula
SVM Border Binary Mean	$\mu_{f: SVM}$	$\frac{1}{N} \sum_{i=1}^N f_{SVM}(\mathbf{x}_i)$
KLD Border Binary Mean	$\mu_{f: KLD}$	$\frac{1}{N} \sum_{i=1}^N f_{KLD}(\mathbf{x}_i)$
SVM Border Confidence Mean	$\mu_{h: SVM}$	$\frac{1}{N} \sum_{i=1}^N h_{SVM}(\mathbf{x}_i)$
KLD Border Confidence Mean	$\mu_{h: KLD}$	$\frac{1}{N} \sum_{i=1}^N h_{KLD}(\mathbf{x}_i)$
SVM Border Binary Variance	$\sigma_{f: SVM}^2$	$\frac{1}{N} \sum_{i=1}^N (f_{SVM}(\mathbf{x}_i) - \mu_{f: SVM})^2$
KLD Border Binary Variance	$\sigma_{f: KLD}^2$	$\frac{1}{N} \sum_{i=1}^N (f_{KLD}(\mathbf{x}_i) - \mu_{f: KLD})^2$
SVM Border Confidence Variance	$\sigma_{h: SVM}^2$	$\frac{1}{N} \sum_{i=1}^N (h_{SVM}(\mathbf{x}_i) - \mu_{h: SVM})^2$
KLD Border Confidence Variance	$\sigma_{h: KLD}^2$	$\frac{1}{N} \sum_{i=1}^N (h_{KLD}(\mathbf{x}_i) - \mu_{h: KLD})^2$

Table 5.2: Tier 2 Colony-Wise Border Features

Category	Description
1	Great texture and sharp edges
2	Imperfect but good colony Some darkening or thinning of the edge or center of the colony Starting to differentiate
3	Compromised colony with a significant thinning of the edge or thickening of the center of the colony
4	Poor colony with some usable cells Mostly differentiated
5	Unusable colony with no usable stem cells Nearly complete differentiation

Table 5.3: Full Colony Rating Scale

## 5.2 Expert Rating of Colony Quality

Having defined the characteristics that we use to create the colony-wise feature vectors  $\mathbf{x}_i$ ,  $i = 1, 2, \dots, 77$ , for each full colony in the categorization process, we now turn to defining the true labels, or ratings  $y_i \in \{1, 2, \dots, 5\}$  that we associate with each colony. This labelling was provided by an experienced stem cell research biologist, and we consider his expert rating to be the “truth” in the sense that it is the standard with which we compare the outputs from our categorization methodology.

This rating is based on overall colony quality using all three stem cell quality criteria outlined at the beginning of Section 5.1 above. Table 5.3 lists the specific guidance used to rate the colonies into the five categories.

The partition of the colonies into as many as five categories poses some significant difficulties to our analysis. First, a data set of 77 stem cell colonies is barely sufficient for making even pairwise classifications, let alone a five-way comparison. Considering that after partitioning the 77 colonies into separate training and test sets for each of the five categories, we have only between seven and ten colonies on which to train for each category, we realize that we are embarking on a truly difficult endeavor. Thus, we expect

that this scarcity of data will affect our results and that better results could be obtained with more data.

Secondly, a five-way labelling reduces the differences between colonies in adjacent categories making it even more difficult to distinguish between them. Of these adjacent categories, we find it particularly difficult to discriminate between categories 1 and 2. This is true both to the expert eye<sup>3</sup> and, as we shall see below, to our Tier 2 categorization method.

To simultaneously ease these two problems, we consider combining the category 1 and 2 colonies into a composite “1/2” category, thus yielding only four categories among which to distinguish. This approach increases the amount of data within the composite category on which to train and eliminates the difficult task of distinguishing between categories 1 and 2. However, we do not abandon the hope of separating categories 1 and 2 and discuss in Section 5.4.3 below our attempts at doing so.

## 5.3 Colony-Wise Classification Methodology

With colony-wise feature vectors  $\mathbf{x}_i$  and associated multiclass labels  $y_i$ , we have accomplished the feature extraction step of the machine learning classification paradigm as applied to the full stem cell colonies. We must now choose a classification method that can use these colony-wise features to accurately categorize the full colony images into the five (or four) quality categories described above. As in Section 2.3, we are free to choose any suitable method, such as a multiway neural network or nearest neighbor classifier. However, we again choose the support vector machine for the same reasons as outlined in Section 2.3.2 earlier, namely its superior theoretical and empirical performance over other methods.

While algorithmically similar to the SVM employed in Chapters 3 and 4 for the task of image segmentation, we now consider a new instantiation of the method. We replace wavelet energy features with the colony-wise features defined above, and instead of making binary pixel-by-pixel classifications, we make multiway colony-wise classifications. Additionally, training and test data consists now of full stem cell colonies rather than textural windows.

In the remainder of this section, we outline the multiclass extension of the now well-known binary support vector machine and describe the various runs that we perform to demonstrate the effectiveness of our Tier 2 methodology.

### 5.3.1 Multiclass SVM Formulation

There exist a number of methods for formulating multiway classifiers. Most of these simply involve running a combination of binary classifications and combining the results to arrive at a multiclass solution. Examples include the *one-vs-one* [32] and *decision tree* [51] approaches. The multiclass problem can also be considered in the more flexible framework

---

<sup>3</sup>In a five category comparison, it is quite possible that even an expert microscopist might classify a generally good colony to category 1 one day and to category 2 another based on the specific biological criteria on which the expert focuses each day.

of *error correcting output codes* (ECOC) [10]. In essence, all of these multiclass approaches are generic as they simply provide a framework for combining binary classifications into multiclass results; thus, any suitable binary method can be employed as the underlying classifier.

As mentioned above, we choose to use the support vector machine as our underlying binary classification method. Additionally, we employ a specific implementation of the ECOC framework, namely the *one-vs-all* approach. We give a brief summary of this approach here and leave further details and development to Appendix A.

Training the one-vs-all  $k$ -class SVM involves creating  $k$  different binary classifiers. For the  $i$ th binary classifier, we train data from the  $i$ th class against a composite data set of the remaining  $k - 1$  classes. This yields a function  $h_i(\mathbf{x})$  that we interpret as returning the confidence that a stem cell colony  $\mathbf{x}$  should be classified into class  $i$ . Training all  $k$  binary classifiers yields a vector of confidence functions  $(h_1(\mathbf{x}), h_2(\mathbf{x}), \dots, h_k(\mathbf{x}))$ .

We then apply the multiway classification function [41],

$$H(\mathbf{x}) = \arg \max_{j \in \{1, \dots, k\}} h_j(\mathbf{x}) \quad (5.11)$$

to classify a colony  $\mathbf{x}$  into one of the  $k$  classes. Intuitively, Equation 5.11 dictates that we classify the colony  $\mathbf{x}$  to the class which is the most confident of  $\mathbf{x}$  belonging to it rather than not. For  $k = 2$ , this multiclass approach collapses to the standard binary SVM classifier.

### 5.3.2 Run Setup

Simply stated, the goal of the Tier 2 problem is to use colony-wise feature vectors to classify stem cell colonies into five (or four) categories. An obvious, but unrefined solution approach dictates that we attempt a full five-way (or four-way) categorization of the available stem cell colonies using all 486 colony-wise features (270 interior texture features and 216 border features) as inputs. This approach has several drawbacks.

First, using all 486 features results in an excessively high dimensional feature vector. Even the SVM, with its ability to handle high dimensional feature vectors, would not be expected to successfully identify which of the 486 features are truly useful in defining the underlying structure of the colony categorization problem. As a remedy, before running the SVM classifier, we perform feature selection in order to reduce the dimensionality of the feature vectors seen by the SVM. We describe our methodology below.

Secondly, as mentioned in Section 5.2 above, we have only approximately ten colonies per category for use as training data. With such little training data, a full five-way (or four-way) categorization becomes a very difficult problem. Thus, we find it beneficial to consider less ambitious efforts, such as pairwise or three-way categorizations. We discuss later the specific category comparison runs that we perform.

The practical result of implementing these two modifications is that we replace the single, infeasible run described above with a series of well-selected smaller runs. By then defining optimality criteria with respect to these smaller problems, we can report results and make conclusions regarding the Tier 2 problem of using colony-wise features to classify stem cell colonies into quality categories.

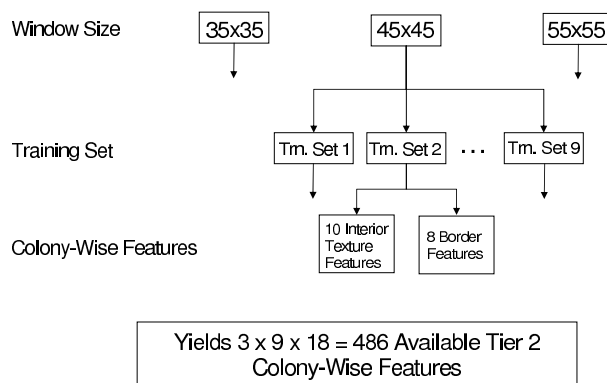


Figure 5-5: Tier 2 Colony-Wise Features

## Feature Selection

The first modification we make to the single, full run described above involves feature selection. As depicted in Figure 5-5, there exist 486 colony-wise features that we can use to characterize a given stem cell colony. This high dimensionality in the face of limited training data will naturally result in ineffective categorization. However, by first reducing the dimensionality of the feature vectors, we hope to attain legitimate results.

Approaches to feature selection have been considered both for general data analysis problems and specifically for the support vector machine. One general approach involves performing a regression analysis on the input data and employing feature selection methods tailored to the regression problem. Examples of such methods include forward stepwise selection, backward stepwise selection, ridge regression, partial least squares and principal components analysis [20]. Only those features found to be the significant in this pre-analysis are then used as inputs to the chosen classification method. Another approach might involve using various *data mining* algorithms such as CART and MART—designed specifically to mine for useful information or features in large and messy data sets—to determine the most important features to be used as inputs in the final classification scheme [20].

Direct feature selection for the SVM is not difficult when a linear kernel is employed. In that case, one need only calculate the  $\mathbf{w}$  vector defining the optimal hyperplane and identify the most influential features as those whose components in the  $\mathbf{w}$  vector have the largest absolute values. For other kernels such as the polynomial and radial basis function kernels that we use in this work, feature selection is rather more difficult. Nevertheless, methods for nonlinear SVM feature selection have been developed and are discussed in the literature [21] [57].

While these advanced feature selection methods might be considered in future work, we employ a simpler combinatorial approach. In general, this method involves choosing various subsets of the 486 features and running the SVM with each of the corresponding smaller feature vectors. Those features contained in the specific feature subset yielding the highest performance are determined to be the most influential.

A full combinatorial implementation where we run the SVM on every possible subset

<b>Group Name</b>	<b>Included Features</b>
Texture Binary	$\mu_{f:SVM}, \mu_{f:KLD}, \sigma_{f:SVM}^2, \sigma_{f:KLD}^2$
Texture Confidence	$\mu_{h:SVM}, \mu_{h:KLD}, \sigma_{h:SVM}^2, \sigma_{h:KLD}^2$
Texture Outlier	$\xi_{SVM}, \xi_{KLD}$
Texture Binary and Confidence	$\mu_{f:SVM}, \mu_{f:KLD}, \sigma_{f:SVM}^2, \sigma_{f:KLD}^2$ $\mu_{h:SVM}, \mu_{h:KLD}, \sigma_{h:SVM}^2, \sigma_{h:KLD}^2$
Texture Binary and Outlier	$\mu_{f:SVM}, \mu_{f:KLD}, \sigma_{f:SVM}^2, \sigma_{f:KLD}^2$ $\xi_{SVM}, \xi_{KLD}$
Texture Confidence and Outlier	$\mu_{h:SVM}, \mu_{h:KLD}, \sigma_{h:SVM}^2, \sigma_{h:KLD}^2$ $\xi_{SVM}, \xi_{KLD}$
All Texture	$\mu_{f:SVM}, \mu_{f:KLD}, \sigma_{f:SVM}^2, \sigma_{f:KLD}^2$ $\mu_{h:SVM}, \mu_{h:KLD}, \sigma_{h:SVM}^2, \sigma_{h:KLD}^2$ $\xi_{SVM}, \xi_{KLD}$

Table 5.4: Colony-Wise Interior Texture Feature Groups

<b>Group Name</b>	<b>Included Features</b>
None	$\emptyset$
Border Binary	$\mu_{f:SVM}, \mu_{f:KLD}, \sigma_{f:SVM}^2, \sigma_{f:KLD}^2$
Border Confidence	$\mu_{h:SVM}, \mu_{h:KLD}, \sigma_{h:SVM}^2, \sigma_{h:KLD}^2$
All Border	$\mu_{f:SVM}, \mu_{f:KLD}, \sigma_{f:SVM}^2, \sigma_{f:KLD}^2$ $\mu_{h:SVM}, \mu_{h:KLD}, \sigma_{h:SVM}^2, \sigma_{h:KLD}^2$

Table 5.5: Colony-Wise Border Feature Groups

of the 486 features would require,

$$\sum_{i=1}^{486} \binom{486}{i} \rightarrow \infty \quad (5.12)$$

runs. Clearly, we must limit the subsets we consider. We do so by first fixing the Tier 1 window size and training set reducing our feature choices to the 10 colony-wise interior texture features listed in Table 5.1 and 8 colony-wise border features of Table 5.2. Instead of considering these features in isolation, we operate on various fixed combinations of features by creating seven interior texture feature groups and four border feature groups. By pairing one of the interior texture groups with one of the border feature groups, we can define the specific feature vector sent to the SVM.

The seven interior texture feature groups are listed in Table 5.4 via their corresponding feature symbol from Table 5.1 while the four border feature groups are shown in Table 5.5 referencing Table 5.2.

Pairing each interior texture group with each border group results in  $7 \cdot 4 = 28$  different feature sets. However, recall that the Tier 1 window size and training set has been fixed throughout this discussion. By employing each of the three window sizes and each of

the nine training sets on each of the 28 combinations described above, we actually have  $3 \cdot 9 \cdot 28 = 756$  possible different feature vectors to be used as inputs into the SVM.

Finally, note that each of these 756 different feature vectors employs only one of the nine Tier 1 training sets. As we discovered in Section 4.3.3, the choice of training set can have a significant impact on Tier 1 analysis. Since we have used the Tier 1 analyzes to create the Tier 2 colony-wise features, we expect that the training set choice will also affect the Tier 2 results. Thus, we also consider feature vectors containing features derived from all nine training sets. Considering the three window sizes, this adds  $3 \cdot 28 = 84$  additional feature vectors.

Thus, our feature selection methodology results in the creation of  $756 + 84 = 840$  different feature vectors. This is clearly more feasible than following the full combinatorial approach with its excessively large number of feature vectors. By running the SVM using each of these 840 feature vectors, we can calculate results for each and determine which specific features yield the best categorization performance.

## Parameter Settings

Implementing the feature selection methodology discussed above requires us to run the SVM categorization algorithm 840 times to calculate results for each of the 840 different feature vectors. For each of these runs, we need to specify various SVM parameters. For simplicity, we choose to fix the SVM's cost parameter  $C$  to 1. However, we expect that the SVM kernel choice may affect the results of each run. Thus, we experiment with two different kernels, the polynomial of degree 2 and the radial basis function. By performing the 840 runs outlined above with each of these two kernels, we arrive at  $840 \cdot 2 = 1680$  different Tier 2 runs, each reflecting a different setting of the colony-wise feature vector and multiclass SVM kernel.

## Category Comparisons

As mentioned at the beginning of this section, we find it beneficial to perform, in addition to the full five-way (or four-way) categorization, various other category comparisons, such as pairwise or three-way comparisons. Specifically, for the five category case, we perform all ten pairwise comparisons, a comparison among categories 1, 3 and 5 and the five-way run. For the four category case where we have combined categories 1 and 2, we perform the six pairwise comparisons along with the full four-way comparison.

For each of these comparison runs, we need to specify training and test data from among the colonies involved. We accomplish this by randomly splitting the applicable colonies into disjoint training and test sets. For example, if we are comparing categories 1 and 2, we randomly split the category 1 colonies into a training and test set and the category 2 colonies into a training and test set<sup>4</sup>. In order to achieve more accurate results, we perform each comparison run ten times using a random training/test partition each

---

<sup>4</sup>As mentioned briefly in Section 4.3.1, for a given Tier 2 run, we recuse from the analysis those colonies contributing data to the specific Tier 1 training set used in the creation of the Tier 2 feature vector employed in that run. In short, we never test on data that has been used previously for training, whether in Tier 1 or Tier 2.



Categories	1	2	3	4	5	1 vs. 3 vs. 5
1 vs.		49.0%	86.9%	89.2%	86.7%	70.0%
2 vs.			83.3%	82.7%	88.2%	1 vs. 2 vs. 3 vs. 4 vs. 5
3 vs.				66.3%	81.0%	46.5%
4 vs.					77.0%	

(a) Five Category Case

Categories	1/2	3	4	5	1/2 vs. 3 vs. 4 vs. 5
1/2 vs.		75.7%	89.0%	91.5%	
3 vs.			80.4%	83.6%	66.5%
4 vs.				87.0%	

(b) Four Category Case

Figure 5-6: General Tier 2 Performance Matrices

Categories	1/2	3	4	5	1/2 vs. 3 vs. 4 vs. 5
1/2 vs.		74.4%	83.6%	92.6%	
3 vs.			84.0%	81.8%	63.1%
4 vs.				84.5%	

Figure 5-7: Example Mean Optimal Performance Matrix

time. We then take the average categorization rate over the ten runs as our performance measure.

Thus, the full Tier 2 analysis consists of performing each of the 1680 Tier 2 runs described above for each of these category comparisons. Arranging the categorization rates for each of the 1680 runs in a *performance matrix*, as shown for a particular feature vector and SVM kernel setting in Figure 5-6, the Tier 2 analysis yields 1680 different performance matrices.

To find the specific feature vector and SVM kernel setting that yields the highest performance out of the 1680 different settings, we must sift through this performance data in a logical manner. We do so by defining two types of optimality, mean and individual. We then search through the 1680 performance matrices to identify the setting that satisfies each type of optimality.

## Defining Optimality

For a specific setting to satisfy *mean optimality*, it must yield, among all 1680 different settings, the highest average performance across the various comparisons in the performance matrix. For example, for a particular setting in the four category case, we might have the performance matrix shown in Figure 5-7. If no other setting of the feature vector and SVM kernel produces a higher average performance than its 80.57%, then this particular setting would be *mean optimal*.

While mean optimality is defined on the level of the feature and parameter setting, *individual optimality* works at the category comparison level. That is, we search for the feature vector and SVM kernel setting that yields the highest performance for *each* category comparison. Thus, in contrast to mean optimality where a single setting is optimal across the entire performance matrix, the enforcement of individual optimality results in a potentially different optimal setting for each comparison. Figure 5-8 illustrates the individually optimal results that we might achieve. For example, in comparing categories 4 and 5, no feature vector and SVM kernel setting will yield better performance than the specific individual optimal setting that produces the 93.5% rate in the figure. Additionally, the setting that produces the 86.2% when comparing between categories 3 and

Categories	1/2	3	4	5		
1/2 vs.		81.4%	90.5%	96.7%	1/2 vs. 3 vs. 4 vs. 5	
3 vs.			86.2%	88.0%		68.8%
4 vs.				93.5%		

Figure 5-8: Example Individual Optimal Performance Matrix

4 would likely be different from the setting leading to the 88.0% rate for the category 3 and 5 comparison.

From a practical stem cell research perspective, these two optimality definitions have different interpretations and uses. The use of individual optimality is appropriate in two practical cases. First, and most realistically, if a researcher has no prior information regarding the quality of a set of colonies and wishes to make a full four-way categorization, he should use the individual optimal setting for the four-way comparison. For the illustrative example in Figure 5-8, the researcher would expect to be correct on 68.8% of the colonies. Secondly, consider a researcher who has a set of colonies that are known a priori to be generally poor. If he wishes to further quantify their poorness, he might only consider classifying them between categories 4 and 5. In that case, he would use the individual optimal setting for the category 4 and 5 comparison and expect a correct categorization 93.5% of the time.

While individual optimality is appropriate for making binary comparisons *with* prior knowledge, mean optimality should be used for binary comparisons *without* prior knowledge. In this case, a researcher could use the mean optimal setting to make all the different binary comparisons on an unknown colony yielding some information regarding the category to which the colony should be classified. Mean optimality is also useful for determining the overall performance ability of the Tier 2 methodology and for identifying trends in the feature and SVM parameter settings.

## 5.4 Quantitative Image Categorization Results

We are now prepared to report the results of these runs on our 77 stem cell colonies. Our goals are four-fold:

1. To report the high colony-wise categorization rates that we achieve with our Tier 2 methodology when categories 1 and 2 are combined
2. To explore the colony-wise feature and SVM parameter settings that produce these high categorization rates
3. To determine whether using our revised border criterion of Section 5.1.2 increases categorization performance
4. To discuss the present successes and difficulties in separating categories 1 and 2

We discuss the first three goals in the first two sections, leaving the fourth goal for Section 5.4.3. Then, in Section 5.4.4 we make some final conclusions and briefly discuss the practical implementation of the Tier 2 methodology.

Categories	1/2	3	4	5		
1/2 vs.		74.4%	83.6%	92.6%		1/2 vs. 3 vs. 4 vs. 5
3 vs.			84.0%	81.8%		63.1%
4 vs.				84.5%		

Figure 5-9: Four-Way Mean Optimal Performance Matrix

Categories	1/2	3	4	5		
1/2 vs.		74.4% 76.0%	83.6% 87.1%	92.6% 90.1%		1/2 vs. 3 vs. 4 vs. 5
3 vs.			84.0% 79.1%	81.8% 76.9%		63.1% 65.1%
4 vs.				84.5% 84.5%		

Figure 5-10: Four-Way Mean Optimal Border Effects Performance Matrix

### 5.4.1 Mean Optimality Results

Figure 5-9 shows the mean optimal performance matrix when we combine categories 1 and 2. These results reflect the use of a  $45 \times 45$  window, all nine training sets, texture binary and outlier features, no border features and the radial basis function kernel. We find these results quite pleasing. All binary comparisons show a correct categorization rate above 74.0% with most higher than 80.0%.

Considering the specific setting that produces these results, we first notice that no border features are included. We will examine this result in more detail below. The inclusion of the texture binary and outlier features supports our hypothesis of Section 5.1.1 that both the percentage of healthy area and the level of homogeneity in the colony contribute to its overall characterization.

To examine further the effects of including the border features derived from our revised border criterion, we report the mean optimal results under two different conditions: required exclusion and required inclusion of the border features. Figure 5-10 shows the resulting performance matrix, where, for each category comparison, the first percentage reflects border feature exclusion and the second number reflects inclusion. Category comparisons for which the use of border features either does not change or improves performance are shaded.

We notice that the border features—specifically the border binary features—improve performance in three of the seven category comparisons and have no effect in a fourth. Additionally, the mean performances of the two cases differ by only 0.7%. Thus, we cannot make any clear conclusions as to the effects of border inclusion, and in the case of mean optimality, we are indifferent between including and excluding the revised border criterion features.

### 5.4.2 Individual Optimality Results

Figure 5-11 shows the individual optimal performance matrix. We first note the high categorization rates across all the comparisons. We are especially successful in separating category 5 from the other categories, achieving no worse than 88.0% and up to 96.7% performance for all three comparisons. Note also that the 68.8% rate when comparing

Categories	1/2	3	4	5		
1/2 vs.		81.4%	90.5%	96.7%		1/2 vs. 3 vs. 4 vs. 5
3 vs.			86.2%	88.0%		68.8%
4 vs.				93.5%		

Figure 5-11: Four-Way Individual Optimal Performance Matrix

Categories	1/2	3	4	5		
1/2 vs.		81.4% 81.0%	90.5% 89.7%	96.7% 94.3%		1/2 vs. 3 vs. 4 vs. 5
3 vs.			86.2% 84.0%	88.0% 86.8%		67.8% 68.8%
4 vs.				93.5% 92.4%		

Figure 5-12: Four-Way Individual Optimal Border Effects Performance Matrix

all four categories is excellent considering than we would expect only a 25.0% rate for a random classifier. We also observe that these individual optimal rates are consistently higher than the corresponding mean optimal rates in Figure 5-9 above. This is to be expected since individual optimality chooses the best feature and parameter setting for each comparison, thus not imposing a common, and possibly worse, setting on all comparisons as with the mean optimal case.

In examining the feature and parameter settings that yield each individual optimal categorization rate, we observe that the border features are used in only one of the seven comparisons, namely the full four-way comparison. We examine this result more below. Additionally, the most common texture features include the texture binary and outlier features.

Figure 5-12 compares border exclusion and inclusion for the individual optimal case. Overall, excluding the border features improves performance by only 1.0%, and while it produces higher rates in six of the seven comparisons, the performance increase in each is minimal. Thus, as with mean optimality, we are indifferent between including and excluding the border features.

### 5.4.3 On Separating Categories 1 and 2

In addition to the four-way results presented above where we employed a composite category consisting of the original categories 1 and 2, we also present results where we do not combine categories 1 and 2. As mentioned in Section 5.2, the similarity of these two categories makes discriminating between them a difficult endeavor. We thus expect to see rather lower classification rates in the performance matrices when comparing categories 1 and 2.

The mean and individual optimal five-way performance matrices are shown in Figures 5-13 and 5-14, respectively. As above, these matrices show the categorization rates both excluding (first percentage) and including (second percentage) border features for each category comparison. Again, if the border features either increase or have no effect on the categorization rate for a specific comparison, the associated cell is shaded.

In examining these matrices, we first note that we do indeed have rather low categorization rates between categories 1 and 2. More encouragingly, though, we see that for

Categories	1	2	3	4	5		1 vs. 3 vs. 5
1 vs.		48.4% 55.8%	74.1% 71.8%	69.3% 81.3%	88.8% 85.6%		72.7% 70.9%
2 vs.			61.4% 62.9%	60.0% 76.7%	82.3% 84.6%		1 vs. 2 vs. 3 vs.4 vs. 5
3 vs.				66.0% 60.0%	85.5% 81.8%		40.6% 40.9%
4 vs.					87.8% 84.4%		

Figure 5-13: Five-Way Mean Optimal Performance Matrix

Categories	1	2	3	4	5		1 vs. 3 vs. 5
1 vs.		58.4% 63.7%	80.0% 76.5%	85.4% 92.0%	95.0% 92.7%		76.4% 74.5%
2 vs.			67.7% 70.7%	86.7% 87.5%	93.8% 94.6%		1 vs. 2 vs. 3 vs.4 vs. 5
3 vs.				85.6% 84.4%	89.0% 93.0%		42.6% 44.7%
4 vs.					96.3% 94.4%		

Figure 5-14: Five-Way Individual Optimal Performance Matrix

both mean and individual optimality, the inclusion of the border features increases the ability of the classifier to distinguish between categories 1 and 2. These increases are not trivial; we observe a full 7.4% increase for mean optimality and 5.3% increase for individual optimality. In both cases, the border binary features are the significant contributors to this increase.

In addition to aiding the discrimination between categories 1 and 2, the inclusion of border features under mean optimality yields a 12% increase in comparing categories 1 and 4 and a full 16.7% increase in comparing categories 2 and 4. In comparison, the worst performance loss we suffer by including border features is only 6.0% in comparing categories 3 and 4. While we do not see as significant of performance gains under individual optimality, neither do we see significant degradation by including border features.

These improvements with the inclusion of the border features suggest that our revised border criterion, while not ideal, is at least sufficiently accurate to lead to some performance increases. However, in future work, it would be useful to consider other border features that capture the original criterion of sharpness and circularity. As mentioned in Section 5.1.2, methods to accomplish this exist in the literature, particularly within the field of melanoma detection and classification.

#### 5.4.4 Conclusions and Implementation

Overall, based on the high categorization rates under both mean and individual optimality when categories 1 and 2 are combined, the Tier 2 approach that we have proposed in this chapter represents a useful and successful methodology for performing quantitative stem cell colony categorization. For mean optimality, five of the six binary comparisons yield categorization rates above 80.0%. For individual optimality, all six binary rates exceed 80.0% with three beyond 90.0%. For the full four-way comparison using individual optimality, we are successful 68.8% of the time.

For the four category case, we observe two consistent colony-wise feature and SVM parameter trends. First, among the interior texture features, the binary and outlier features appear most significant. Secondly, the border features from the revised border criterion are generally not helpful in increasing categorization performance.

However, these border features *do* have a positive influence when we attempt to sepa-

rate categories 1 and 2. Their inclusion increases our ability to separate categories 1 and 2 by 7.4% under mean optimality and 5.3% under individual optimality. They also yield performance increases by up to 16.7% for some binary category comparisons with only marginal losses in others.

Based on this effective categorization ability of our Tier 2 methodology, we make the following recommendations for its practical implementation. We consider a stem cell researcher with a new collection of colony images that he wishes to categorize. He should first utilize the four category scenario and run the full four-way classification using its individual optimal setting:

- Window Size of  $45 \times 45$
- Training Set 5 (tight texture + loose texture)
- Border Binary Features
- Texture Confidence and Outlier Features
- Radial Basis Function Kernel for Tier 2 SVM

Based on the results above, he would expect a 68.8% correct categorization of his colonies into the four different categories for this run.

For any colonies classified to the composite “1/2” category, he should run the pairwise comparison between categories 1 and 2 of the five category case using the individual optimal setting for that comparison:

- Window Size of  $45 \times 45$
- Training Set 4 (tight texture + black texture)
- Border Binary Features
- Texture Outlier Features
- Tier 2 Polynomial of Degree 2 for Tier 2 SVM

He would expect to classify 63.7% of these colonies correctly into category 1 and category 2.

Based on these two runs, the researcher could expect to have a relatively good idea of the quality of each of his new colonies. If he wishes to make further pairwise comparison, he should employ the individual optimal settings for each of the desired comparisons.

# Chapter 6

## Conclusions and Future Work

This thesis makes two contributions to the general field of textural image segmentation and three contributions to the advancement of stem cell research. We summarize and make conclusions regarding the work performed in this thesis by referencing these five contributions.

- We demonstrate the applicability of using the non-parametric support vector machine classifier with wavelet energy features to the textural segmentation of images. We report a high success rate of 92.3% over a diverse set of benchmark textural images.

The ability of the wavelet decomposition to perform multiscale analysis of both the spatial and frequency components of images makes it one of the most sophisticated image processing tools currently available. Additionally, the non-parametric support vector machine classifier is highly attractive due to its strong theoretical justification and empirical performance across a wide range of applications. To the best of our knowledge, the combination of wavelet subband energy features with the SVM classifier for the task of textural image segmentation has not been previously studied. We introduce and test this particular pairing of feature representation and classification method by performing binary image segmentation on a set of benchmark textural images. We achieve a segmentation accuracy of 92.3% over the 18 benchmark images.

- We adapt a parametric wavelet-based texture classification method originally proposed for the task of content-based image retrieval to the task of textural image segmentation. We report a high classification rate of 91.8% over a diverse set of benchmark textural images.

To serve as a comparison to the non-parametric SVM classifier, we adapt for the task of textural image segmentation a parametric wavelet-based classification approach originally developed by Vetterli and Do for content-based image retrieval [11]. The method is parametric in the sense that it models the coefficients of a wavelet decomposition as realizations of a generalized Gaussian probability distribution. By estimating the distribution parameters at each wavelet subband, we can make classifications based on the Kullback-Leibler distance between distributions, resulting in a parametric image segmentation. On the same set of benchmark textural image as above, we achieve an accuracy of 91.8%.

- We introduce a two-tiered, hierarchical paradigm for the creation of a standard, automated, non-invasive stem cell quality control tool. Our implementation of this paradigm represents, to the best of our knowledge, the first known attempt at creating a routine, non-invasive method for measuring stem cell quality under conditions favorable to their growth.

We introduce this paradigm as a general framework through which we provide stem cell researchers with both qualitative and quantitative tools to aid in the stem cell quality control process. The two tiers that make up this hierarchical approach constitute the two additional stem cell research contributions described below.

- As the first tier of this new paradigm, we perform textural image segmentation on stem cell colony images to create a rich set of graphical aids to assist medical researchers in determining the quality of their colonies.

As neither the non-parametric SVM nor parametric KLD method significantly outperforms the other in the benchmark analysis, we employ both for the task of segmenting stem cell colony images into regions of varying quality. We use the binary pixel-by-pixel classifications of both methods to create binary reclassified colony images that pinpoint the areas of high and low quality within the interior regions of the colonies. Employing the confidence information implicit to the SVM classifier and derived in this thesis for the KLD classifier, we develop higher resolution confidence reclassified images and movies as additional visual representations of the colonies. We provide guidance for the practical implementation of these graphical representations into the stem cell quality control process.

- Employing inputs derived from the first tier, we introduce in the second tier a highly accurate method for making quantitative categorizations of stem cell colonies based on overall colony quality. Across a wide range of tests on real stem cell images, we attain an average performance of 80.0%.

To expand upon the qualitative graphical outputs discussed above, we develop a methodology for making quantitative classifications of full stem cell colonies into discrete quality categories. We extract colony-wise textural features from the graphical outputs and introduce new colony-wise border quality features. We then demonstrate the high categorization rates that can be achieved by using these colony-wise features as inputs to a multiclass support vector machine classifier. Across various runs, we achieve average categorization rates on the order of 80.0%.

## 6.1 Future Technical Challenges

Despite the contributions above, significant future challenges still exist in refining and expanding upon our technical work. We document the most critical of these.

- In theory, the feature vector inputs into machine learning classification methods should be *independent*. However, as mentioned in Section 2.2.2, the use of the sliding



neighborhood method for the extraction of textural features for each pixel in an image results in potentially *dependent* feature vector inputs. Thus, a theoretical and practical examination of the effects of employing dependent inputs would constitute a useful future research task.

- Referencing Tables 3.5 and 3.6 in the benchmark textural image analysis of Section 3.3.1, we notice that image pairs P5 and P16 are particularly difficult to segment. Further study into the reasons for this difficulty might yield useful insights into the performance abilities of the support vector machine and Kullback-Leibler distance classifiers.
- In both the benchmark study of textural image segmentation and in the application to stem cell imaging, we have considered only *binary* segmentation. An extension to *multiway* segmentation of benchmark images would provide an additional contribution to the general analysis of Chapter 3. Additionally, we could perform a multiway stem cell colony segmentation into the four types of texture discussed in Section 4.2.3. This would then lead to reduced Tier 2 colony-wise feature dimensionality by replacing the nine different training set features with a single multiway feature.
- While not particularly beneficial for the benchmark image segmentations of Chapter 3, the implementation of an adaptive textural window technique might lead to more accurate stem cell colony segmentations. As above, it would also reduce Tier 2 feature dimensionality as we would not need to consider all three window sizes separately.
- As discussed in Section 3.3.2, non-parametric classification methods such as the support vector machine require a significant amount of training data in order to achieve high classification rates. However, as mentioned in Section 5.2, the seven to ten colonies per category for use in training the multiclass Tier 2 SVM are likely not sufficient. If more stem cell colonies could be acquired, our methodology could be run with the additional data and new, hopefully higher, categorization rates calculated.
- When defining the colony-wise features in Sections 5.1.1 and 5.1.2, we frequently employed the mean operator. Since the mean statistic can be adversely affected by outliers, a more robust measure such as the median might be more appropriate, particularly for the calculation of the colony-wise interior texture outlier features  $\xi_{SVM}$  and  $\xi_{KLD}$ .
- In Section 5.3.2, we suggested several approaches to feature selection that could be used in the Tier 2 analysis to prune the excessive number of colony-wise features. While we chose a simple combinatorial approach, the study and implementation of the more advanced feature selection methods might lead to more effective categorization performance.

- In Section 5.4.3, we concluded that while our revised texture-based border criterion did lead to improved separation of stem cell colony categories 1 and 2, its overall effect on the Tier 2 analysis was marginal. Thus, we suggest a study of the original border criterion by implementing the various methods outlined in Section 5.1.2 for the direct quantification of border sharpness and circularity.
- Our two-tiered stem cell analysis methodology of Chapters 4 and 5 is presently only semi-automated. User inputs are required to define the colony border, extract the colony interior and identify the colony centroid. The implementation of an advanced, automated border segmentation procedure, such as that suggested in Section 5.1.2, would represent a necessary first step toward increasing automation.

## 6.2 Future Applications

Despite our focus on the application and extension of basic textural image segmentation methods to the field of stem cell image analysis, the ideas and methodologies presented in this thesis are applicable to a wide range of both medical and non-medical applications. In fact, any segmentation or classification task for which textural features are of interest would suffice as an area of future application.

Toward further stem cell analysis, we proposed in Section 4.1.4 the use of our Tier 2 methodology in providing quantitative measures of the effects of biological experiments performed on stem cell colonies. This application would increase experimental standardization and allow researchers to run many experiments without the time-consuming task of analyzing every result themselves. Additionally, it would represent an *active* involvement of our method in stem cell researchers' quests both for a deeper understanding of human physiology and for the development of novel medical treatments.

As we mentioned in Chapter 1, the classification of skin abnormalities as either cancerous melanomas or harmless lesion bears a close resemblance to the stem cell segmentation and categorization problem discussed in this thesis. The application of our methods toward the texture-based classification of this and other forms of cancer, such as colon and breast cancer, would also be beneficial.

Additionally, our two-tiered methodology could be applied to various non-medical applications in such diverse fields as remote sensing, machine vision in robots and intelligence gathering.

# Appendix A

## The Support Vector Machine

In this appendix, we describe in more detail the various formulations of the support vector machine classifier. We begin with the binary classification formulations, discussing first the simplest of these formulations before introducing increasingly more complex and powerful versions. Along the way, we discover additional insights into the unique behavior of the SVM classifier. After covering these binary formulations, we consider a framework for performing multiclass classification using the SVM.

This appendix is driven primarily by the implementation aspects of the SVM rather than the theoretical justification. A brief outline of the theory is provided in Section 2.3.2 of Chapter 2. Additional theoretical work can be found in the literature [5] [54].

### A.1 Binary Classification Formulations

As standard terminology for the various formulations to follow, let  $\mathbf{x}_i \in \mathfrak{R}^d$  be a  $d$ -dimensional *feature vector* representing some input object. Associated with each feature vector is a *label*  $y_i \in \{-1, +1\}$  dictating the *class* into which the object falls. For now, we limit  $y_i$  to only one of two classes, the  $-1$  class or  $+1$  class. We assume that we have  $N$  feature vectors  $\mathbf{x}_i$ ,  $i = 1, 2, \dots, N$ , comprising a *training data set* whose true class labels  $y_i$ ,  $i = 1, 2, \dots, N$ , are fully known. The goal of the SVM is to use these training points to find a *hyperplane*, or linear separator, that will separate the two classes. Furthermore, this hyperplane should be oriented such that it maximizes *margin*, or the distance from the hyperplane to the nearest point of either class.

If the convex hulls of the points from each class do not overlap in the original  $d$ -dimensional *input space*, the classes are considered *linearly separable* as they can be separated by a linear surface in the input space. If the convex hulls do overlap in the original space but *do not* overlap in some higher dimensional *feature space*, the classes are considered *nonlinearly separable*; in this case, the points can be separated by a linear surface in the feature space that maps to a nonlinear separator in the input space. If we require all training points to be correctly classified, we can create a *hard margin separator* to accomplish this. However, we can also create a *soft margin separator* that allows some training points to be misclassified. The formulations below capture these different situations.

Unless otherwise noted, all of the formulations below follow the terminology and general methodology of Edgar Osuna [35].

### A.1.1 Linear Primal Formulation

The first two formulations that we consider, the *linear primal* and *linear dual*, create hard margin separators of linearly separable input points. That is, we assume that the two classes under consideration can be separated without error by a hyperplane in the original input space. We define this hyperplane as all points  $\mathbf{x}$  such that,

$$\mathbf{w} \cdot \mathbf{x} + b = 0 \quad (\text{A.1})$$

where  $\mathbf{w} \in \mathfrak{R}^d$  dictates the orientation of the hyperplane with respect to the origin and  $b \in \mathfrak{R}$  is a scalar bias term.

To enforce the hard margin property of the formulation, we require that all training points  $\mathbf{x}_i$ ,  $i = 1, 2, \dots, N$ , satisfy the constraints,

$$\mathbf{w} \cdot \mathbf{x}_i + b \geq 1 \quad \forall i \text{ such that } y_i = +1 \quad (\text{A.2})$$

$$\mathbf{w} \cdot \mathbf{x}_i + b \leq -1 \quad \forall i \text{ such that } y_i = -1 \quad (\text{A.3})$$

Recalling that  $y_i = 1$  for all points in class +1 and  $y_i = -1$  for all points in class -1, we can write the above expressions together as,

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 \quad \forall i = 1, 2, \dots, N \quad (\text{A.4})$$

The distance from a point  $\mathbf{x}_i$  to the hyperplane is  $\frac{|\mathbf{w} \cdot \mathbf{x}_i + b|}{\|\mathbf{w}\|}$ . If we impose the normalization,

$$\min_{i=1, \dots, N} |\mathbf{w} \cdot \mathbf{x}_i + b| = 1 \quad (\text{A.5})$$

the distance from the hyperplane to the nearest point is simply  $\frac{1}{\|\mathbf{w}\|}$ . If we think of our problem of maximizing the margin as finding the hyperplane that is farthest from the closest data point from either class while ensuring the separation criterion above, we can pose the following problem:

$$\begin{aligned} & \text{maximize}_{\mathbf{w}, b} && \frac{1}{\|\mathbf{w}\|} \\ & \text{s.t.} && y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1, \quad i = 1, \dots, N \end{aligned}$$

Noting that maximizing  $\frac{1}{\|\mathbf{w}\|}$  is equivalent to minimizing  $\frac{1}{2}\|\mathbf{w}\|^2$ , we have the following quadratic optimization problem as the linear primal SVM formulation:

$$\begin{aligned} & \text{minimize}_{\mathbf{w}, b} && \frac{1}{2}\|\mathbf{w}\|^2 \\ & \text{s.t.} && y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1, \quad i = 1, \dots, N \end{aligned}$$

Solving this problem returns the optimal pair  $(\mathbf{w}^*, b^*)$  from which we create the classification functions,

$$h(\mathbf{x}) = \mathbf{w}^* \cdot \mathbf{x} + b^* \quad (\text{A.6})$$

$$f(\mathbf{x}) = \text{sign}(\mathbf{w}^* \cdot \mathbf{x} + b^*) \quad (\text{A.7})$$

for a new data point  $\mathbf{x}$ .

Geometrically,  $h(\mathbf{x})$  returns the signed Euclidean distance from the new point  $\mathbf{x}$  to the optimal hyperplane, and  $f(\mathbf{x})$  returns the binary  $+1$  or  $-1$  classification of  $\mathbf{x}$ . We interpret  $h(\mathbf{x})$  as the confidence we have in our binary decision. Larger positive/negative values of  $h(\mathbf{x})$  imply higher confidence in our corresponding  $+1/-1$  decision.

### A.1.2 Linear Dual Formulation

While the primal formulation above will yield an optimal hyperplane, we find it instructive to consider the dual problem as well. If we let  $\lambda_i$  be the Lagrangian multiplier for the  $i$ th constraint in the primal problem, we have the Lagrangian function,

$$L(\mathbf{w}, b, \lambda) = \frac{1}{2} \|\mathbf{w}\|^2 - \sum_{i=1}^N \lambda_i [y_i (\mathbf{w} \cdot \mathbf{x}_i + b) - 1] \quad (\text{A.8})$$

Differentiating with respect to  $\mathbf{w}$  and  $b$  and setting the partial derivatives equal to zero, we have,

$$\frac{\partial L(\mathbf{w}, b, \lambda)}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^N \lambda_i y_i \mathbf{x}_i = 0 \quad (\text{A.9})$$

$$\frac{\partial L(\mathbf{w}, b, \lambda)}{\partial b} = \sum_{i=1}^N \lambda_i y_i = 0 \quad (\text{A.10})$$

From Equation A.9 we find the optimal value for  $\mathbf{w}$  to be,

$$\mathbf{w}^* = \sum_{i=1}^N \lambda_i^* y_i \mathbf{x}_i \quad (\text{A.11})$$

where the  $\lambda_i^*$  parameters are the optimal multipliers to be gleaned from the solution to the dual problem. Substituting this expression for  $\mathbf{w}$  and the statement  $\sum_{i=1}^N \lambda_i y_i = 0$  from Equation A.10 into Equation A.8, we have the simplified Lagrangian as a function only of the dual multipliers:

$$L(\lambda) = \sum_{i=1}^N \lambda_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \lambda_i \lambda_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j \quad (\text{A.12})$$

Maximizing this Lagrangian subject to the non-negativity constraint on the multipliers and Equation A.10 above, we have the following dual formulation:

$$\begin{aligned} \text{maximize}_{\lambda} \quad & \sum_{i=1}^N \lambda_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \lambda_i \lambda_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j \\ \text{s.t.} \quad & \sum_{i=1}^N \lambda_i y_i = 0 \\ & \lambda_i \geq 0, \quad i = 1, \dots, N \end{aligned}$$

If we let  $\Lambda = (\lambda_1, \lambda_2, \dots, \lambda_N)$  be the vector of multipliers,  $D$  be a symmetric matrix

with entries  $D_{ij} = y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j$  and  $\mathbf{1}$  be a vector of ones, we can re-write the above formulation in matrix form as,

$$\begin{aligned} & \text{maximize}_{\Lambda} && \Lambda \cdot \mathbf{1} - \frac{1}{2} \Lambda \cdot D \cdot \Lambda \\ \text{s.t.} &&& \Lambda \cdot y &= 0 \\ &&& \Lambda &\geq 0 \end{aligned}$$

Using duality theory, we can derive expressions for the optimal primal variables  $\mathbf{w}^*$  and  $b^*$  in terms of the optimal multipliers returned by the solution to the dual formulation. In fact, we already have such an expression for  $\mathbf{w}^*$  in Equation A.11 above. Additionally, the complementary slackness condition says that,

$$\lambda_i^* [y_i (\mathbf{w}^* \cdot \mathbf{x}_i + b^*) - 1] = 0 \quad \forall i = 1, 2, \dots, N \quad (\text{A.13})$$

which we expand and simplify to find,

$$b^* = \frac{1 - y_i \mathbf{w}^* \cdot \mathbf{x}_i}{y_i} = y_i - \mathbf{w}^* \cdot \mathbf{x}_i \quad (\text{A.14})$$

for any index  $i$  such that  $\lambda_i^* > 0$ . While we can use any  $y_i$  and  $\mathbf{x}_i$  for which  $\lambda_i^* > 0$  to calculate  $b^*$  in Equation A.14, we obtain a more stable estimate by averaging the  $b^*$  values calculated over all  $i$  for which  $\lambda_i^* > 0$  [20]. Thus, we can find the optimal pair  $(\mathbf{w}^*, b^*)$  easily from the dual solution and write the decision functions of Equations A.6 and A.7 as,

$$h(\mathbf{x}) = \sum_{i=1}^N y_i \lambda_i^* (\mathbf{x} \cdot \mathbf{x}_i) + b^* \quad (\text{A.15})$$

$$f(\mathbf{x}) = \text{sign} \left( \sum_{i=1}^N y_i \lambda_i^* (\mathbf{x} \cdot \mathbf{x}_i) + b^* \right) \quad (\text{A.16})$$

The complementary slackness condition of Equation A.13 also yields one of the unique features of the SVM, namely that the resulting hyperplane is fully specified by only a (generally small) subset of the training points. Note that if a point  $\mathbf{x}_i$  satisfies the primal constraint  $y_i (\mathbf{w}^* \cdot \mathbf{x}_i + b^*) \geq 1$  with strict inequality, in order to satisfy the complementary slackness condition,

$$\lambda_i^* [y_i (\mathbf{w}^* \cdot \mathbf{x}_i + b^*) - 1] = 0 \quad (\text{A.17})$$

we must have  $\lambda_i^* = 0$ . Recalling that  $\mathbf{w}^* = \sum_{i=1}^N \lambda_i^* y_i \mathbf{x}_i$ , we see that points for which  $\lambda_i^* = 0$  do not have any effect on the resulting hyperplane. Geometrically, the points  $\mathbf{x}_i$  with this property are those whose distance to the hyperplane exceeds the optimal margin. Thus, we find that only those points lying on the margin have non-zero dual multipliers and are active in defining the optimal hyperplane. We call these points *support vectors*, and the set of support vectors is generally significantly smaller than the full set of training points. This results in a concise representation of the optimal hyperplane as a function of only a small set of points. Additionally, it allows for the derivation of Osuna's efficient *active set algorithm* for the dual problem [35].

### A.1.3 Soft Margin Primal and Dual Formulations

Suppose we now drop the assumption that the two classes are completely separable by a hyperplane in the original input space. In this case, we create a soft margin separator in the input space that allows some training points to be misclassified. However, we impose a penalty in the objective function when such misclassifications occur. Let  $\xi_i \geq 0$  represent the amount of violation of the primal constraint  $y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1$  by each point  $\mathbf{x}_i$ . Additionally, specify a scalar  $C$  such that a penalty of  $C \sum_{i=1}^N \xi_i$  is added to the objective function. Then, the primal problem simply becomes,

$$\begin{aligned} & \text{minimize}_{\mathbf{w}, b, \xi} && \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^N \xi_i \\ & \text{s.t.} && y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 - \xi_i, \quad i = 1, \dots, N \\ & && \xi_i \geq 0, \quad i = 1, \dots, N \end{aligned}$$

As before, solving this problem returns the pair  $(\mathbf{w}^*, b^*)$  that defines the optimal hyperplane. This soft margin separator allows some training points to be misclassified in order to yield a larger margin. The chosen value of  $C$  governs this tradeoff between misclassified training points and margin.

We can also derive the dual formulation,

$$\begin{aligned} & \text{maximize}_{\Lambda} && \Lambda \cdot \mathbf{1} - \frac{1}{2} \Lambda \cdot D \cdot \Lambda \\ & \text{s.t.} && \Lambda \cdot y = 0 \\ & && \Lambda \leq C \mathbf{1} \\ & && \Lambda \geq 0 \end{aligned}$$

where, as above,

$$\mathbf{w}^* = \sum_{i=1}^N \lambda_i^* y_i \mathbf{x}_i \tag{A.18}$$

$$b^* = y_i - \mathbf{w}^* \cdot \mathbf{x}_i \tag{A.19}$$

and the decision functions are,

$$h(\mathbf{x}) = \sum_{i=1}^N y_i \lambda_i^* (\mathbf{x} \cdot \mathbf{x}_i) + b^* \tag{A.20}$$

$$f(\mathbf{x}) = \text{sign} \left( \sum_{i=1}^N y_i \lambda_i^* (\mathbf{x} \cdot \mathbf{x}_i) + b^* \right) \tag{A.21}$$

We notice that the only difference between this soft margin dual formulation and the hard margin dual formulation of Section A.1.2 is that the dual parameters are now bounded above by the cost parameter  $C$ .

### A.1.4 Nonlinear Soft Margin Dual Formulation

Up to this point, we have worked only in the original  $d$ -dimensional input space. If the training points could be separated without error, we derived a hard margin separator. If

we could not separate the points without error, we formed a soft margin separator that permitted misclassification at a cost. An alternative approach to this second situation involves projecting the input data into a higher dimensional feature space and finding a hyperplane there that can separate the points without error. This hyperplane in the feature space will map to a nonlinear separator in the original space. If we additionally allow some points to be misclassified in the feature space in order to create a large margin, we arrive at the nonlinear soft margin formulation. We consider the dual version of this problem below.

We begin by mapping the input points to some higher, possibly infinite dimensional space  $\mathcal{F}$  via the explicit mapping  $\phi : \mathbb{R}^d \rightarrow \mathcal{F}$ . We note here that the input points  $\mathbf{x}_i$  appear only in dot product expressions in the linear dual formulation via the matrix  $D$  where  $D_{ij} = y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j$ , and in the resulting classification functions of Equations A.15 and A.16. Thus, we can replace each instance of the dot product  $\mathbf{x}_i \cdot \mathbf{x}_j$  with  $\phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j)$  to solve the dual problem in the higher dimension feature space [5].

Implementing this explicit mapping procedure, however, presents a few difficulties. First, it may be difficult to actually find a useable expression for the mapping  $\phi$ . Secondly, the operation  $\phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j)$  involves direct computation in the higher dimensional space. If the dimensionality of this space is significant, this operation can be computationally consuming. To get around these problems, we employ a *kernel function*  $K$  that implicitly computes the high dimensional dot product  $\phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j)$  as the operation  $K(\mathbf{x}_i, \mathbf{x}_j)$  in the original input space. This solves the problem of finding an explicit mapping and allows us to make all computations in the lower dimensional input space. Three common kernel functions include [5],

**Polynomial Kernel (degree  $p$ ):**  $K(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i \cdot \mathbf{x}_j + 1)^p$

**Gaussian Radial Basis Function Kernel:**  $K(\mathbf{x}_i, \mathbf{x}_j) = e^{-\|\mathbf{x}_i - \mathbf{x}_j\|^2 / 2\sigma^2}$

**Sigmoidal Neural Network Kernel:**  $K(\mathbf{x}_i, \mathbf{x}_j) = \tanh(\kappa \mathbf{x}_i \cdot \mathbf{x}_j - \delta)$

To implement this kernel-based approach, we simply replace each instance of the dot product  $\mathbf{x}_i \cdot \mathbf{x}_j$  in the soft margin dual formulation with  $K(\mathbf{x}_i, \mathbf{x}_j)$ . This yields the following nonlinear soft margin dual formulation:

$$\begin{aligned} \text{maximize}_{\Lambda} \quad & \Lambda \cdot \mathbf{1} - \frac{1}{2} \Lambda \cdot D \cdot \Lambda \\ \text{s.t.} \quad & \Lambda \cdot \mathbf{y} = 0 \\ & \Lambda \leq C \mathbf{1} \\ & \Lambda \geq \mathbf{0} \end{aligned}$$

where  $D$  is now the symmetric matrix with entries  $D_{ij} = y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$ . The classification functions are now,

$$h(\mathbf{x}) = \sum_{i=1}^N y_i \lambda_i^* K(\mathbf{x}, \mathbf{x}_i) + b^* \quad (\text{A.22})$$

$$f(\mathbf{x}) = \text{sign} \left( \sum_{i=1}^N y_i \lambda_i^* K(\mathbf{x}, \mathbf{x}_i) + b^* \right) \quad (\text{A.23})$$



We can think of this formulation as the master formulation since it is the most powerful and expressive of the various SVM formulations. In fact, all the other formulations described above can be derived from it with proper choices for the kernel function  $K$  and the cost parameter  $C$ .

## A.2 Multiclass Formulations

We now depart from the basic binary classification problem and consider the multiclass problem. As with the binary classification problem, let  $\mathbf{x}_i \in \mathfrak{R}^d$  be a  $d$ -dimensional *feature vector* representing some input object, and let its *label*  $y_i \in \{1, 2, \dots, k\}$  dictate the *class* into which the object falls. Note, however, that we now allow the object to fall into one of  $k$  classes rather than just the  $-1$  and  $+1$  classes. As above, we assume that we have  $N$  feature vectors  $\mathbf{x}_i$ ,  $i = 1, 2, \dots, N$ , comprising a *training data set* whose true class labels  $y_i$ ,  $i = 1, 2, \dots, N$ , are fully known.

We describe the multiclass method that we employ in this thesis, the *one-vs-all* formulation, as a specific implementation of the general multiclass approach known as *error correcting output coding* (ECOC)<sup>1</sup>.

### A.2.1 Error Correcting Output Codes

Error correcting output coding provides a systematic procedure for solving multiclass classification problems. One useful feature of this approach is that ECOC simply reduces the multiclass problem to a series of binary classification problems; thus, one need only understand the basic binary classification problem to implement this multiclass approach. Another advantage of ECOC is that it is a very general framework; in fact, any suitable binary classification method can be used to solve the underlying binary classification problems. Example classifiers that have been used with ECOC include the naive Bayes classifier [2] and the support vector machine [41]. As we have focused on the SVM in this thesis, we discuss ECOC in the context of the SVM.

Since ECOC reduces our problem to a series of binary classifications, we must first partition the original multiclass data into data sets on which we can apply the binary SVM. In other words, we reduce the multiclass data down to two classes by grouping some of the original classes together. ECOC defines a binary *code matrix*  $\mathbf{C}$  that holds the information regarding these partitions. The  $\mathbf{C}$  matrix is  $k \times n$  where  $k$  is the number of classes in the original multiclass problem and  $n$  is the number of binary classifications to which we reduce the multiclass problem. The  $j$ th row of  $\mathbf{C}$  corresponds to a *codeword* for label  $j$ . For example, a code matrix for the multiclass problem considered in the stem cell application of Chapter 5 might take the form,

---

<sup>1</sup>The work in this section was originally written for a term project in the MIT course 6.867, *Machine Learning*. The work was performed jointly with MIT student Ramsay Key.

Stem Cell Category 1	1	0	1	0	0	1
Stem Cell Category 2	0	1	0	0	1	0
Stem Cell Category 3	1	0	0	1	1	1
Stem Cell Category 4	1	1	1	1	0	0
Stem Cell Category 5	1	0	1	1	0	0

Recall that each column  $v$ ,  $v \in \{1, \dots, n\}$ , of the code matrix represents a different binary classification problem. For the  $v$ th binary classification, we group together those classes with a 1 in the  $v$ th column and consider this composite data as the +1 class. We then group together those classes with a 0 in the  $v$ th column and consider this composite data as the  $-1$  class. We can then train an SVM for each of these  $n$  composite data sets to create  $n$  binary classification functions  $(f_1(\mathbf{x}), \dots, f_n(\mathbf{x}))$  and  $n$  confidence value functions  $(h_1(\mathbf{x}), \dots, h_n(\mathbf{x}))$ .

Given a new input  $\mathbf{x}$  to be classified, we calculate  $n$  confidence values by inputting  $\mathbf{x}$  into the  $n$  confidence value functions to yield the *confidence vector*  $(h_1(\mathbf{x}), \dots, h_n(\mathbf{x}))$ . We use this confidence vector to produce a solution to the original multiclass problem via the multiclass classifier [41],

$$H(\mathbf{x}) = \arg \min_{j \in \{1, \dots, k\}} \sum_{v=1}^n g(h_v(\mathbf{x}) \mathbf{C}_{jv}) \quad (\text{A.24})$$

where  $\mathbf{C}_{jv}$  is the  $v$ th entry of the  $j$ th row of the code matrix  $\mathbf{C}$  and  $g : \mathfrak{R} \rightarrow \mathfrak{R}$  is a user-defined *loss function*. We follow previous work and choose to use the linear loss function  $g(x) = -x$  [41]. With this specific loss function, the multiclass classifier  $H(\mathbf{x})$  looks at each possible multiclass classification for  $\mathbf{x}$ , sums up the confidences from the SVM that we achieve for each class (which is based on the structure of the coding matrix) and chooses the class with the highest overall confidence.

Obviously, the multiclass classification function is highly dependent on the form of the coding matrix, whose choice is left up to the user. One common option is a random matrix where each 0/1 entry in the coding matrix is governed by the flip of a fair coin. Another common option is to use BCH codes, which have high column and row separation [41]. A simpler option, and the one employed in this work, is the one-vs-all matrix; this is simply the  $k \times k$  identity matrix.

## A.2.2 One-vs-All Formulation

We now turn more specifically to our chosen coding matrix, the  $k \times k$  identity matrix, and explore how this matrix fits within the ECOC framework. Since the  $v$ th column of the identity coding matrix holds a 1 in the  $v$ th row and zeros elsewhere, when we train the binary SVM for the  $v$ th column, we are simply training the  $v$ th class versus a composite data set of all the other classes. This yields  $k$  one-vs-all SVMs; one for each of the  $k$  classes. Thus, to classify a new point  $\mathbf{x}$ , our multiclass classification function using the

identity coding matrix and linear loss function simplifies as,

$$H(\mathbf{x}) = \arg \min_{j \in \{1, \dots, k\}} \sum_{v=1}^n g(h_v(\mathbf{x}) \mathbf{C}_{jv}) \quad (\text{A.25})$$

$$= \arg \min_{j \in \{1, \dots, k\}} g(h_j(\mathbf{x}) C_{jj}) \quad (\text{A.26})$$

$$= \arg \min_{j \in \{1, \dots, k\}} -h_j(\mathbf{x}) \quad (\text{A.27})$$

$$= \arg \max_{j \in \{1, \dots, k\}} h_j(\mathbf{x}) \quad (\text{A.28})$$

Thus, we simply classify  $\mathbf{x}$  to the class whose associated one-vs-all SVM yields the highest confidence. By choosing such a simple coding matrix, we are perhaps sacrificing some performance that we might gain from employing a more complex matrix. However, this one-vs-all coding matrix has been shown to perform well in practice [41].

[This page intentionally left blank.]

# Bibliography

- [1] *Image Processing Toolbox User's Guide*. The MathWorks, Inc., Natick, MA, Third edition, 2001.
- [2] A. Berger. Error-correcting output coding for text classification. In *IJCAI'99: Workshop on Machine Learning for Information Filtering*, 1999.
- [3] Alan C. Bovick, Marianna Clark, and Wilson S. Geisler. Multichannel texture analysis using localized spatial filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(1):55–73, January 1990.
- [4] P. Brodatz. *Textures: A Photographic Album for Artists and Designers*. Dover, New York, 1966.
- [5] C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.
- [6] Tianhorng Change and C. C. Jay Kuo. Texture analysis and classification with tree-structured wavelet transform. *IEEE Transactions on Image Processing*, 2(4):429–441, October 1993.
- [7] R. Chellappa, R. L. Kashyap, and B. S. Manjunath. Model-based texture segmentation and classification. In C. H. Chen, L. F. Pau, and P. S. P. Wang, editors, *Handbook of Pattern Recognition & Computer Vision*, pages 249–282. World Scientific, Singapore, Second edition, 1999.
- [8] G. Van de Wouwer, P. Scheunders, and D. Van Dyck. Statistical texture characterization from discrete wavelet representation. *IEEE Transactions on Image Processing*, 8(4):592–598, April 1999.
- [9] Gert Van de Wouwer. *Wavelets for Multiscale Texture Analysis*. PhD thesis, University of Antwerp, 1998.
- [10] Thomas G. Dietterich and Ghulum Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, 2:263–286, 1995.
- [11] Minh N. Do and Martin Vetterli. Wavelet-based texture retrieval using generalized Gaussian density and Kullback-Leibler distance. *IEEE Transactions on Image Processing*, 11(2):146–158, February 2002.

- [12] M. Kociolek, A. Materka, M. Strzelecki, and P. Szczypiński. Discrete wavelet transform—derived features for digital image texture analysis. In *Proc. International Conference on Signals and Electronic Systems*, pages 163–168, Lodz, Poland, September 2001.
- [13] Mary M. Galloway. Texture analysis using gray level run lengths. *Computer Graphics and Image Processing*, 4:172–179, 1975.
- [14] Armin Gerger, Wilhelm Stolz, Rene Pompl, and Josef Smolle. Automated epiluminescence microscopy—tissue counter analysis using CART and 1-NN in the diagnosis of melanoma. *Skin Research and Technology*, 9:105–110, 2003.
- [15] Costantino Grana, Giovanni Pellacani, Rita Cucchiara, and Stefania Seidenari. A new algorithm for border description of polarized light surface microscopic images of pigmented skin lesions. *IEEE Transactions on Medical Imaging*, 22(8):959–964, August 2003.
- [16] Heinz Handels, Th. Roß, J. Kreuzsch, H. H. Wolff, and S. J. Poppl. Feature selection for optimized skin tumor recognition using genetic algorithms. *Artificial Intelligence in Medicine*, 16(3):283–297, July 1999.
- [17] Robert M. Haralick. Statistical and structural approaches to texture. In *Proc. IEEE*, volume 67, pages 786–804, 1979.
- [18] Robert M. Haralick, K. Shanmugam, and Its’Hak Dinstein. Textural features for image classification. *IEEE Transactions on Systems, Man and Cybernetics*, 3(6):610–621, 1973.
- [19] Peter E. Hart, Richard O. Duda, and David G. Stork. *Pattern Classification*. John Wiley & Sons, Inc., New York, Second edition, 2001.
- [20] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, New York, 2001.
- [21] Lothar Hermes and Joachim M. Buhmann. Feature selection for support vector machines. In *Proc. of the International Conference on Pattern Recognition (ICPR’00)*, volume 2, pages 716–719, 2000.
- [22] Anil K. Jain. *Fundamentals of Digital Image Processing*. Prentice Hall, Englewood Cliffs, NJ, 1989.
- [23] Anil K. Jain and Farshid Farrokhnia. Unsupervised texture segmentation using Gabor filters. *Pattern Recognition*, 24(12):1167–1186, 1991.
- [24] T. Joachims. Making large-scale support vector machine learning practical. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods: Support Vector Machines*. MIT Press, Cambridge, MA, 1998.

- [25] Kwang In Kim, Keechul Jung, Se Hyun Park, and Hang Joon Kim. Support vector machines for texture classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(11):1542–1550, November 2002.
- [26] Andrew Laine and Jian Fan. Texture classification by wavelet packet signatures. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(11):1186–1191, November 1993.
- [27] K. I. Laws. Rapid texture identification. In *Proc. SPIE Image Processing for Missile Guidance*, volume 238, pages 376–380, 1980.
- [28] Tim K. Lee. *Measuring Border Irregularity And Shape Of Cutaneous Melanocytic Lesions*. PhD thesis, Simon Fraser University, January 2001.
- [29] W. Y. Ma and B. S. Manjunath. A comparison of wavelet transform features for texture image annotation. In *Second International Conference on Image Processing, ICIP 95*, volume 2, pages 256–259, October 1995.
- [30] Jianchang Mao and Anil K. Jain. Texture classification and segmentation using multiresolution simultaneous autoregressive models. *Pattern Recognition*, 25(2):173–188, 1992.
- [31] Michel Misiti, Yves Misiti, Georges Oppenheim, and Jean-Michel Poggi. *Wavelet Toolbox User's Guide*. The MathWorks, Inc., Natick, MA, 1996.
- [32] P. Moreno and R. Rifkin. Using the Fisher kernel method for web audio classification. In *International Conference on Acoustics, Speech and Signal Processing*, Istanbul, Turkey, 2000.
- [33] National Institutes of Health. Stem cell information. <http://stemcells.nih.gov/index.asp>, September 2002.
- [34] S. Osowski, T. Markiewicz, M. Basa, and T. H. Linh. SVM network for texture recognition. In *International Conference in Signals and Electronic Systems*, Wroclaw, Poland, 2002.
- [35] Edgar Osuna. *Support Vector Machines: Training and Applications*. PhD thesis, Massachusetts Institute of Technology, 1998.
- [36] Edgar Osuna, Robert Freund, and Federico Girosi. Training support vector machines: An application to face detection. In *1997 Conference on Computer Vision and Pattern Recognition*, Puerto Rico, June 1997.
- [37] H. Pien, M. Desai, and J. Shah. Segmentation of MR images using curve evolution and prior information. *International Journal of Pattern Recognition and Artificial Intelligence*, 11(8):1233–1245, 1997.
- [38] K. Rajpoot and N. Rajpoot. Wavelet based segmentation of hyperspectral colon tissue imagery. In *Proc. 7th IEEE International Multitopic Conference*, Islamabad, Pakistan, December 2003.

- [39] Trygve Randen and John Hakon Husøy. Filtering for texture classification: a comparative study. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(4):291–309, April 1999.
- [40] Todd R. Reed and J. M. Hans du Buf. A review of recent texture segmentation and feature extraction techniques. *CVGIP: Image Understanding*, 57(3):359–372, May 1993.
- [41] Jason D. M. Rennie and Ryan Rifkin. Improving multiclass text classification with the support vector machine. Technical report, Massachusetts Institute of Technology, 2002. Revised AI Memo-2001-026.
- [42] Ryan Rifkin. *Everything Old is New Again: A Fresh Look at Historical Approaches in Machine Learning*. PhD thesis, Massachusetts Institute of Technology, September 2002.
- [43] P. Rubegni, A. Ferrari, G. Cevenini, D. Piccolo, M. Burrioni, R. Perotti, K. Peris, P. Taddeucci, M. Biagioli, G. Dell’Eva, S. Chimenti, and L. Andreassi. Differentiation between pigmented spitz naevus and melanoma by digital dermoscopy and stepwise logistic discriminant analysis. *Melanoma Research*, 11(1):37–44, February 2001.
- [44] Yong Rui, Thomas Huang, and Shih-Fu Chang. Image retrieval: Current techniques, promising directions and open issues. *Journal of Visual Communication and Image Representation*, 10(4):39–62, April 1999.
- [45] Bernhard Scholkopf, Kah-Kay Sung, Chris Burges, Federico Girosi, Partha Niyogi, Tomaso Poggio, and Vladimir Vapnik. Comparing support vector machines with Gaussian kernels to radial basis function classifiers. *IEEE Transactions on Signal Processing*, 45(11):2758–2765, November 1997.
- [46] Anton Schwaighofer. Matlab interface to SVMlight, version v0.92. <http://www.cis.tugraz.at/igi/aschwaig/software.html>, August 2002.
- [47] Jayant Shah. A common framework for curve evolution, segmentation and anisotropic diffusion. In *Proc. IEEE Conference on Computer Vision and Patter Recognition*, San Francisco, 1996.
- [48] R. Sivaramakrishna, K. Powell, M. Lieber, W. Chilcote, and R. Shekhar. Texture analysis of lesions in breast ultrasound images. *Computerized Medical Imaging and Graphics*, 26(5):303–307, September–October 2002.
- [49] J. R. Smith and S. F. Chang. Transform features for texture classification and discrimination in large image databases. In *Proc. IEEE International Conference on Image Processing (ICIP-94)*, Austin, TX, November 1994.
- [50] Gilbert Strang and Truong Nguyen. *Wavelets and Filter Banks*. Wellesley-Cambridge Press, Wellesley, MA, 1997.



- [51] Fumitake Takahashi and Shigeo Abe. Decision-tree-based multiclass support vector machines. In *Proc. 9th International Conference on Neural Network Information Processing*, volume 3, pages 1418–1422, November 2002.
- [52] Mihran Tuceryan and Anil K. Jain. Texture analysis. In C. H. Chen, L. F. Pau, and P. S. P. Wang, editors, *Handbook of Pattern Recognition & Computer Vision*, volume 2, pages 207–248. World Scientific, Singapore, 1999.
- [53] Michael Unser. Sum and difference histograms for texture classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(1):118–125, January 1986.
- [54] Vladimir N. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, New York, Second edition, 2000.
- [55] MIT Vision and Modeling Group. Vistex database. <http://www-white.media.mit.edu/vismod/imagery/VisionTexture/vistex.html>, 1995.
- [56] Thomas Wagner. Texture analysis. In Bernd Jahne, Horst Haussecker, and Peter Geissler, editors, *Handbook of Computer Vision and Applications*. Academic Press, San Diego, 1999.
- [57] J. Weston, S. Mukherjee, O. Chapelle, M. Pontil, T. Poggio, and V. Vapnik. Feature selection for SVMs. In Sara A Solla, Todd K Leen, and Klaus-Robert Muller, editors, *Advances in Neural Information Processing Systems 13*. MIT Press, Cambridge, MA, 2001.