2007

A Method Of Content-based Image Retrieval For The Generation Of Image Mosaics

Michael Snead
University of Central Florida

A METHOD OF CONTENT-BASED IMAGE RETRIEVAL FOR THE GENERATION OF IMAGE MOSAICS

by

MICHAEL CHRISTOPHER SNEAD
B.S. University of Central Florida, 2005

A thesis submitted in partial fulfillment of the requirements
for the degree of Master of Science
in the School of Electrical Engineering and Computer Science
in the College of Engineering & Computer Science
at the University of Central Florida
Orlando, Florida

Spring Term
2007

ABSTRACT

An image mosaic is an artistic work that uses a number of smaller images creatively combined together to form another larger image. Each building block image, or tessera, has its own distinctive and meaningful content, but when viewed from a distance the tesserae come together to form an aesthetically pleasing montage. This work presents the design and implementation of MosaiX, a computer software system that generates these image mosaics automatically. To control the image mosaic creation process, several parameters are used within the system. Each parameter affects the overall mosaic quality, as well as required processing time, in its own unique way. A detailed analysis is performed to evaluate each parameter individually.

Additionally, this work proposes two novel ways by which to evaluate the quality of an image mosaic in a quantitative way. One method focuses on the perceptual color accuracy of the mosaic reproduction, while the other concentrates on edge replication. Both measures include preprocessing to take into account the unique visual features present in an image mosaic. Doing so minimizes quality penalization due the inherent properties of an image mosaic that make them visually appealing.

TABLE OF CONTENTS

LIST OF FIGURES vii

LIST OF TABLES xii

CHAPTER ONE: INTRODUCTION 1

Mosaic Image History 1

Digital Image Mosaics and Image Retrieval 3

MosaiX 4

Thesis Organization 6

CHAPTER TWO: CONTENT-BASED IMAGE RETRIEVAL 7

Motivation for CBIR 7

General CBIR Theory 8

Color-based CBIR 9

Color Representation 10

The RGB Color Space 12

The CMY & CMYK Color Spaces 13

The HSV & HSL Color Spaces 15

The YIQ & YUV Color Spaces 16

The CIE L*a*b* Color Space 17

Color-based CBIR Methodology and Research 18

Color Histogram-based CBIR Methods 18

Color Moment-based CBIR Methods 22

Color Correlogram-based CBIR Methods 26

Texture-based CBIR 28
Texture Representation 28
Edge-Based Texture Measures 29
Local Binary Partitions 30
Laws Texture Energy Measures 31
Texture-based CBIR Methodology and Research 32
Shape/Object-based CBIR 34
CHAPTER THREE: MOSAIX 36
The MosaiX System 38
Preprocessing Phase 39
Resolution Matching 40
Feature Extraction 41
Feature Vector Clustering 44
Products of Preprocessing 45
Image Mosaic Generation Phase 46
Tessellation 47
Image Retrieval 48
Mosaic Assembly 50
Products of Image Mosaic Generation 51
CHAPTER FOUR: EXPERIMENTAL RESULTS 52
Measurements and Methods 52
Experiments 56
Impact of Color Space 59

Color Accuracy 59
Edge Accuracy 61
Processing Time 62
Impact of Number of Clusters 63
Color Accuracy 64
Edge Accuracy 66
Processing Time 68
Impact of Image Library Size 70
Color Accuracy 71
Edge Accuracy 73
Processing Time 75
CHAPTER FIVE: CONCLUSIONS AND FUTURE WORK 77
The Impact of Preprocessing Parameters 77
Future Work 79
APPENDIX: EXAMPLE IMAGE MOSAICS 80
LIST OF REFERENCES 106

LIST OF FIGURES

Figure 1: Mosaic Image of Abraham Lincoln [Dali]. 2

Figure 2: Lighthouse image mosaic created by Robert Silvers' system, Photomosaic [1]..... 3

Figure 3: An example image mosaic generated with MosaiX. 5

Figure 4: A color and grayscale version of an image illustrating the advantages of color. 9

Figure 5: White light spread into a continuous spectrum by a prism. 10

Figure 6: Wavelengths making up the visible range of the electromagnetic spectrum 11

Figure 7: Absorption of light by the cones in the human eye as a function of wavelength..... 12

Figure 8: Color cube for normalized RGB coordinates. 13

Figure 9: Color cube for normalized CMY coordinates. 14

Figure 10: Color hexacone for HSV representation. 15

Figure 11: RGB color histograms of similar and dissimilar images. 19

Figure 12: Histograms whose perceptual similarity does not match calculated measures. 21

Figure 13: Gas tank image sliced into five overlapping fuzzy regions [10]. 24

Figure 14: Three different textures with the same intensity distribution. 29

Figure 15: Jigsaw Image Mosaic generated by the system described in [21]. 36

Figure 16: The MosaiX system architecture and data flow. 39

Figure 17: High-level flow diagram of the MosaiX preprocessor phase. 40

Figure 18: The resolution matching process on a single image. 41

Figure 19: Image sub-regions used during feature extraction. 42

Figure 20: Images with identical global features, but different visual content. 43

Figure 21: High-level flow diagram of the MosaiX image mosaic generation phase. 46

Figure 22: The 4x4 tessellation of an image into tesserae. 48

Figure 23: Difference in assembly from single match (A) and from a set of matches (B)..... 51

Figure 24: Mosaic quality evaluation low-frequency elimination preprocessing steps..... 54

Figure 25: Visual example of the edge extraction process on a mosaic and its target. 56

Figure 26: The fifteen test targets used in the MosaiX analysis. 58

Figure 27: Mean RMS error comparison by color model and image library size. 60

Figure 28: Mean RMS error comparison by color model and number of clusters. 60

Figure 29: Mean edge accuracy comparison by color model and image library size..... 61

Figure 30: Mean edge accuracy comparison by color model and number of clusters..... 62

Figure 31: Mean RMS error by number of clusters. 64

Figure 32: Mean RMS error by number of clusters for each color space. 65

Figure 33: Mean RMS error by number of clusters for each image library size. 65

Figure 34: Mean edge score by number of clusters. 66

Figure 35: Mean edge score by number of clusters for each color space. 67

Figure 36: Mean edge score by number of clusters for each image library size. 67

Figure 37: Mean execution time by number of clusters. 69

Figure 38: Mean execution time by number of clusters for each color space. 69

Figure 39: Mean execution time by number of clusters for each image library size..... 70

Figure 40: Mean RMS error by size of image library..... 71

Figure 41: Mean RMS error by size of image library for each color space..... 72

Figure 42: Mean RMS error by size of image library for each number of clusters. 72

Figure 43: Mean edge accuracy by size of image library. 73

Figure 44: Mean edge accuracy by size of image library for each color space. 74

Figure 45: Mean edge accuracy by size of image library for number of clusters. 74
Figure 46: Mean execution time by image library size. 75
Figure 47: Mean execution time by image library size for each color space. 76
Figure 48: Mean execution time by image library size for each number of clusters. 76
Figure 49: Reference target image for Appendix mosaics 81
Figure 50: Image mosaic - library size 5000, color model HSV, 2 clusters 81
Figure 51: Image mosaic - library size 5000, color model HSV, 5 clusters. 82
Figure 52: Image mosaic - library size 5000, color model HSV, 10 clusters. 82
Figure 53: Image mosaic - library size 5000, color model HSV, 20 clusters. 83
Figure 54: Image mosaic - library size 5000, color model L*a*b*, 2 clusters. 83
Figure 55: Image mosaic - library size 5000, color model L*a*b*, 5 clusters. 84
Figure 56: Image mosaic - library size 5000, color model L*a*b*, 10 clusters. 84
Figure 57: Image mosaic - library size 5000, color model L*a*b*, 20 clusters. 85
Figure 58: Image mosaic - library size 5000, color model RGB, 2 clusters. 85
Figure 59: Image mosaic - library size 5000, color model RGB, 5 clusters. 86
Figure 60: Image mosaic - library size 5000, color model RGB, 10 clusters. 86
Figure 61: Image mosaic - library size 5000, color model RGB, 20 clusters. 87
Figure 62: Image mosaic - library size 5000, color model YIQ, 2 clusters. 87
Figure 63: Image mosaic - library size 5000, color model YIQ, 5 clusters. 88
Figure 64: Image mosaic - library size 5000, color model YIQ, 10 clusters. 88
Figure 65: Image mosaic - library size 5000, color model YIQ, 20 clusters. 89
Figure 66: Image mosaic - library size 10000, color model HSV, 2 clusters. 89
Figure 67: Image mosaic - library size 10000, color model HSV, 5 clusters. 90

Figure 68: Image mosaic - library size 10000, color model HSV, 10 clusters. 90
Figure 69: Image mosaic - library size 10000, color model HSV, 20 clusters. 91
Figure 70: Image mosaic - library size 10000, color model L*a*b*, 2 clusters. 91
Figure 71: Image mosaic - library size 10000, color model L*a*b*, 5 clusters. 92
Figure 72: Image mosaic - library size 10000, color model L*a*b*, 10 clusters. 92
Figure 73: Image mosaic - library size 10000, color model L*a*b*, 20 clusters. 93
Figure 74: Image mosaic - library size 10000, color model RGB, 2 clusters. 93
Figure 75: Image mosaic - library size 10000, color model RGB, 5 clusters. 94
Figure 76: Image mosaic - library size 10000, color model RGB, 10 clusters. 94
Figure 77: Image mosaic - library size 10000, color model RGB, 20 clusters. 95
Figure 78: Image mosaic - library size 10000, color model YIQ, 2 clusters. 95
Figure 79: Image mosaic - library size 10000, color model YIQ, 5 clusters. 96
Figure 80: Image mosaic - library size 10000, color model YIQ, 10 clusters. 96
Figure 81: Image mosaic - library size 10000, color model YIQ, 20 clusters. 97
Figure 82: Image mosaic - library size 15000, color model HSV, 2 clusters. 97
Figure 83: Image mosaic - library size 15000, color model HSV, 5 clusters. 98
Figure 84: Image mosaic - library size 15000, color model HSV, 10 clusters. 98
Figure 85: Image mosaic - library size 15000, color model HSV, 20 clusters. 99
Figure 86: Image mosaic - library size 15000, color model L*a*b*, 2 clusters. 99
Figure 87: Image mosaic - library size 15000, color model L*a*b*, 5 clusters. 100
Figure 88: Image mosaic - library size 15000, color model L*a*b*, 10 clusters. 100
Figure 89: Image mosaic - library size 15000, color model L*a*b*, 20 clusters. 101
Figure 90: Image mosaic - library size 15000, color model RGB, 2 clusters. 101

Figure 91: Image mosaic - library size 15000, color model RGB, 5 clusters. 102

Figure 92: Image mosaic - library size 15000, color model RGB, 10 clusters. 102

Figure 93: Image mosaic - library size 15000, color model RGB, 20 clusters. 103

Figure 94: Image mosaic - library size 15000, color model YIQ, 2 clusters. 103

Figure 95: Image mosaic - library size 15000, color model YIQ, 5 clusters. 104

Figure 96: Image mosaic - library size 15000, color model YIQ, 10 clusters. 104

Figure 97: Image mosaic - library size 15000, color model YIQ, 20 clusters. 105

LIST OF TABLES

Table 1: Feature weight values used in the MosaiX distance metric..... 50

Table 2: Relative impact of preprocessing parameters on measured quantities. 77

CHAPTER ONE: INTRODUCTION

In today's technologically ambitious society it would be nearly impossible to find an artistic outlet that has not yet been affected by the extraordinary advances in computer technology. For example, the use of computer generated imagery is now almost standard in big budget blockbuster movies as the effects have become increasingly realistic. At the same time professional and amateur photographers alike are leaving their antiquated film and darkrooms behind in favor of compact flash cards and Adobe Photoshop. This huge and rapid shift in the photographic community, at all levels of skill, has caused the Eastman Kodak Company to severely cut its production of film and film-based cameras. In addition, Kodak will soon end its century old production of black and white photographic paper, clearly reflecting the industry-wide shift from analog to digital technologies. Computer science has even spawned new kinds of art independent of the artistic community, such as ASCII art in which images are formed using nothing but printable ASCII characters. Certainly there exists a link between the computing sciences and the arts. One remarkable example of what can happen when art fuses with computer science is the photographic image mosaic.

Mosaic Image History

A mosaic is exceptionally simple in concept; it is created by combining variously colored smaller pieces, called tessera, into a larger more significant overall shape. In ancient times, these works were often made from colored ceramic or stone tiles, decorating floors and walls and creating stunning figural compositions of everything from great battles to mythological scenes. Over time, the concept evolved into more complicated forms, such as the portraits created by 16th

Century painter Giuseppe Arcimboldo. His work was fashioned by painting ordinary objects such as fruits, vegetables, flowers, fish, and even books in such a way that the whole assortment formed an identifiable likeness to the portrait subject. Soon the idea of using other images as tessera emerged leading the way to works such as the 1976 Salvador Dali portrait of Abraham Lincoln. This work was composed by combining images with their own unique visual content, including a nude female figure, into a realizable portrait of the former U.S. President.

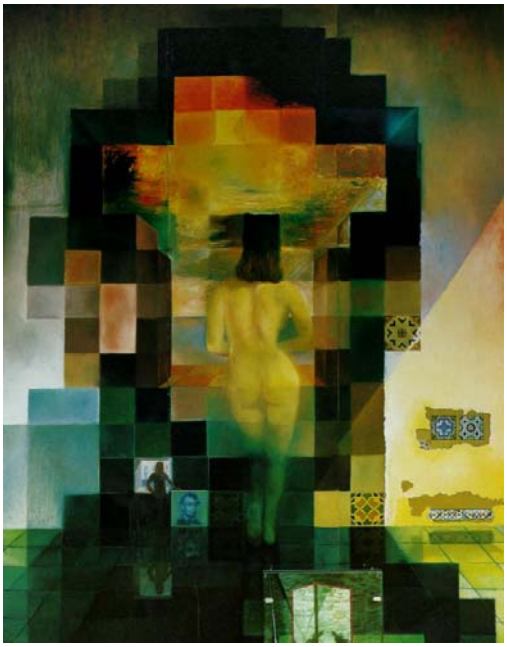

Figure 1: Mosaic Image of Abraham Lincoln [Dali].

In the mid 1990s, this idea of using other images as the tessera for a larger mosaic image sparked an idea in mind of MIT graduate student Robert Silvers. He devised an automated computer system that would take any image as input, and produce as output an impressively accurate visual representation of that original input built entirely from smaller equally sized photographs [1]. These computer generated image mosaics, or Photomosaics as Silvers called them, soon found their way into homes in the form of posters, jigsaw puzzles and various other

forms of merchandise. Despite Silvers holding patents on his process in several countries, dozens of software packages soon began to surface that would allow even the most novice users to create their own image mosaics from their digital photo collections.

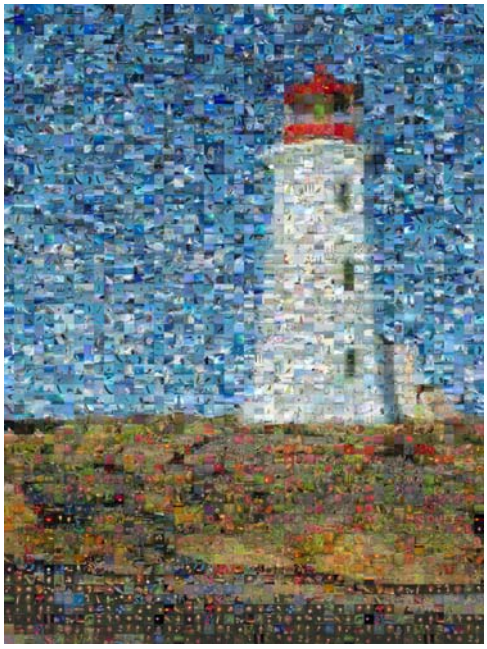

Figure 2: Lighthouse image mosaic created by Robert Silvers' system, Photomosaic [1].

Digital Image Mosaics and Image Retrieval

Regardless of the specific implementation, the software systems used for generating image mosaics will generally follow the same three basic steps. First the input image is subdivided into a number of smaller images referred to as tiles or tesserae. Next, each tessera is replaced by another image of similar visual content. Finally, the replacement images are spatially reassembled into a final image and output to the user in some form. The final montage of smaller images will come together to closely resemble the original subdivided image when viewed from a distance. The key procedure, and what inevitably sets one image mosaic system

apart from any other in terms of both quality and speed, lies in that second step: finding images with similar visual content to a given source tessera. In the computer vision world this process is known as Content-based Image Retrieval (CBIR) and has been an active topic of academic research for several years.

Many methods have been devised for the task of CBIR, but nearly all follow the same basic strategy. The first stage is to preprocess all database images and pull from each of them small pieces of information representative of their specific content. These pieces, called features, can be extracted in a multitude of ways depending on the specific goals of the image retrieval system. Once preprocessing is complete, queries can now be performed by providing the system with an example set of desired features, and performing some distance measure between the query feature set and all the feature sets of the image database. Images with the smallest distance to the target set are returned to the user as potential matches.

Given the importance of CBIR to the generation of an image mosaic Chapter Two of this thesis will be devoted to exploring CBIR theory and related research in much greater detail.

MosaiX

The notion of an image mosaic presents an intriguing concept for both artists and scientists alike. Although some academic research has been centered on the topic of image mosaics, overall, it is few and far between. Most methodologies and implementations are closely guarded secrets of their creators, including the original system devised by Robert Silvers [1]. Additionally, the fact that the quality of an image mosaic is generally a subjective measure, based on human observation, could be a contributing factor to its lack of professional study.

In this thesis, a new strategy for generating image mosaics is proposed that builds upon proven ideas and methods of CBIR found in recent academic research. Our system, MosaiX, first takes a target input image and performs a regular rectangular tessellation to create a collection of smaller images, or tesserae. Each of these tesserae is then replaced by an image of similar visual content from a large collection of digital images. In the final step these replacement images are reassembled to form a complete mosaic image whose overall appearance closely resembles that of the original target image. Since the size of the image library used for finding replacement images is quite large, a compressed representation of image color content, database clustering, and an efficient distance metric are used to reduce processing time. An example of an image mosaic created with MosaiX can be seen in Figure 3.

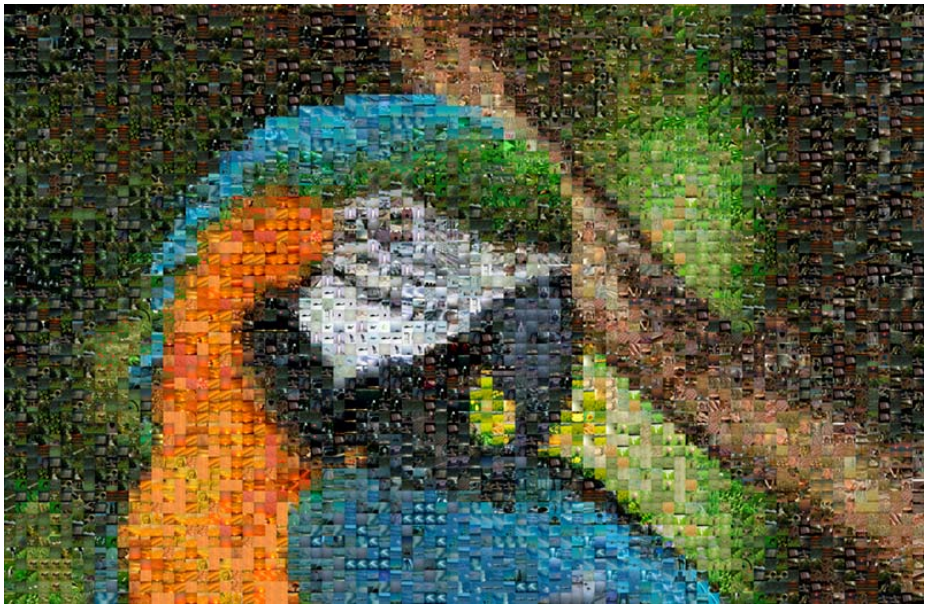

Figure 3: An example image mosaic generated with MosaiX.

Due to the aforementioned lack of published academic research in the area of image mosaic generation, direct comparison between MosaiX and other mosaic implementations will not be advantageous to this report. Instead, this thesis focuses on the methods employed by our

system, as well as an exploration into the affects of several system parameters. In addition, a new method of evaluating the overall quality of an image mosaic is also presented, and is used in the analysis on the affects of system parameters.

Thesis Organization

This thesis is composed of four more chapters. Chapter Two provides a detailed look at both the theory behind CBIR and the relevant research into its systems and methodologies. Chapter Three begins with a brief review of the image mosaic concept, followed by an analysis of current image mosaic techniques and technologies, before moving on to a highly detailed discussion of our proposed system, MosaiX. Chapter Four contains descriptions of the experiments performed as well as their respective results. Additionally, this chapter proposes methods by which an image mosaic's visual quality will be judged. The final chapter draws conclusions from the results obtained and proposes new ideas and directions this work could take in the near future.

CHAPTER TWO: CONTENT-BASED IMAGE RETRIEVAL

As previously mentioned in chapter one, Content-based Image Retrieval (CBIR) is the searching of an image database based on what is captured by the individual images of the collection. There are various ways of implementing these searches and they will be explored shortly. However, before moving on to the various strategies and methodologies, it helps to have a cursory understanding of the motivation behind the initial emergence of CBIR technology.

Motivation for CBIR

As a result of recent advancements in digital storage technology, it is now possible to create large and extensive databases of digital imagery. These collections may contain millions of images and terabytes of data. For users to make the most of these databases effective, efficient methods of searching must be devised. Prior to automated indexing methods, image databases were indexed according to keywords that were both decided upon and entered by a human categorizer. Unfortunately, this practice comes with two very severe shortcomings. First, as a database becomes increasingly large the manpower required to index each image becomes less practical. Secondly, two different people, or even the same person on two different days, may index similar images inconsistently. The result of these inefficiencies is a less than optimal search result for the end user of the system.

Having a computer do the indexing based on a CBIR scheme attempts to address the shortcomings of human-based indexing. Since a computer can process images at a much higher rate, while never tiring, the manpower issue is solved. Additionally, as long as the algorithms

used in the indexing procedure are kept constant, all images will be indexed consistently, solving the inherent problems resulting from fallible human-based indexing.

It should also be noted that current CBIR methodologies are not without their limitations. For example, each CBIR system needs to be tuned for its particular use in order to give optimal results. A retrieval system designed for querying medical x-ray images will more than likely prove to be a poor system for retrieving satellite images of South American rain forests. In addition, presently employed algorithms cannot yet consistently extract abstract features of images, such as emotional response, that would be relatively easy for a human to observe. For example, it would be rather difficult for current methods to classify images of heart shaped objects as well as two figures holding hands under a general category of “love” or “affection”.

General CBIR Theory

Like the process of generating image mosaics, regardless of their specific implementation most CBIR systems follow the same overall architecture. The first step is to preprocess the image database, extract from each image a set of specific representative features, and store them for later use in retrieval. At search time, a set of desired features are presented to the system and compared with the stored features of the database’s images, generated in the preprocessing stage, via some predefined distance measure. The images with the smallest distances to the query image are then returned to the user in rank order.

There have been large amounts of research into what type of features should be extracted and what measures should be used to compare them. The most popular features used in CBIR systems generally fall within three major categories: color, texture, and object/shape. The

following subsections will look at each of these three major categories individually, providing the basis for their use and related CBIR methodologies.

Color-based CBIR

Comparing the color content of images is an obvious, and consequently popular, choice for performing image retrieval duties. Color acts as a robust descriptor that can often simplify the tasks of object identification and extraction from a given image [4]. For example, in Figure 4 it is much easier to locate image pixels of the flower from the rest of the image when using the color image as opposed to the grayscale version. Due to the very nature of color representation, the color data itself provides multiple measurements at any given pixel location in an image. Because of the inherent properties of color, the last two decades have produced a number of interesting methods by which color image retrieval can be performed. A selection of these methods will be discussed following a review of the fundamentals of color and its methods of representation.

Figure 4: A color and grayscale version of an image illustrating the advantages of color.

Color Representation

The exact procedure by which the human brain processes and interprets color vision is a complex physiopsychological phenomenon that is far from fully understood. However, based on experimental and theoretical results, the physical aspects of light itself can be quite clearly expressed. In the 17th century, Sir Isaac Newton observed that when pure white light is passed through a prism it spreads uniformly into a continuous spectrum of colors from violet to red. He also noticed that within this spectrum no color abruptly ends, but rather there is always a smooth transition from one color to the next. Figure 5 illustrates this phenomenon. Each color in the spectrum corresponds to a specific wavelength, which in turn corresponds to a unique color. Various mixtures of these wavelengths can combine to form other colors as well.

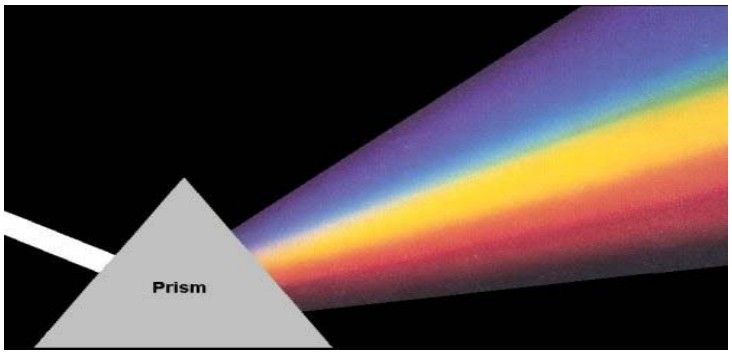

Figure 5: White light spread into a continuous spectrum by a prism.

The colors perceived by the human visual system are interpreted from the light that is reflected off objects in the immediate environment. As seen in Figure 6, visible light makes up only a small portion of the whole electromagnetic spectrum. Objects that reflect all wavelengths of the visible spectrum will appear as white, whereas objects that favor reflectance of certain wavelengths will appear as some shade of color to the observer. For example, objects that

readily reflect wavelengths in the 420 to 470 nm range while absorbing most others will appear as some shade of blue.

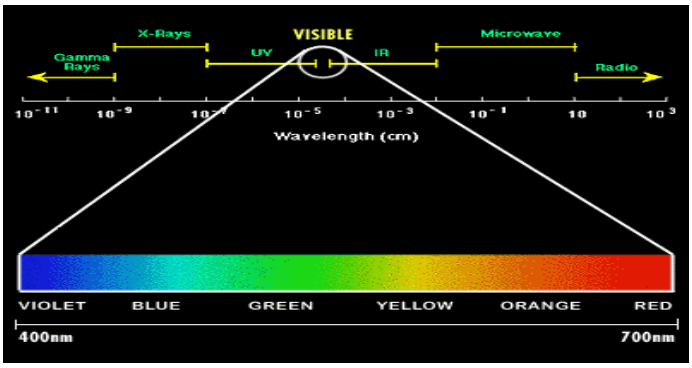

Figure 6: Wavelengths making up the visible range of the electromagnetic spectrum

Within the eye, special sensor cell structures, commonly referred to as cones, are responsible for the sensing of color in human vision. Based on experimental evidence, researchers have determined that the six to seven million of these cones that reside in the human eye can be subdivided into three distinct sensing categories that roughly correspond to the colors red, green, and blue. Of these cones approximately 65% belong to the red group, 33% to the green group, and just 2% to the blue group. It should be noted that these group proportions do not directly reflect color sensitivity; the blue cones are in fact the most sensitive, making up for their lack of presence. Figure 7 shows the absorption of light by the red, blue, and green cones as a function of wavelength in the eye, as well as noting the peak wavelengths to which they are most sensitive, on a normalized scale. Due to these properties of the eye's cone cells, colors are perceived as variable combinations of these primary colors. However, due to the continuous nature of the visible spectrum and the variable sensitivity of the cones, no single color can be labeled as red, green, or blue. Combinations of these primaries based on fixed wavelengths

cannot be used to produce all spectrum colors; in order to create all possible colors, the wavelengths of red, green, and blue would have to be permitted to vary.

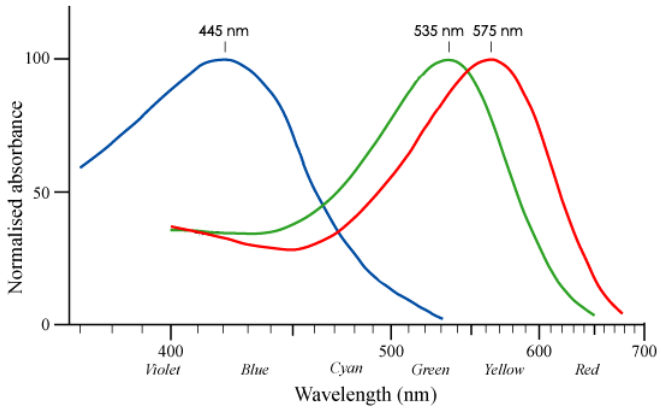

Figure 7: Absorption of light by the cones in the human eye as a function of wavelength

A color model is an abstract mathematical model that facilitates the specification of colors in some regular and accepted way [4]. The model, also known as a color space or color system, specifies a coordinate system and a subspace within that coordinate system where each point directly corresponds to a single color. Today, most color spaces are geared towards use in either computer hardware (monitors, printers, etc.) or towards applications where color manipulation is a primary objective (i.e. graphic design). There are currently numerous color spaces in practical use simply due to the fact that color science is an extensive field that covers many applications. A selection of models that have been found useful in the various areas of image processing will now be explored in greater detail.

The RGB Color Space

Within the RGB color space, each color appears as a three-dimensional point in a subspace of the standard Cartesian coordinate system. Each axis represents one of the three

color components (red, green, and blue) from which all colors in the system will be represented. These values are often normalized for convenience so that all values of red, green, and blue fall within the range [0, 1]. Figure 8 visually depicts the RGB subspace. Those images represented in the RGB color model consist of three component images, also called channels, one for each primary color. These components come together to represent the color at a given pixel in an additive way (each pixel is colored by a combination of red, green, and blue). This translates directly to the way a color monitor displays color, and for this reason it is often the default color space used in most applications. One notable drawback of the RGB color space is that it is not perceptually uniform, meaning that the calculated distance in RGB space does not truly correspond to the perceptual color difference [6].

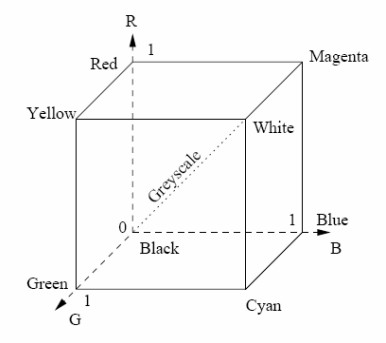

Figure 8: Color cube for normalized RGB coordinates.

The CMY & CMYK Color Spaces

Like the RGB color model, CMY color space is a subspace of standard three-dimensional Cartesian space, taking the shape of a unit cube. Each axis represents the basic secondary colors cyan, magenta, and yellow. Unlike RGB, however, CMY is a subtractive color model, meaning that where in RGB the origin represents pure black, the origin in CMY represents pure white. In

other words, increasing values of the CMY coordinates move towards darker colors where increasing values of the RGB coordinates move towards lighter colors (see Figure 9). Conversion from RGB to CMY can be done using the simple formula

$$\begin{bmatrix} C \\ M \\ Y \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} R \\ G \\ B \end{bmatrix}, \tag{1}$$

where it has been assumed that all color values have been normalized to the range [0, 1]. This equation reiterates the subtractive nature of the CMY model. Although equal parts of cyan, magenta, and yellow should produce black, it has been found that in printing applications this leads to muddy results. Thus in printing applications a fourth component of true black is added to create the CMYK color model. Four-color printing refers to using this CMYK model. As with the RGB model, point distances in the CMY space do not truly correspond to perceptual color differences.

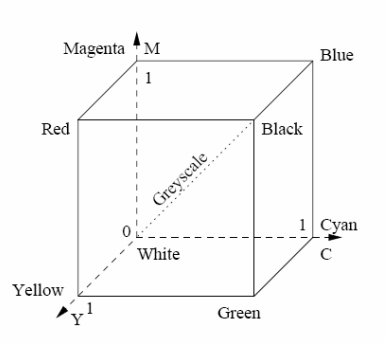

Figure 9: Color cube for normalized CMY coordinates.

The HSV & HSL Color Spaces

The HSV (hue, saturation, and value) and HSL (hue, saturation, lightness) color spaces are very different from the previously explored RGB and CMY/K color models in that both systems separate the overall intensity value of a point from its chromaticity [3]. The HSV space can be visualized in three dimensions as a downward pointing hexacone. The line running down the center of the cone's vertical axis represents the intensity value V. Hue is represented as the angle relative to the red axis, which resides on the plane perpendicular to the intensity axis. Saturation refers to a point's perpendicular distance from the intensity axis. Figure 10 illustrates this hexacone representation of the HSV color model.

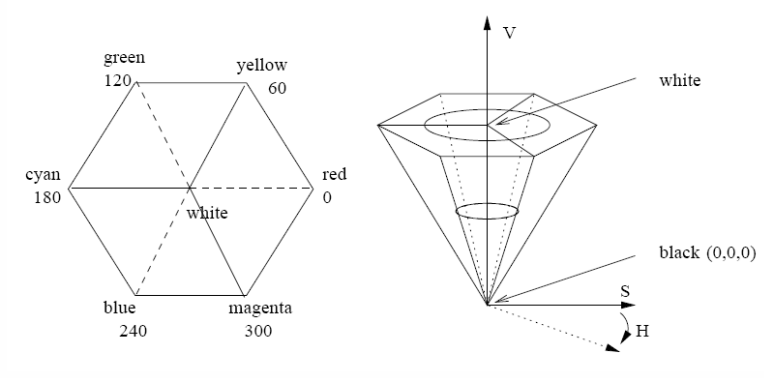

Figure 10: Color hexacone for HSV representation.

The HSL color model is very much similar to the HSV system. A double hexacone, with two apexes at both pure white and pure black rather than just one at pure black, is used to visualize the subspace in three-dimensions. In HSL, the saturation component always goes from a fully saturated color to the corresponding gray value; whereas in HSV, with V at its maximum, saturation goes from a fully saturated color to white, which may not be considered intuitive to some. Additionally, in HSL the intensity component always spans the entire range from black

through the chosen hue to white. In HSV, the intensity component only goes from black to the chosen hue. Because of the separation of chromaticity from intensity in both the HSV and HSL color spaces, it is possible to process images based on intensity only, leaving the original color information untouched. Because of this, HSV and HSL have found wide spread use in computer vision research.

The YIQ & YUV Color Spaces

Developed by and for the television industry, the YIQ color system arose from a need to compress broadcasted digital imagery with as little visual degradation as possible [3]. Like the HSV and HSL models, the luminance value Y is separated from the chromaticity values I and Q. This allows engineers to encode the luminance value with more bits than are used for the chromaticity values, since the human vision system is far more sensitive to changes in intensity. An approximate linear transformation from a given set of RGB coordinates to the YIQ space is given by the following formula:

$$\begin{bmatrix} Y \\ I \\ Q \end{bmatrix} = \begin{bmatrix} 0.30 & 0.59 & 0.11 \\ 0.60 & -0.28 & -0.32 \\ 0.21 & -0.52 & 0.31 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}. \tag{2}$$

A similar color model YUV is also used for television broadcasting as well as in the popular compression algorithms JPEG and MPEG. The approximate linear transform from RGB coordinates to the YUV space is given by the following formula:

$$\begin{bmatrix} Y \\ U \\ V \end{bmatrix} = \begin{bmatrix} 0.30 & 0.59 & 0.11 \\ -0.15 & -0.29 & -0.44 \\ 0.62 & -0.51 & 0.10 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}. \tag{3}$$

The CIE L*a*b* Color Space

The CIE L*a*b* color space was developed to be perceptually uniform and posses a Euclidean metric [5]. This means that Euclidean distance between two points (colors) would correlate strongly with human visual perception. CIE L*a*b* is based directly off of the CIE XYZ color model, where the X, Y, and Z components represent tristimulus capable of expressing any color that can be perceived by the average human observer [6]. These primary colors are nonreal, meaning that they are unable to be realized by actual color stimuli. It is not possible to directly transform RGB coordinates to CIE L*a*b* space due to the fact that RGB is not an absolute color space; it cannot produce all humanly discernable colors. Instead, RGB coordinates can be mapped into the CIE XYZ model through use of the following formula:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0.41 & 0.36 & 0.18 \\ 0.24 & 0.72 & 0.07 \\ 0.02 & 0.12 & 0.95 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}, \tag{4}$$

where R, G, and B are on the interval [0, 1]. From the XYZ space it is now possible to map into CIE L*a*b* space with the following equations [6]:

$$L^* = 116\left(\frac{Y}{Y_n}\right)^{\frac{1}{3}} - 16 \tag{5}$$

$$a^* = 500\left[\left(\frac{X}{X_n}\right)^{\frac{1}{3}} - \left(\frac{Y}{Y_n}\right)^{\frac{1}{3}}\right] \tag{6}$$

$$b^* = 200\left[\left(\frac{Y}{Y_n}\right)^{\frac{1}{3}} - \left(\frac{Z}{Z_n}\right)^{\frac{1}{3}}\right] \tag{7}$$

The values X_n, Y_n, and Z_n represent the tristimulus values of a reference white point.

Color-based CBIR Methodology and Research

With a firm understanding of how color is both physically perceived and represented in a mathematical way, it is possible to explore how color can be used to index and retrieve digital imagery from a large database. The basic standard behind color indexing an image database is, given a query image, to find and return all database images whose color composition and content are very similar to that of the query image. On the surface, this seems like a relatively easy task; however, the sheer amount of research into this topic alone proves that it is not without its difficulties. The following subsections will explore some of the most popular and effective methods of color-based image retrieval, addressing both their potential strengths as well as their associated weaknesses.

Color Histogram-based CBIR Methods

Color histograms have been extensively used in image retrieval systems since they were first introduced for color indexing in 1991 by Swain and Ballard [7]. A color histogram $H(I)$ of a given image I mapped into some discrete color space (RGB, HSV, YIQ, etc.) containing n colors is defined as the vector $(h_{c_1}, h_{c_2}, \ldots, h_{c_n})$ where each component h_{c_i} (also called a bin) corresponds to the total number of pixels of the color c_i within the image I. Additionally, it is common to scale each component of the bin (h_{c_i}) by the reciprocal of the total number of pixels in the image (a process known as histogram normalization). This histogram vector provides a respectable descriptor of what kind of color content is present in an image. It is therefore assumable that images with similar visual content should have roughly similar histograms, and

images with dissimilar content should have distinctly dissimilar histograms. Figure 11 visualizes this point by showing the color histograms of similar (examples 1 and 2) and dissimilar (examples 1 and 3) images.

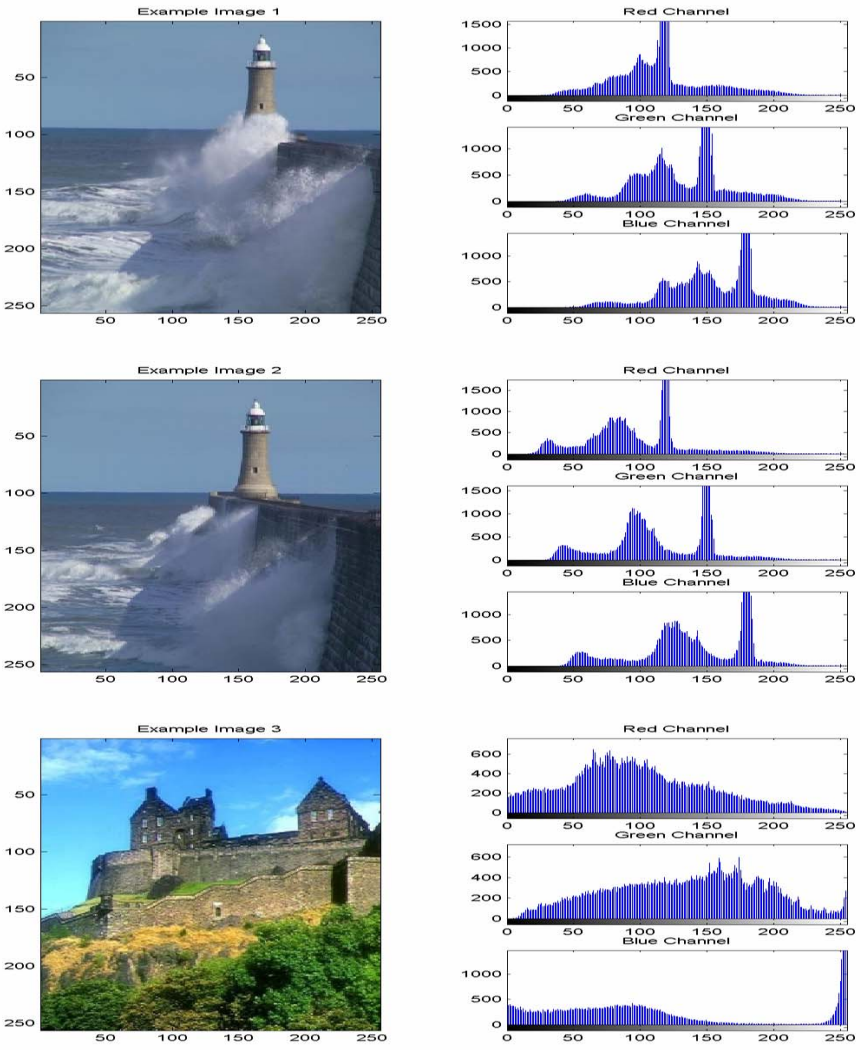

Figure 11: RGB color histograms of similar and dissimilar images.

In addition to being a measurement of the overall color content in an image, histograms have certain characteristics which make them well suited for image retrieval tasks. Many of these properties come from the fact that the information stored in the histogram vector has no

spatial relevance. In other words, the location of image pixels in no way affects the construction of the histogram. As a result, a relative invariance to image rotations and translations, caused by the same scene being imaged with differing viewpoints, is achieved by histogram indexing.

To retrieve images based on their color histograms, some similarity or distance measure must first be defined. In their work, Swain and Ballard proposed the use of what they called Histogram Intersection to gauge similarity. This measure was defined to be

$$Int(I,M)=\sum_{i=1}^{n}\min\left(h_{c_i}^{I},h_{c_i}^{M}\right), \tag{8}$$

where I and M are a pair of histograms of n bins each generated from the images we wish to compare. This gives a measure of how many pixels in one image have corresponding pixels of the same value in the other image. Another popular option for comparing the histograms of two images is the standard L_p-distance measure which is defined as

$$d_{L_p}(I,M)=\left(\sum_{i=1}^{n}\left|h_{c_i}^{I}-h_{c_i}^{M}\right|^{p}\right)^{\frac{1}{p}}. \tag{9}$$

Unfortunately, however, both of these measures are quite sensitive to even small changes in lighting, which leads to perceptually similar histograms being measured as quite dissimilar. As an example of this phenomenon, Figure 12 shows three hypothetical image histograms, each with the same number of bins (10) and the same total pixel count (3). Assuming the histogram bins have been structured such that nearby bins correspond to similar colors, then intuitively the difference between H_1 and H_2 should certainly be smaller than the difference between H_1 and H_3. However, the actual computed difference based on an L_1-distance metric are

$$d_{L_1}(H_1,H_2)=6>d_{L_1}(H_1,H_3)=4. \tag{10}$$

Given this result, the image described by H_3 would be retrieved, rather than H_2 had H_1 been used as a query image; this is an obviously less-than perfect outcome given the goal of CBIR. A similar outcome would result if we had used histogram intersection as well. This adverse effect is due to the fact that neither of the aforementioned comparison metrics takes into account the perceptual similarity between the various color bins.

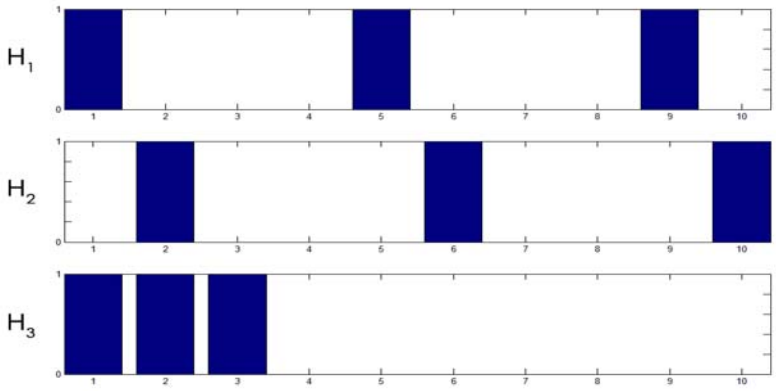

Figure 12: Histograms whose perceptual similarity does not match calculated measures.

In an effort to account for perceptual similarity of colors between the different histogram bins, IBM's QBIC System performs histogram comparisons using the following definition:

$$d_{hist}(I,M) = (H(I) - H(M))^T A (H(I) - H(M)), \tag{11}$$

where $H(I)$ and $H(M)$ represent the color histograms of images I and M respectively and the value A is a similarity matrix [8]. The elements a_{ij} of the similarity matrix are given by

$$a_{ij} = 1 - \frac{d_{ij}}{\max(d_{ij})}, \tag{12}$$

with the value d_{ij} being the L_2-distance between the colors i and j. Thus colors that are seemingly very similar would have values closer to one, while dissimilar colors would have values approaching zero. Such an approach does indeed improve results, but experiments by [9]

have shown that the improvement is not significant in reducing the number of dissimilar images that are returned as being similar.

Overall, while histograms do have several important characteristics that make them interesting choices for image indexing and retrieval, their shortcomings have lead to a search for other, more stable color-based CBIR methods.

Color Moment-based CBIR Methods

Based on a belief that the task of color indexing and retrieval could only be improved significantly if the index itself contained a more robust description of an image's color content, Stricker and Orengo proposed an entirely new index based on color distribution features [9]. This notion was based on probability theory, where a probability distribution can be effectively described by a number of features such as central moments, median, and mode. If the color distribution of an image (i.e. its normalized histogram) were to be interpreted as a probability distribution, then in turn one could describe image color content with the same range of features. Stricker and Orengo chose to use the first moment (average value) and the second and third central moments (standard deviation and skewness) from each color channel as their index. These moments for a given color channel i are calculated by the following formulas:

$$E_i = \frac{1}{N}\sum_{j=1}^{N} p_{ij}\,, \tag{13}$$

$$\sigma_i = \left(\frac{1}{N}\sum_{j=1}^{N}\left(p_{ij} - E_i\right)^2\right)^{\frac{1}{2}}, \tag{14}$$

$$s_i = \left(\frac{1}{N} \sum_{j=1}^{N} \left(p_{ij} - E_i \right)^3 \right)^{\frac{1}{3}}, \tag{15}$$

where p_{ij} is the j-th pixel of an image with N total pixels per channel. For image retrieval, these indexes are compared by a weighted L_1-distance measure, with user specified weight coefficients. If we let Q and M be the distributions of a pair of images with K color channels each, then the weighted distance between the two distributions is defined as

$$d_{L_1^w}(Q, M) = \sum_{i=1}^{K} w_{i1} \left| E_i^Q - E_i^M \right| + w_{i1} \left| \sigma_i^Q - \sigma_i^M \right| + w_{i1} \left| s_i^Q - s_i^M \right|. \tag{16}$$

The weight coefficients can be used to fine tune the distance measure for a specific application. For example, if we were working within the HSV color space, and held a match in overall color higher than a match in intensity, we could turn up the weights for the hue channel while turning down the corresponding value weights. Results reported in [9] showed a distinct advantage over the most popular histogram based methods. In addition, like histogram based indexing, the use of distribution features allows for an inherent invariance to rotations and translations due to changes in viewpoints (no spatial information is encoded with the index).

In an attempt to strengthen the discrimination power of the color distribution feature index, Stricker and Dimai proposed encoding a minimal amount of spatial information into the index. This was accomplished by tessellating images into five partially overlapping, fuzzy regions (see Figure 13), and then extracting the first three moments (as was done in [9]) from each region and storing them as the index [10]. The fuzzy aspect of the regions allowed pixels towards the center of the regions to have more impact on the moment calculations than those pixels towards the edges of the regions. A very similar weighted L_1-distance measure was used

to compare the new spatially encoded indexes, as was used in [9]. However, a modification was made to allow for invariance of rotations of ±90 degrees (i.e. changes from portrait to landscape orientations) by comparing the corner region R_1 of a query image with all corner regions of a given database image (R_1, R_2, R_3, and R_4) and returning the minimum distance of those four comparisons. This process then is repeated for R_2, R_3, and R_4 of the query image. The center region R_0 is of course strictly compared only to other images' center regions. Such a strategy can be loosely thought of as comparing the query image with 0°, 90°, 180°, and 270° rotated versions of the database images.

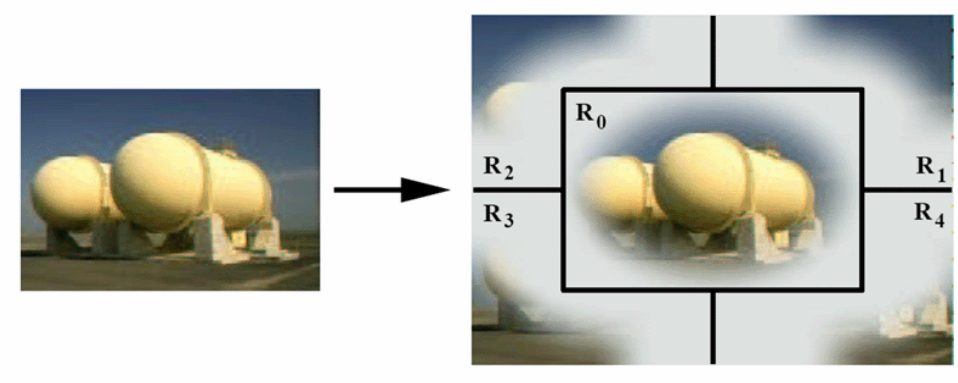

Figure 13: Gas tank image sliced into five overlapping fuzzy regions [10].

Mostafa, Abbas, and Wahdan also attempted to encode spatial information along with indexes built from color distribution features [11]. Their system used a color induced segmentation of images that first divided an image into an initial set of four equal sized rectangles and then, based on some criteria, decided if any subregion should be further divided into another set of rectangles. This would continue until either the size of the rectangles reached a specified minimum size or no regions required further subdivision. The criteria for deciding if a region required another division is reported as a color based homogeneity predicate. At each

stage of the segmentation, as well as from the original image, the first two color moments are extracted. These features are stored in a tree structure with the root representing the features of the original whole image. The next level of the tree would represent the features of the first pass through the segmentation, and so on until all features from the image are so expressed. The distances between images using this index are calculated in a multistage fashion. First the top level (root of the tree) features are compared and, if the distance is less than or equal to a predefined tolerance, then the comparison moves down to the next level of the tree. If the distance is found to be greater than the defined tolerance level, calculations stop and the image is discarded as a mismatch. This cycle continues until all levels of the query image have been compared. Images that are less than or equal to the tolerance at all segmentation levels are declared matches and returned to the user. This multistage methodology has the interesting advantage of allowing gross mismatches to be filtered out early in the query process.

Another line of attack based on the original ideas of [9] was proposed in a work by Shih and Chen [12]. Their method begins by first hard-partitioning an image into a given number of uniform, non-overlapping blocks. From each block the first H color moments are calculated and stored as a feature vector. After all blocks have been processed, the generated feature vectors are clustered into some unknown number of clusters. The number of clusters is determined by the clustering algorithm and may not be the same for all database images. Centroids of these clusters in feature space now represent the primitive features of the image and are stored as the final index. A given query image Q is compared to a database image S by looping over all primitives of Q and finding the best matching (smallest distance) primitive of S. These best matching distances are multiplied by the size of the cluster (the number of blocks

in the cluster from image Q) and summed together, producing a final combined distance. The reciprocal of this distance value is then used as the final measure of overall similarity.

Color Correlogram-based CBIR Methods

In addition to the use of color histograms and color distribution moments, several other ideas have been proposed that attempt to index and retrieve digital images based on their color content. The use of the color correlogram, introduced in 1997, is one such method [13]. Color correlograms (henceforth correlogram) of images take the form of a table indexed by color pairs where the k-th entry for $\langle i, j \rangle$ is used to specify the probability of finding a pixel of color j at a distance of k from a pixel of color i in a given image. More formally, the correlogram is defined by

$$\gamma^{(k)}(I) = \Pr_{p_1 \in I_{c_i}, p_2 \in I} \left[p_2 \in I_{c_j} \,||\, p_1 - p_2| = k \right]. \tag{17}$$

The value of $\gamma_{c_i, c_j}^{(k)}$ gives the probability of a pixel with color c_i being a distance of k away from a pixel with color c_i. The values of p_1 and p_2 are pixels with locations (x_1, y_1) and (x_2, y_2) respectively, and the quantity $|p_1 - p_2|$ is defined as

$$|p_1 - p_2| = \max(|x_1 - x_2|, |y_1 - y_2|). \tag{18}$$

To compare the similarity of correlograms the authors of [13] proposed the use of a modified L_1-distance measure. For a pair of images Q and I, the distance between their correlograms is defined by the following:

$$\left|Q-I\right|_{\gamma}=\sum_{i,j\in[m],k\in[d]}\frac{\left|\gamma_{c_i,c_j}^{(k)}(Q)-\gamma_{c_i,c_j}^{(k)}(I)\right|}{1+\gamma_{c_i,c_j}^{(k)}(Q)+\gamma_{c_i,c_j}^{(k)}(I)}. \tag{19}$$

The sets $[m]$ and $[d]$ represent all possible color values and interpixel distances respectively.

This formulation for indexing and retrieval is robust in its tolerance to changes in image appearance based on a change in viewing position. Additionally it also has the ability to describe both local and global spatial correlation of colors. Experiments from [13] show that this index can outperform traditional histogram-based approaches.

Retrieval based on correlograms was later improved upon through the use of supervised learning [14]. For the sake of simplicity, this work redefined the distance measure to a weighted L_1-distance measure defined by

$$\left|Q-I\right|_{\gamma}=\sum_{i,j\in[m],k\in[d]}w_{ij}\cdot\left|\gamma_{c_i,c_j}^{(k)}(Q)-\gamma_{c_i,c_j}^{(k)}(I)\right|. \tag{20}$$

The weights $w_{i,j}$ were trained using sets of correlograms corresponding to either images of similar or dissimilar visual content. Given a pair of correlograms, a difference vector δ is defined to be the component-wise difference between correlograms, and the mean value of this vector is represented by $\bar{\delta}$. If the difference vector was generated using a pair of correlograms from images deemed similar, the weight vector components are updated with the following heuristic update rule:

$$w_{ij}=w_{ij}\cdot\left(1+\bar{\delta}-\delta_{ij}\right). \tag{21}$$

For correlograms generated from images that are considered to be dissimilar, the update rule is only slightly changed to:

$$w_{ij}=w_{ij}\cdot\left(1+\bar{\delta}+\delta_{ij}\right). \tag{22}$$

The effect of these updates is that dimensions showing larger differences for examples of dissimilar image correlograms are increased in influence, while dimensions showing larger differences for examples of similar image correlograms are decreased in influence. That is, the weight vector identifies what dimensions are more important based on examples provided. Results from [14] showed distinct improvements in the retrieval effectiveness of correlograms when combined with the learned dimension weights.

Texture-based CBIR

Another popular approach to CBIR involves the use of texture in order to index database images. Texture in the realm of image processing gives information about the local spatial arrangement of colors or intensities in a given image [3]. Images that have similar texture properties should therefore have the same spatial arrangements of colors or intensities, but not necessarily the same colors. Because of this, the use texture-based image indexing and retrieval techniques is quite different than those used strictly for color. Before exploring texture-based indexing and retrieval methods, the concepts of texture and texture representation should first be reviewed in greater detail.

Texture Representation

As previously mentioned, texture is used to capture the spatial arrangement of pixel values rather than the values themselves. Figure 14 illustrates this point. Here three image blocks are shown with three distinct textures (block, checkerboard, and striped); however each would produce the exact same histogram. Although these are relatively simple simulated

textures, they clarify the difference between color representation and texture representation at a perceptual level.

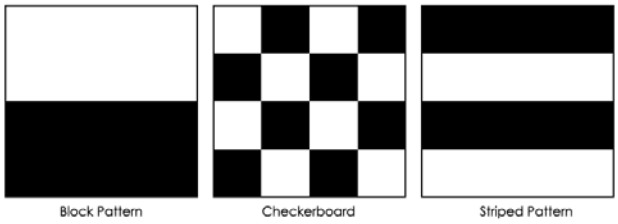

Figure 14: Three different textures with the same intensity distribution.

There are two main approaches to the problem of defining exactly what texture is: the structural approach and the statistical approach. In the structural approach, textures are represented by a group of primitive texels in some regular or recurring relationship. The statistical approach relies on a quantitative measure of the arrangement of intensities in a given region. Although attractive, the structural approach proves to work well only on very regular, (i.e. man-made) patterns whereas the statistical approach provides a more general line of attack that is more often used in real-world applications. Because of this property, the scope our discussion will be limited to a review of quantitative texture measurements.

Edge-Based Texture Measures

As edge detection methods are quite well-known and rather simple to apply, edge density and direction have been used as quantitative methods of measuring texture in many applications. The number of edge pixels in some preset region can give insight into the overall busyness of that region [3]. In addition, the directions of the edges are useful in texture classification and are often readily available as an offshoot of many edge-detection processes. Consider an image sub-

region of composed of N pixels, on which a gradient-based edge detector has been applied (such as the Sobel). Two distinct outputs for each pixel p will be observed: the gradient magnitude (denoted $Mag(p)$) and the gradient direction (denoted $Dir(p)$). Using only $Mag(p)$, a measure of the edge density for some threshold T can be defined as

$$F_{edgeness} = \frac{|p|Mag(p) \geq T|}{N}. \tag{23}$$

If the inclusion of orientation was desired, then histograms could be utilized for both $Mag(p)$ and $Dir(p)$. Given a region of pixels R on which a gradient edge-detector has been applied, the histograms of gradient magnitudes and gradient directions are denoted by $H_{mag}(R)$ and $H_{dir}(R)$ respectively. These histograms are normalized according to the total number of pixels in the region R, and contain a relatively small number of bins (often less than 10). This combined description of edge magnitudes and directions provides a distinct quantitative measure of the texture in region R.

Local Binary Partitions

Local binary partitions provide a very simple way of expressing the texture in an image [3]. To generate this descriptor, the eight-connected neighbors of every image pixel are analyzed to see if their intensity values are greater or less than that of the current pixel. From this analysis, an eight-digit binary number is produced for each pixel in the form of $b_1b_2b_3b_4b_5b_6b_7b_8$. Each digit b_i is equal to 1 if the intensity of the i-th neighbor is greater than the given pixel; otherwise, it is set to 0. The texture of the entire image is then described by a histogram of these binary numbers.

Laws Texture Energy Measures

The generation of texture features can also be accomplished through the use of local convolution masks. In 1980, Laws proposed a unique texture-energy strategy that gauges the quantity of pixel variation within a small image window [2]. This work involved the use of distinct 5×5 convolution kernels that are generated from the following four basis vectors:

$$L5=\begin{bmatrix}1 & 4 & 6 & 4 & 1\end{bmatrix} \tag{24}$$

$$E5=\begin{bmatrix}-1 & -2 & 0 & 2 & 1\end{bmatrix} \tag{25}$$

$$S5=\begin{bmatrix}-1 & 0 & 2 & 0 & -1\end{bmatrix} \tag{26}$$

$$R5=\begin{bmatrix}1 & -4 & 6 & -4 & 1\end{bmatrix} \tag{27}$$

Each of these basis vectors corresponds to some texture feature such as ripple detection ($R5$) and spot detection ($S5$). The 5×5 convolution kernels are created via matrix multiplication of each basis vector pair. For example, the convolution kernel for $E5L5$ is computed as

$$\begin{bmatrix}-1\\-2\\0\\2\\1\end{bmatrix}\times\begin{bmatrix}1 & 4 & 6 & 4 & 1\end{bmatrix}=\begin{bmatrix}-1 & -4 & -6 & -4 & -1\\-2 & -8 & -12 & -8 & -2\\0 & 0 & 0 & 0 & 0\\2 & 8 & 12 & 8 & 2\\1 & 4 & 6 & 4 & 1\end{bmatrix}. \tag{28}$$

Before any convolution takes place, however, some preprocessing is done to an image in order to remove some effects of illumination. This is accomplished by sliding a 15×15 window around the image and subtracting the local average from each pixel, producing an image where the average intensity of each window is near zero. The preprocessed image is then convolved with each of the sixteen possible kernels, derived from the four basis vectors, resulting in the

production of sixteen filtered images. Filtered images are then transformed to energy maps, E_k, by the following formula:

$$E_k[r,c] = \sum_{j=c-y}^{c+7} \sum_{i=r-7}^{r+7} |F_k[i,j]|. \quad (29)$$

Here $F_k[i,j]$ represents the result of filtering the preprocessed image with the k-th mask at pixel $[i,j]$. The coordinate pair $[r,c]$ represents the target pixel in the energy map currently being calculated. Of these sixteen energy maps, certain ones representing symmetric pairs are mathematically combined to produce a final nine energy maps. The final result gives each pixel in the original images a nine-component feature vector on which texture processing can be performed. For example, these feature vectors can then be used to cluster image regions of relatively uniform texture, creating an image segmented by texture.

Texture-based CBIR Methodology and Research

Performing image retrieval based on texture features in many ways resembles the basic methods of color-based CBIR. For example, if the local binary partition method detailed earlier was used to generate a texture histogram as a database index, then any of the histogram distance measure formulas described for color-based CBIR could also be applied to find image distances based on texture as well. Another method borrowing from the ideas of color-based CBIR is the gridded texture distance [3]. In this method, a fixed rectangular grid is placed over a given query image as well as all database images, allowing for spatial information to be captured in the distance calculation. From each grid region an image description vector is calculated, such as

the type generated with Laws' method. The gridded texture distance between images I and Q is then defined by

$$d_{gridded\_texture}(I,Q)=\sum_{g}\hat{d}_{texture}\left(T^{I}(g),T^{Q}(g)\right), \tag{30}$$

where $\hat{d}_{texture}$ is some predefined distance measure, such as an L_p-distance metric, and $T^{k}(g)$ represents the texture description vector of image k in grid region g.

More elaborate texture retrieval methods have also been experimented with in the last several decades. Many of these methods work by first transforming images into alternate representations though use of the Fourier or wavelet transforms. In [15] texture-based feature vectors were generated by computing the energy and standard deviation of wavelet subbands of images decomposed with rotated, complex wavelet filters. These feature vectors were then compared using the Canberra distance metric which is defined as

$$Canb(x,y)=\sum_{i=1}^{d}\frac{|x_i-y_i|}{|x_i|+|y_i|}, \tag{31}$$

where x and y are two d-dimensional feature vectors. Use of the Canberra distance was chosen over standard Euclidean distance due to its normalization properties. In [16], the authors attempted to create a system that would adaptively learn to choose the correct wavelet transform to query a texture database via supervised learning. Their theory was based on the idea that the human visual system uses different types of information for classifying different patterns. Results from their experiments showed overall improvements when compared to other systems that were based on only a single transform.

Stepping away from wavelet transforms, the authors of [17] took an approach similar to the color moment techniques discussed earlier. Their feature extraction operates on the original

image with eight templates derived from local Fourier transform (LFT) techniques. Each of these templates characterizes some information on a particular aspect of the original image. By applying the templates to the original image, an intermediate characterization map can be obtained. From each of the characterization maps, the first and second moments are extracted as features. Images are then compared and retrieved using these features. Experimental results showed the features do provide decent texture-based indexing.

Shape/Object-based CBIR

Unlike color and texture, shapes and objects are not global attributes of an image. In the color and texture realm, distance measures are used to establish if a given image has a specified color or texture, and whether or not it exists in the same approximate position as the query image. Shape and object recognition techniques, on the other hand, require the use of more sophisticated segmentation and classification steps before any kind of similarity metric can be performed. In many cases this is often done manually, such as a user tracing the outline of a target shape or object in a given query image; however, the use of automatic image segmentation is becoming increasingly common. Overall, shape and object CBIR methodologies are generally targeted systems, often only looking for a very narrow range of information in a collection of images.

The topic of flesh finding, first pioneered by Forsyth and Fleck, provides an excellent example of a shape/object-based CBIR implementation [18]. Their approach, aimed at identifying potentially pornographic imagery, attempts to automatically find and segment regions within images that have the color and texture of human skin. Once the skin pixels have been

identified, they attempt to classify whether the distribution of potential skin pixels within the image corresponds with a naked human form. This is done by first checking that there is enough flesh visible (30% of the image was the threshold in [18]), and then determining whether the regions were in appropriate spatial relations to be considered human body parts. The reported results showed the system could correctly identify objectionable material about fifty percent of the time. While this not as accurate as other state-of-the-art object recognition algorithms, the results were still impressive considering the relative simplicity of the implementation (no predefined, geometric models are used for object/shape matching).

There are, of course, many other methods of shape/object-based CBIR that have been explored in the last several years. However, these types of strategies have little use in the realm of generating artistic image mosaics and, as a result, their methods will not be discussed further. Interested readers are referred to [3], [19], and [20] for more details on shape/object-based CBIR systems and methodologies.

CHAPTER THREE: MOSAIX

Generating an image mosaic, regardless of specific implementation details, will always require three explicit steps. First, an input image is tessellated into a series of smaller images, referred to as tesserae. Next, each tessera is replaced with another image of similar visual content from some large collection of possible images. Finally, these tesserae are reassembled into a completed montage whose overall visual content closely resembles that of the original image. Systems that generate image mosaics are set apart by how they implement each of these three key production stages. For example, some systems have unique ways in which they tessellate and reassemble their image mosaics. The system presented in [21] generates images that have arbitrarily shaped tesserae to create what they refer to as a Jigsaw Image Mosaic (JIM). An example of this type of mosaic is seen in Figure 15. More often, however, the defining characteristic of a system that generates image mosaics is the way in which replacement images are selected from a collection of potential candidates. More specifically, what kind of CBIR scheme is employed within.

Figure 15: Jigsaw Image Mosaic generated by the system described in [21].

Although the method of tessera image selection within an image mosaic system is often a closely guarded secret of its creators, details of a few systems are known. For example, the patent for Robert Silvers' original photomosaic system strongly implies that replacement images are selected based on a pixel-by-pixel RMS error [1]:

$$E_{RMS}(A,B) = \sqrt{\sum_{k=1}^{K}\sum_{n=1}^{N}\left(p_{kn}^{A} - p_{kn}^{B}\right)^2}\ . \tag{32}$$

In this equation, $E_{RMS}(A,B)$ defines the RMS error between two images, A and B, each with K channels and N pixels per channel. The values p_{kn}^{A} and p_{kn}^{B} represent the intensity values at pixel n in channel k for images A and B respectively. Thus, for a given target tessera, the database image with the smallest RMS error is used in the final image mosaic. This kind of selection method would certainly provide good results; however, the dimensionality of this measure would be extremely high for even moderately sized tessera images. For example, a 64x64 pixel, 3-channel image would have a total of 12,288 dimensions. Such high-dimensionality, combined with a sizable collection of images with which to compare, leads to excessive computation times.

The imaging software company ArcSoft attempted to improve upon the selection technique in their commercial product PhotoMontage. Their method, described in [22], attempts to first identify only those database images with the highest probability of being a good match to the current target tessera. These images are identified by comparing the overall average value of the target tessera with that of the database images. The final selection is then made using the RMS error method only on those images with similar overall average value. This method allows for very dissimilar images to be quickly eliminated prior to performing the computationally costly, high-dimensional RMS error calculation.

Unlike Silvers' Photomosaic and ArcSoft's PhotoMontage, the image mosaic generation system proposed in this thesis, MosaiX, attempts eliminate the use of high-dimension RMS-error calculations in replacement image selection by utilizing lower dimensional indexing and retrieval techniques explored in recent academic CBIR research.

The MosaiX System

Our proposed image mosaic generation methodology, MosaiX, is broken down into two high-level phases: the preprocessing phase and the image mosaic generation phase. During preprocessing, a user supplied image library undergoes several distinct operations that will accurately index the individual library images based on their color content. Within the image mosaic generation phase, a target image is recomposed with images from the indexed image library using an efficient content-based image retrieval method. Figure 16 visually illustrates the MosaiX system architecture from a high-level, flow-based view. Detailed explanations of the operations performed within each phase will be explored in the next several subsections.

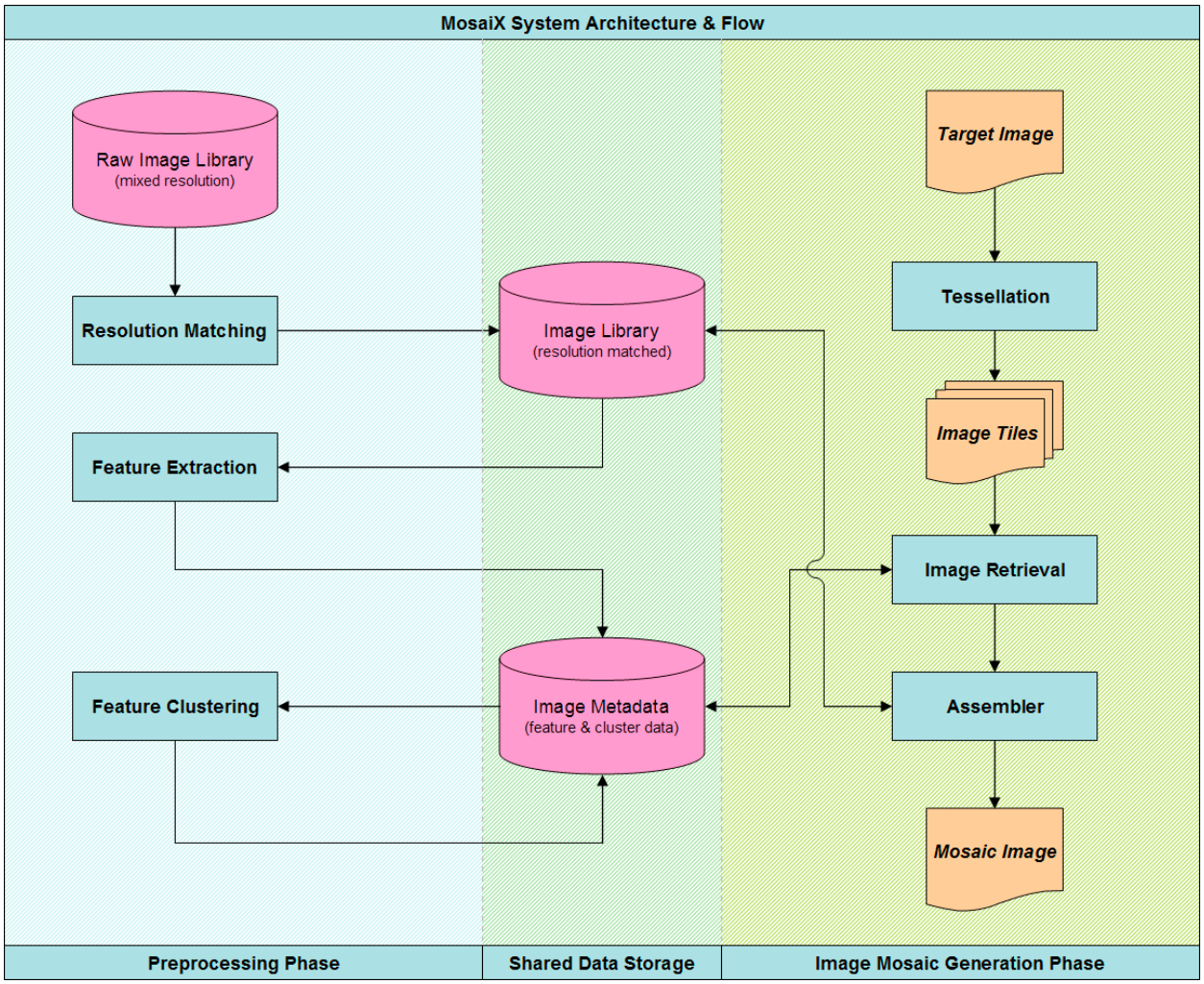

Figure 16: The MosaiX system architecture and data flow.

Preprocessing Phase

Within the MosaiX system, the preprocessing phase is responsible for the tasks of resolution matching, feature extraction, and feature clustering (as seen in Figure 17). These three subtasks are what allow MosaiX to represent a large collection of high-dimensional images in an extremely compact, low-dimensional, manner. This generation of compact representation is

known as indexing and, as we know from Chapter Two, is the first half of the CBIR equation. Each of the preprocessing subtasks will be individually detailed in the next subsections.

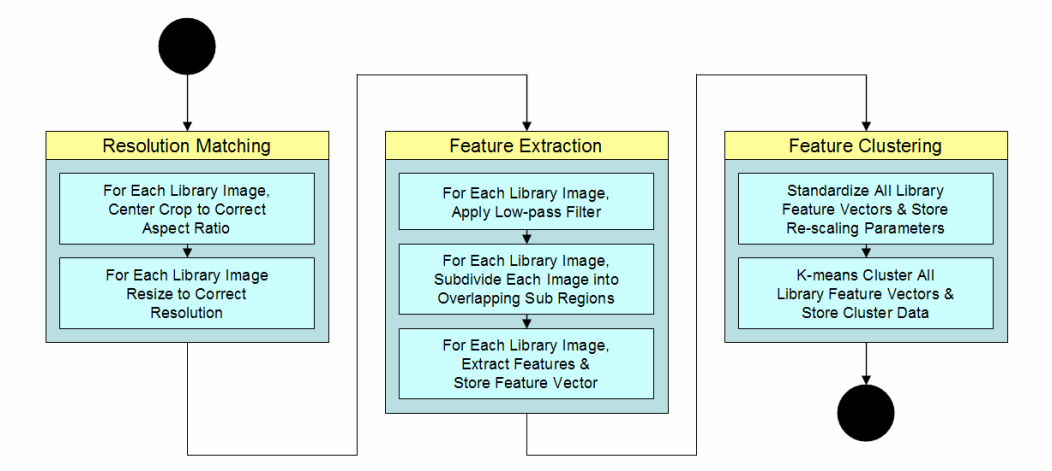

Figure 17: High-level flow diagram of the MosaiX preprocessor phase.

Resolution Matching

The MosaiX preprocessor begins its work when a user presents the system with a large collection of mixed resolution imagery. In order for later feature extraction to be consistent, every image must be of the same pixel dimensions. The resolution matching process performs this operation by resizing each image of the user supplied library to a set of user specified dimensions. Resizing of an image is done in two steps to ensure superior results. First, the image is center-cropped to the correct aspect ratio as determined from the user specified dimensions. Center-cropping is preferred over cropping from one of the image corners due to the natural tendency for most photographers to center a photo's subject within the image frame. Secondly, the now cropped image is then resampled to the final image resolution using basic bilinear interpolation. This process is performed on each and every image of the user supplied

image library with the resulting images stored in a centralized location. The resolution matching process, as performed on a single image, is illustrated in Figure 18. With all images resolution-matched and safely stored away in a separate library database, the feature extraction process can now begin.

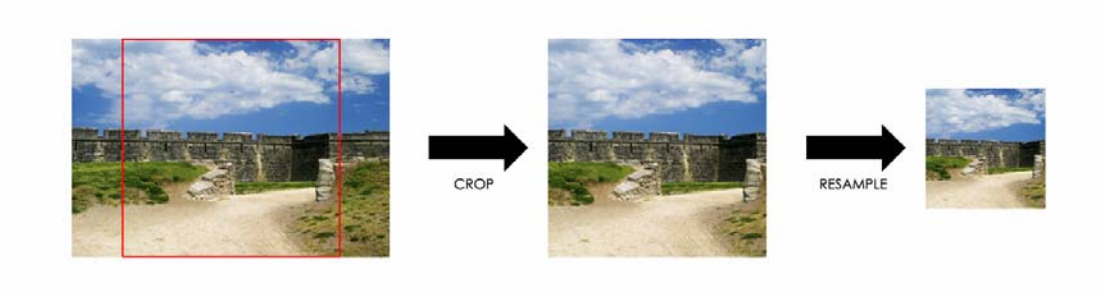

Figure 18: The resolution matching process on a single image.

Feature Extraction

During the feature extraction operation, a specific set of color-based features are extracted from each resolution-matched library image and stored as an index. The use of color-based features was chosen over both texture and shape due to inherent properties of image mosaics. More specifically, the tessera extracted from a target image generally contain information at a distinctly different scale than the images that will be used to replace them. This difference in scale between query images and database images would most likely prove detrimental to texture-based or shape-based CBIR methods.

Color feature extraction on a given resolution-matched library image begins by first passing the image through a low-pass filter to smooth out high-frequency information. This filtering is done based on the assumption that the final image mosaic will be viewed from a distance, and thus the high-frequency content of the tessera images will have little effect on the

mosaic's overall perception by the human observer. Next, the image is subdivided into five overlapping sub-regions representing the north, south, east, west, and center regions. The north and south regions correspond to the top and bottom halves of the image, east and west to the right and left halves of the image, and the center corresponds to the central quarter of the image. These regions are depicted in Figure 19.

Figure 19: Image sub-regions used during feature extraction.

From each color channel, in each of the five sub-regions, the average value E, standard deviation σ, and skewness s are calculated creating a 45-dimension feature vector. This feature vector is then stored as an index (metadata) for the given image. These features are the same as those used in [9] and are evaluated using the following formulae:

$$E_{rk} = \frac{1}{N}\sum_{j=1}^{N} p_{rkj}\,, \tag{33}$$

$$\sigma_{rk} = \left(\frac{1}{N}\sum_{j=1}^{N}\left(p_{rkj} - E_{rk}\right)^2\right)^{\frac{1}{2}}, \tag{34}$$

$$s_{rk} = \left(\frac{1}{N}\sum_{j=1}^{N}\left(p_{rkj} - E_{rk}\right)^3\right)^{\frac{1}{3}}, \tag{35}$$

where p_{rkj} is the j-th pixel of channel k, in sub-region r, with N total pixels per channel per region. Choice of color model used during feature extraction is left as an input parameter to the

system. The effect of this selection on overall image mosaic quality is experimentally explored in Chapter Four.

The aforementioned subdivision of the library images allows for some spatial information to be encoded into the index along with color data, thus providing greater accuracy in the later retrieval stage. For example, consider the images shown in Figure 20. Despite having distinctly different visual content, globally, both images have the exact same average, standard deviation, and skewness values. By subdividing the image into distinct spatial regions, the difference in visual content becomes readily apparent and allows for a more accurate estimation of a given target image in the image mosaic generation phase.

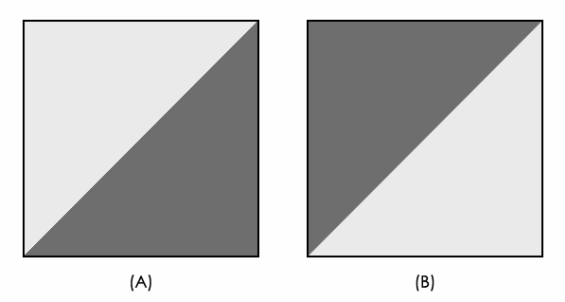

Figure 20: Images with identical global features, but different visual content.

The net result of the feature extraction process is a collection of feature vectors that provide a compressed representation of the color content for each library image. This set of feature vectors is collectively referred to as a *feature matrix,* where rows correspond to individual library images and columns correspond to each of the extracted features. This feature matrix is safely stored into a database for later use in feature vector clustering, as well as image retrieval during the image mosaic generation phase.

Feature Vector Clustering

For any given query on a image database, it is generally understood that the vast majority of the images within will not be close matches. For this reason, it would be advantageous to develop a method that quickly identifies which images have the highest probability to match a given query. To do this, MosaiX includes an additional preprocessing step that clusters the image-extracted feature vectors into groups of relatively similar content. Our implementation utilizes basic k-means clustering [23] (with a Euclidean metric) on the matrix of feature vectors generated during the feature extraction stage. The number of clusters, k, is left as an input parameter to the system. The effect of the number of clusters on both processing time and image mosaic quality is detailed in Chapter Four.

To improve clustering results, each feature vector is re-scaled using a process known as standardization prior to performing the k-means algorithm. This re-scaling results in each feature vector component having a mean of zero and standard deviation of one across all library images. To do this, one needs to subtract from each component the mean value of that dimension across all images and then divide by the standard deviation. This standardization prevents feature vector components from dominating the clustering process simply due to a large overall magnitude that may be inherent in some feature. These standardized feature vectors are now run through the k-means algorithm, producing a cluster label for each feature vector (and thus each image) as well as the k centroids of each cluster. The cluster labels and cluster centroids are stored into the same database as the original feature vectors. Additionally, the mean and standard deviation of each feature vector component is also stored. This is required so

that new image feature vectors can be appropriately re-scaled and classified to one of the existing clusters designated by the centroids.

At image mosaic generation time, these cluster labels are used to reduce the number of images that must be compared to each target image tessera. The method by which this is accomplished is described in detail during the discussion on MosaiX image retrieval in the next major subsection of this chapter.

Products of Preprocessing

The preprocessing phase of the MosaiX system produces as output two separate databases of information. The first database, henceforth referred to as the *image library*, contains the resolution-matched copies of the original collection of images supplied by the user. The second database contains the extracted feature vectors of each image within the image library. Additionally, cluster information in the form of a cluster label for each feature vector, the corresponding centroids for each of the k labels, and the feature vector re-scaling parameters are also stored into this database. The information stored in this second database is collectively referred to as the *image metadata* of the image library. These two databases are illustrated in the central portion of the MosaiX architecture and flow diagram seen in Figure 16. During the image mosaic generation phase of the MosaiX system, the metadata will be used to perform image retrieval and the image library will be used to assemble the final output mosaic.

Image Mosaic Generation Phase

With preprocessing of a user supplied library complete, the image mosaic generation phase of the MosaiX system can be invoked. Generation of an image mosaic requires the user to supply a series of input parameters. First and foremost, the user must supply a desired input target image to be replicated as an image mosaic. Next, the user needs to specify the number of tesserae he/she wishes to use in the output image mosaic. Finally, a desired image library and its associated metadata must be selected so that MosaiX knows from where to select its replacement images during the image mosaic generation process. After all required parameters have been collected from the user, the process of creating an image mosaic can initiate. Three major sub-processes exist in this phase: tessellation, image retrieval, and mosaic assembly. Tessellation divides the target image into many smaller images called tessera, image retrieval identifies an image with similar visual content to each tessera, and mosaic assembly uses the retrieved images to build the final output image mosaic. A high-level flow diagram of these image mosaic generation steps is depicted in Figure 21. These tasks are each explored in detail in the next several subsections.

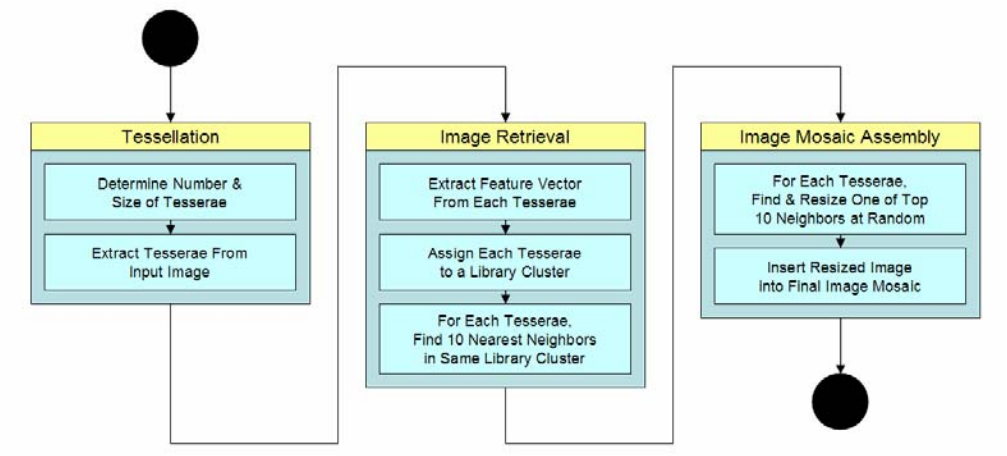

Figure 21: High-level flow diagram of the MosaiX image mosaic generation phase.

Tessellation

Tessellation is the stage of image mosaic generation when the user-supplied input target image is tessellated (subdivided) into a series of smaller images, referred to as tesserae. Just how many tesserae MosaiX divides the target image into is determined by the user. One simply inputs how many subdivisions he wishes to use in either the horizontal or vertical direction, and the MosaiX system then calculates the exact number of tesserae required in the other direction, based on the aspect ratio of the images in the chosen image library and the aspect ratio of the target image selected by the user. For example, consider if a user had elected to use an image library of square (1:1 aspect ratio) images on a target 3000x2000 pixel (3:2 aspect ratio) image and desired an image mosaic with 100 tesserae in the horizontal direction. The width of a tessera could then be calculated at 30 pixels wide, which is required to fit 100 tesserae into a space 3000 pixels wide. Given that the image library contains images that are square, one can deduce that a tessera would also need to be square, thus giving us final tessera dimensions of 30x30 pixels. Using the calculated tessera dimensions, MosaiX determines the number of tesserae that will be required in the vertical direction by dividing the height of the target image by the height of the tesserae. In the above example, this would yield a result of 66.6 tesserae in the vertical direction. We do not wish to use partial tessera in the final image mosaic and thus this number is truncated to 66. This truncation results in the output mosaic of this example being slightly smaller in the vertical direction than the target input image.

With the exact number of tesserae required in each direction now known, the tessellation can begin. Tesserae are extracted starting from the top left corner of the target image and proceeding to the right, moving down to the next tesserae row as required. The result of the

tessellation process is a collection of tesserae images (illustrated in Figure 22), each representing a small portion of the target image. In addition to the tesserae images themselves, the MosaiX system also keeps track of the spatial location of each individual tessera relative to the original target image. This spatial information will be used during the mosaic assembly process to ensure replacement images are correctly placed in the final image mosaic.

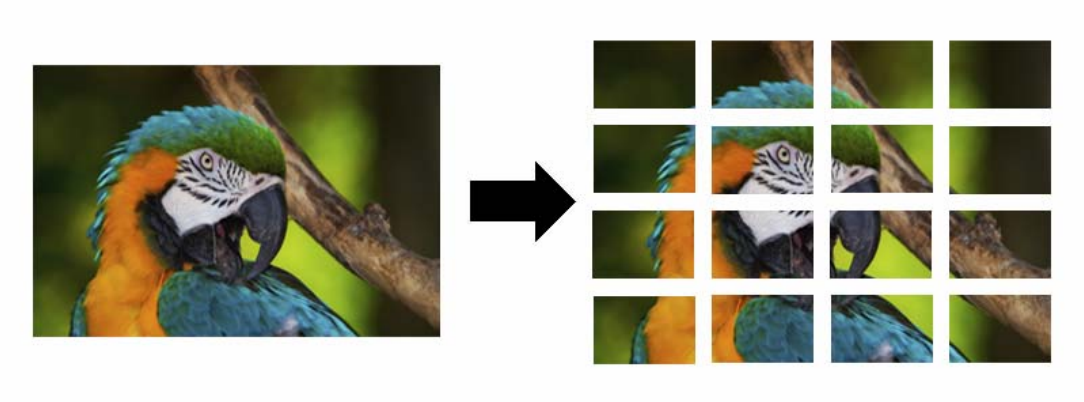

Figure 22: The 4x4 tessellation of an image into tesserae.

Image Retrieval

During the image retrieval stage, the most visually similar image for each individual tesserae is identified from the selected image library. To accomplish this goal, the MosaiX system must first perform the exact same feature extraction process on each tesserae as was done for the library images during the preprocessing stage, producing a feature vector for each tesserae. Next, using the standardization data stored in the image library metadata, each tessera's feature vector is re-scaled with the same parameters as were used on the library feature vectors, thus mapping them to the same feature space.

In order to quickly eliminate library images that have a very low probability of being good matches, each tessera's feature vector will be assigned to one of the clusters generated during library preprocessing. This assignment is made by determining the Euclidean distance between each possible cluster centroid and each tessera's feature vector. Cluster labels are assigned to each tessera using the cluster label of the nearest corresponding centroid. This step effectively classifies each tessera to some grouping of images within the image library that is described by one of these cluster centroids. As a result, one can now limit direct comparisons of a given tessera's feature vector to the subset of library feature vectors that represent the images most likely to be close visual matches.

For direct comparison of tessera feature vectors to a subset of library feature vectors, a simple weighted L_1-distance metric (as seen in Equation (36)) is employed. This measure compares the three extracted distribution features (mean E, standard deviation σ, and skewness s) in each color channel (indexed by k) from each of the five image sub-regions (indexed by r). The values of the weights were hand tuned over several experimental runs and in several different color models. Table 1 contains the values of the weight components used in the final MosaiX system. As previously mentioned, only image library feature vectors from the same cluster label are compared, this distance equation resulting in a significant increase in execution time since unlikely images are ignored. After comparing all feature vectors from the desired cluster, the indices of the ten feature vectors with the smallest weighted distance are recorded. This process is then repeated until all tesserae have been processed.

$$L_1^w(A,B) = \sum_{r=1}^{5}\sum_{k=1}^{3} w_E\left|E_{rk}^A - E_{rk}^B\right| + w_\sigma\left|\sigma_{rk}^A - \sigma_{rk}^B\right| + w_s\left|s_{rk}^A - s_{rk}^B\right| \qquad (36)$$

After all tesserae from the input target image have been processed and their top five closest matching library images have been identified, MosaiX moves on to its finishing stage where the final output image mosaic is assembled.

Table 1: Feature weight values used in the MosaiX distance metric

w_E	w_σ	w_s
1.00	0.25	0.15

Mosaic Assembly

The image mosaic assembler function of the MosaiX system is, by far, the simplest of the major system sub-processes. During this stage, the images selected as best matches during the image retrieval stages are spatially reassembled to form the final output image mosaic. To do this, MosaiX selects, at random, one of the top ten indices returned for each tesserae. This random selection from the top ten possible matches is done to overcome the problem of the same library image matching to several neighboring tesserae. Figure 23 shows this problem and the effect of the proposed solution on an example image mosaic. Next, images corresponding to each of these indices are retrieved from the resolution-matched image library and resized to the same dimensions as a single tessera. These resized images are then placed into their correct positions in the final image mosaic, which is then saved as an uncompressed TIFF image. The saved image can then be printed or digitally displayed as the user desires.

Figure 23: Difference in assembly from single match (A) and from a set of matches (B).

Products of Image Mosaic Generation

The image mosaic generation phase of the MosaiX system produces two files as output. The first is the previously mentioned uncompressed TIFF image file of the completed image mosaic. In addition, a basic data file is also created that contains relevant information about the creation of its corresponding image mosaic. Recorded into this data file are: image library used, color model used, generation time, filename of the target image, and a matrix with the indices the images selected for use in the mosaic reproduction (in their correct spatial locations). The data file allows for the image mosaic to be recreated in TIFF form should the original file be lost.

CHAPTER FOUR: EXPERIMENTAL RESULTS

The design of the MosaiX system allows for three key input parameters to be user defined: the number of images within the image library used, the number of clusters to divide that library into, and the color model used during color feature extraction. Within this chapter, execution time and image mosaic quality are explored as functions of these three parameters. As little research exists in the realm of image mosaic generation, no explicit methodology by which to gauge an image mosaic's quality has ever been defined. As a result, this chapter proposes a completely new strategy for quantifying the quality of an image mosaic with respect to its intended target image.

Measurements and Methods

The execution time required for generating a complete image mosaic is a relatively straightforward measurement to compute. In the case of these experiments, execution time is defined as the number of seconds elapsed between the start of the target image's tessellation stage and the completion of the image retrieval stage. Because the task of mosaic assembly is constant with regards to the three input parameters explored, it is therefore not included in the execution time measurements.

The evaluation of a completed image mosaic's quality, unlike the clear-cut calculation of execution time, is not a trivial task. A naive approach would be to simply calculate the RMS error between the intended target image and its mosaic reproduction. Such an approach has been used as a metric for evaluating the effectiveness of lossy compression methods and other image processing problems. Unfortunately, the very nature of an image mosaic prevents such a simple

strategy from proving effective. What gives an image mosaic its intrigue is the fact that at close observation, each tessera has its own unique visual content that is different from the section of the target image that it replaced. This difference becomes insignificant when viewed from a distance and all tesserae come together to create final mosaic image. Judging the quality of a mosaic based simply on an RMS error between the mosaic and its intended target would penalize the very feature that makes the photographic image mosaic artistically unique.

In order to more effectively gauge the relative quality of an image mosaic with respect to its target image, a more complex procedure is required. Our image mosaic quality measure evaluates two aspects of the mosaic reproduction: color accuracy and edge accuracy. Both measurements begin by first preprocessing the original target image and the mosaic reproduction, in order to avoid the penalization of the aforementioned features that make image mosaics creatively pleasing. First the images (target and mosaic) are each filtered with a Gaussian low-pass kernel to subdue a large portion of the high-frequency information within the images. The resulting images are then down-sampled by a factor of two, before being filtered yet again with another Gaussian low-pass kernel and then down-sampled again by a factor of two. The rationale for low-pass filtering the images is to remove from them the details that would have little perceptual impact when viewed from a distance; in this case, the high-frequency details. A visual flow of this preprocessing is shown in Figure 24.

Figure 24: Mosaic quality evaluation low-frequency elimination preprocessing steps.

With the target image and mosaic reproduction preprocessed to minimize the high frequency content of both images, the quality metrics for color and edge accuracy are applied. For color accuracy, the images are transformed into CIE L*a*b* space and an RMS error (see Equation (32)) is calculated between the two low-frequency images. The CIE L*a*b* color model was designed to be perceptually uniform and have a Euclidean metric, thus making it the

most appropriate space in which to calculate the RMS error. The resulting quantity becomes the color accuracy score for all experimental runs and represents a measure of how well the mosaic perceptually resembles the target image when viewed from a distance.

Edge accuracy is measured by taking the preprocessed target and mosaic images and applying a Sobel edge detector to each image. The gradient magnitude images are thresholded to produce binary images with pixel values of one corresponding to edge pixels and values of zero corresponding to non-edge pixels. The threshold value use is determined heuristically based on the nature of the data in the intended target image. That is, whatever threshold on the target produces good edge results is also used on the mosaic reproduction. The binary edge images are further processed using a series of morphological erosions and dilations to clean up the results. Figure 25 illustrates this edge extraction process. To produce the final edge accuracy score the number of edge pixels in the mosaic reproduction that match edge pixels in the target image are divided by the total number of edge pixels in the target image. This score produces a measure of how well true edges from the target image were reproduced in the generated image mosaic.

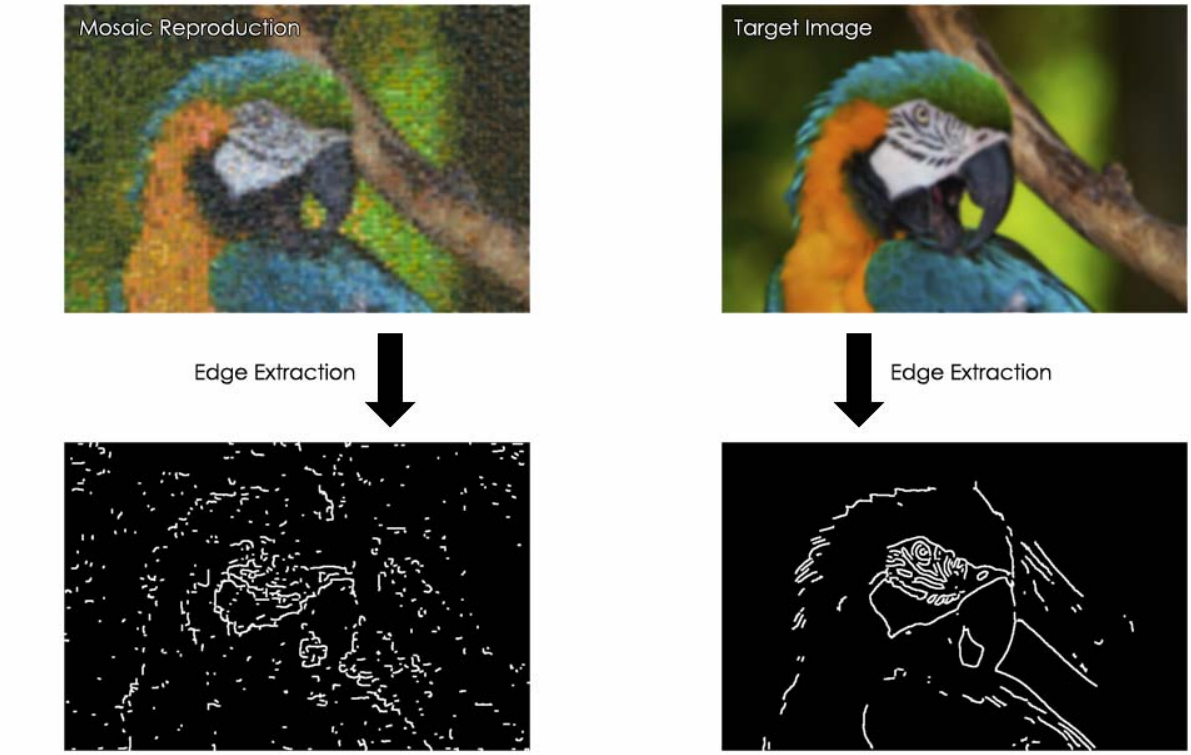

Figure 25: Visual example of the edge extraction process on a mosaic and its target.

Experiments

The image mosaic generation system, MosaiX, described in Chapter 3 was implemented using Matlab 6 within the Windows XP operating system on a PC with a 2800 MHz processor and 2 gigabytes of system memory. A large collection of images with a wide range of semantic content and color distributions was obtained and portioned into three images libraries of five thousand, ten thousand, and fifteen thousand images respectively. Each of these libraries was then resolution matched with the MosaiX preprocessor such that all images were 256x256 pixels in size. Feature extraction was then performed on each resolution-matched library using the RGB, HSV, YIQ, and CIE L*a*b* color spaces, producing feature vectors for each library in

each color space. Finally, each generated feature matrix was clustered using two, five, ten, and twenty cluster centroids. The result was image metadata for forty-eight possible permutations of library size, color model, and number of clusters.

A set of fifteen test images (see Figure 26) of varying complexity and content were obtained and used as test target images for generating image mosaics with the MosaiX system. For consistency, all test images used were approximately 3000x2000 pixels in dimension. For each test target image, a mosaic reproduction with seventy tessera horizontally and forty-three tessera vertically was produced for each for each of the forty-eight possible preprocessing combinations of library size, color model, and number of clusters. The resulting seven-hundred and twenty mosaics were scored for edge accuracy and color accuracy, as well as having their processing times recorded. The following subsections analyze these resulting mosaics to identify the effects of the different preprocessing parameters on these three measurements.

Figure 26: The fifteen test targets used in the MosaiX analysis.

Impact of Color Space

As mentioned in Chapter Two, a color model defines a regular and accepted way colors are to be specified within some coordinate system. Unfortunately, these color spaces can vary significantly depending on their intended end use. This raises the question of how a choice of color model used during feature extraction can affect the mosaic reproduction process. Our experiments focused on four different models: RGB, HSV, YIQ, and CIE L*a*b*. The affects on color accuracy, edge accuracy, and processing times with respect to color space are analyzed in the next several subsections.

Color Accuracy

Our measure of color accuracy was designed to determine perceptual dissimilarity between a mosaic reproduction and its intended target image. For this reason, it is unsurprising to see that the color models designed with human perception in mind were able to outperform other models in these terms. Figure 27 and Figure 28 clearly show that the YIQ and CIE L*a*b* color models produce mosaic reproductions that are on average eight percent closer to the intended target image than mosaics created with either the RGB or HSV models. This difference holds true independent of the size of the image library used or the number of clusters that library is divided into. Again, these results reflect the very nature of the YIQ and CIE L*a*b* color spaces in that they were designed with human perception in mind. On the contrary, a particular issue with the use of HSV space within the MosaiX system is the fact that the hue and saturation components are expressed in polar coordinates. For example, a hue angle (in radians) value near

zero is perceptually similar to a value near 2π, but our defined distance measure for retrieval (see Equation (36)) will judge these two hues to be very far apart and thus not similar at all.

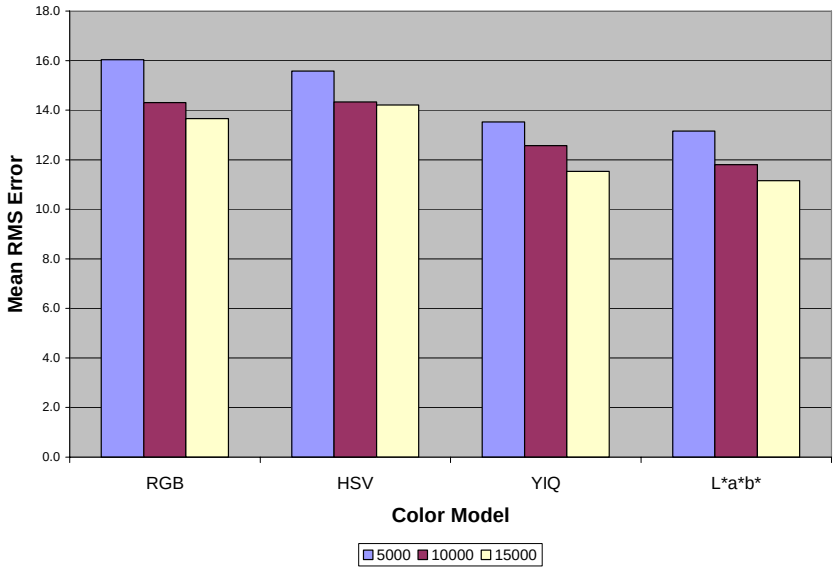

Figure 27: Mean RMS error comparison by color model and image library size.

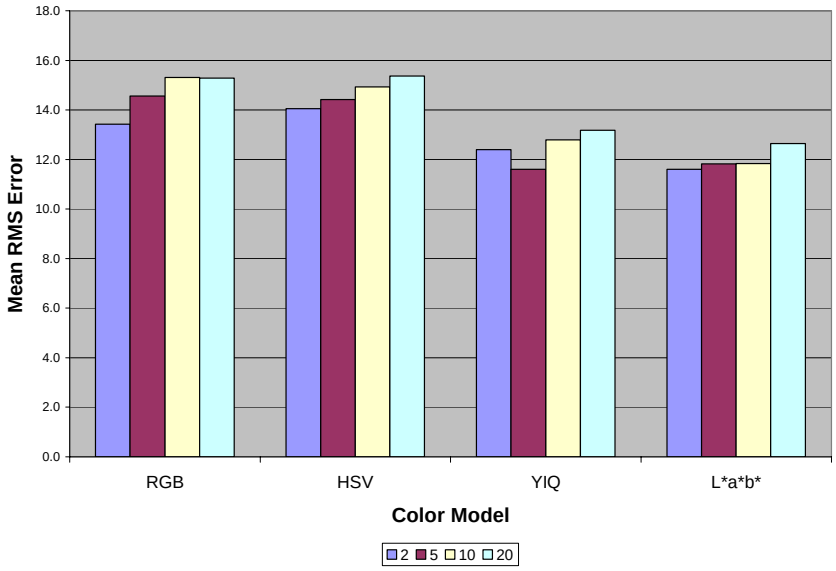

Figure 28: Mean RMS error comparison by color model and number of clusters.

Edge Accuracy

Unlike the results found when examining color accuracy, the analysis of how well a given mosaic reproduces the edge detail of its intended target yields somewhat surprising results when measured with respect to color model. From Figure 29 and Figure 30, it can be observed that the RGB color model consistently reproduces edges within a mosaic more accurately than any other tested model, regardless of the image library size or the number of clusters used. Whereas YIQ, HSV, and CIE L*a*b* produce accuracies in the range of eighteen to twenty-five percent, RGB measurements fall into the twenty-seven to thirty percent range. An obvious reason for this advantage is not clearly evident from this analysis. The RGB model must posses some feature that, when combined with our retrieval metric, produces mosaic results with stronger edge reproduction properties than all other tested models.

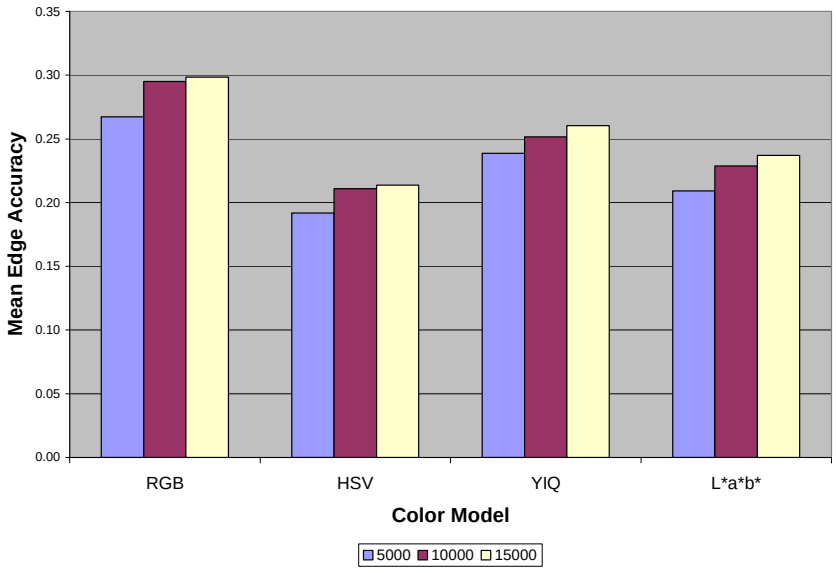

Figure 29: Mean edge accuracy comparison by color model and image library size.

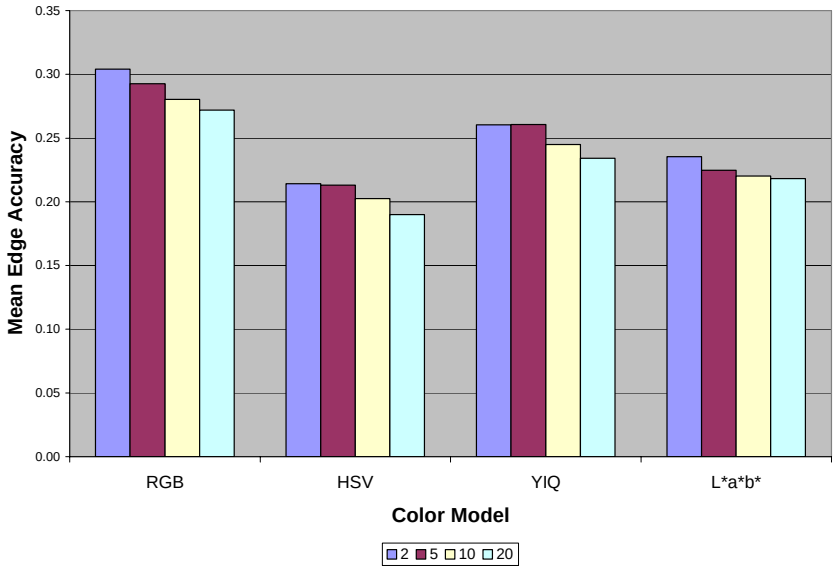

Figure 30: Mean edge accuracy comparison by color model and number of clusters.

Processing Time

Given that the size of the feature vectors produced during feature extraction and the structure of the image retrieval metric (see Equation (36)) are constant regardless of the color model used, one would expect the choice of color model to have little or no effect on the time required to generate an image mosaic. However, experimental results, as seen in Figure 38 and Figure 47, show that there is in fact a definite difference in execution time depending on the choice of color model. However, these results are really not that surprising given the nature of our testing implementation. A target image, when presented to the MosaiX system, is always initially loaded in the RGB color model. In order to extract features in any other model, the RGB coordinates must be transformed into the desired color space. Because the time required to perform this transformation is included in our measurement of execution time, results as seen in the aforementioned figures are produced. Since no transformation is required for the RGB

model, mosaics produced using this model have shorter generation times than mosaics produced in any other models (all other parameters held constant). Since the CIE L*a*b* transformation from RGB coordinates actually requires two separate transformation processes, its timings are consistently the highest. HSV and YIQ require only a single mapping, and thus have timing results that fall between CIE L*a*b* and RGB. Another factor that may play into the variance of processing time with respect to color model is perhaps that some color models may produce more equally-sized clusters than others.

Impact of Number of Clusters

In an effort to decrease the number of gross mismatches during the image retrieval stage of mosaic generation, MosaiX employs k-means clustering on image library feature vectors during its preprocessing phase. This clustering endeavors to group together those feature vectors (representing the library images) that are very close together within the feature space. In doing so, image tesserae can be chosen, during image mosaic generation time, only from a group of images that are most likely to resemble the section of the target image it will be replacing. By the early elimination of images that would clearly be inappropriate replacement tessera, the total number of images that need to be searched is reduced, and thus faster generation times can be achieved. Just how many clusters one should divide the image library into is of great importance. Choosing too few will provide relatively slow generation times. On the other hand, if one chooses too many clusters the probability of a feature vector being inappropriately assigned (especially those on the edges of clusters) increases. Such an occurrence may result in the most appropriate replacement image for a certain tessera being removed from consideration.

Color Accuracy

As expected, when the number of clusters used to divide the image library is increased, the overall perceptual similarity between a mosaic reproduction and its intended target image will decrease. This outcome was observed in our experiments in Figure 31, Figure 32, and Figure 33 as an increase in the mean error value when the number of clusters is made larger. As previously explained, the cause of this phenomenon is that as the number of clusters grows, so too does the probability that an image will be labeled to an inappropriate cluster. This holds especially true for images that lie on the outer edges of a cluster's domain. In such cases, an image may have more in common with neighboring images from another cluster (on the edges of its cluster's domain) than it does with most of the images in its assigned group. These areas were cluster boundaries meet increase with the number of clusters, thus explaining the affects seen in Figure 31, Figure 32, and Figure 33. The rate at which the perceptual similarity degrades as a function of the number of clusters, however, is not excessive.

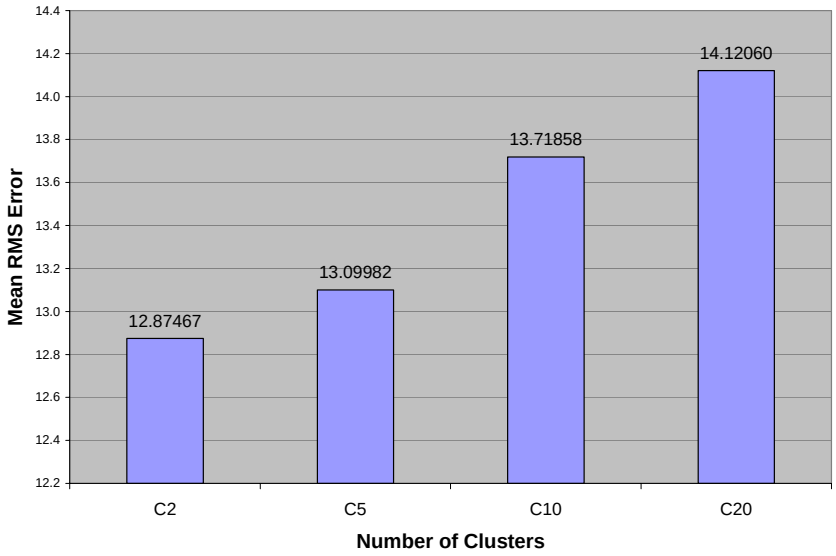

Figure 31: Mean RMS error by number of clusters.

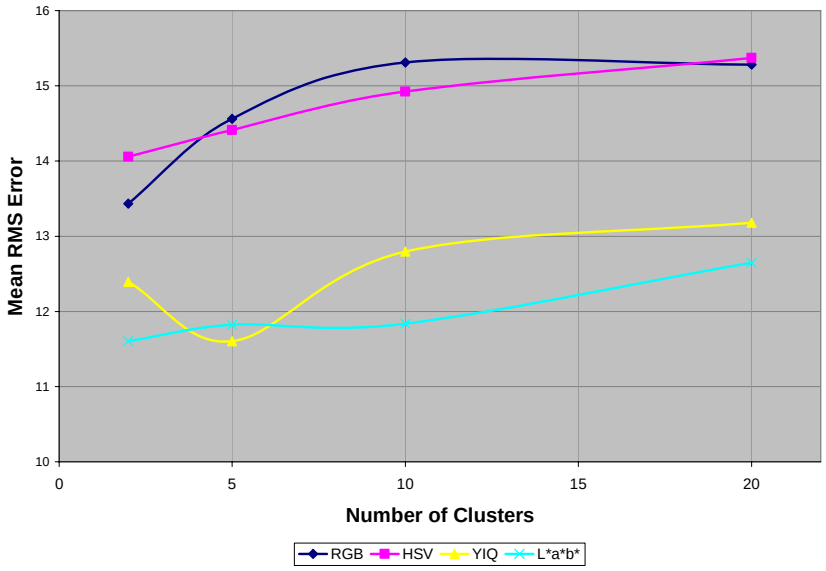

Figure 32: Mean RMS error by number of clusters for each color space.

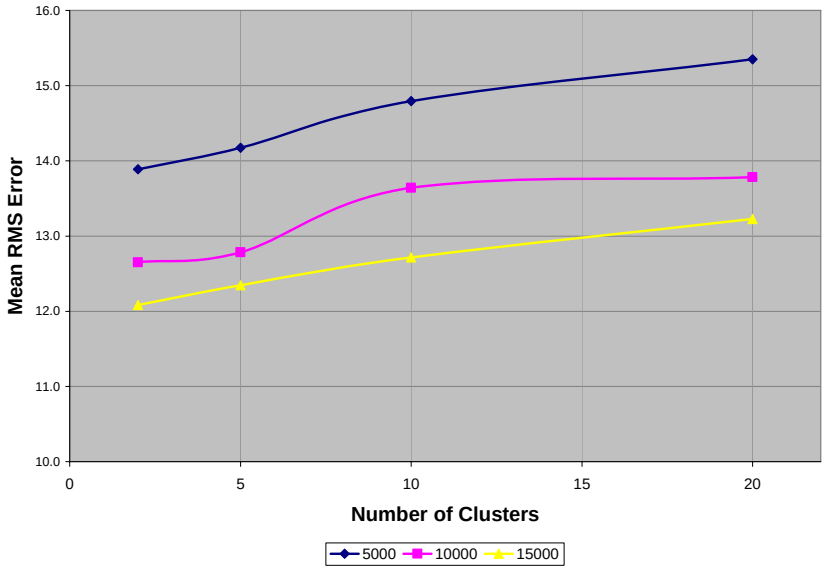

Figure 33: Mean RMS error by number of clusters for each image library size.

Edge Accuracy

Just as with color accuracy, the edge accuracy of a mosaic reproduction with respect to its target will suffer as the number of clusters used is increased. This trend is clearly reflected in our experiments in the results seen in Figure 34, Figure 35, and Figure 36. The origin of this decreased edge accuracy can be attributed to the same cause that decreased color accuracy when the number of clusters was increased. More clusters allows for a greater probability of inappropriate labeling along the boundaries of the clusters. Thus at retrieval time, the very best replacement image may not be available because it lies on the outskirts of a completely different cluster grouping. The rate at which edge accuracy is compromised, however, is relatively low. Figure 34 shows a decrease in accuracy of only three percent despite a tenfold increase in the number of clusters used to subdivide the image library.

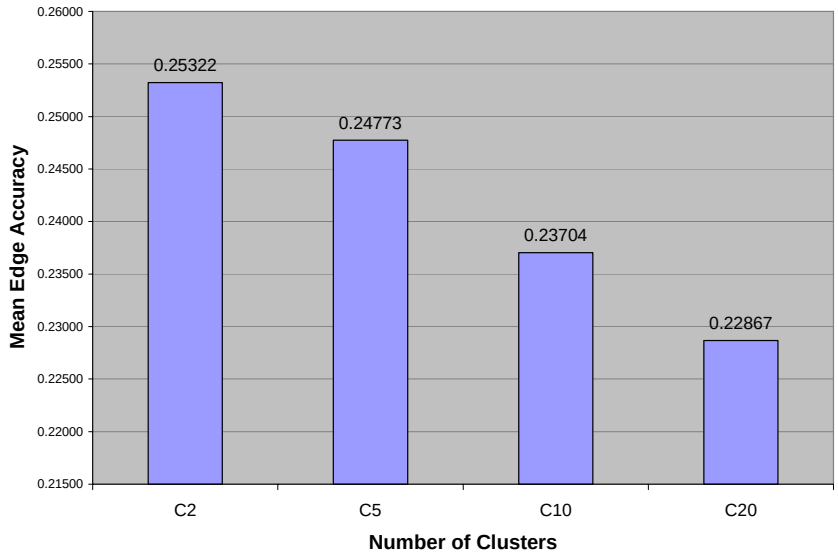

Figure 34: Mean edge score by number of clusters.

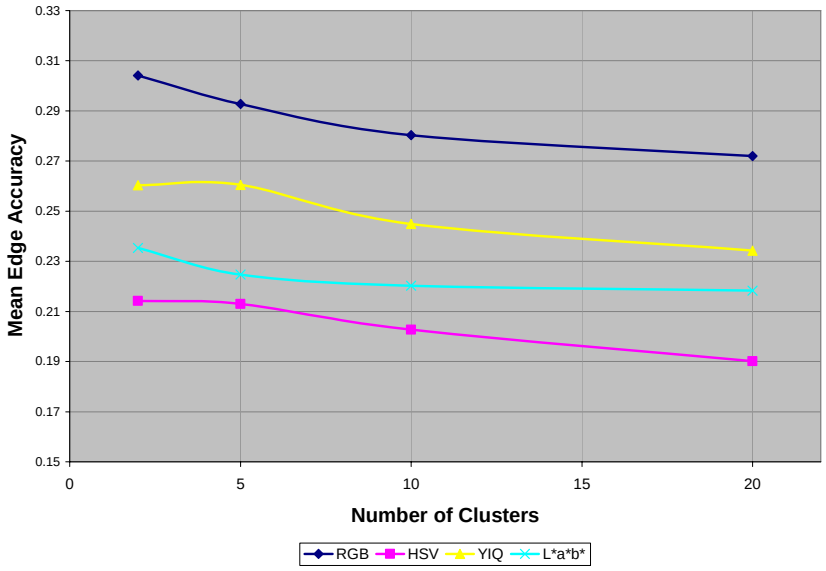

Figure 35: Mean edge score by number of clusters for each color space.

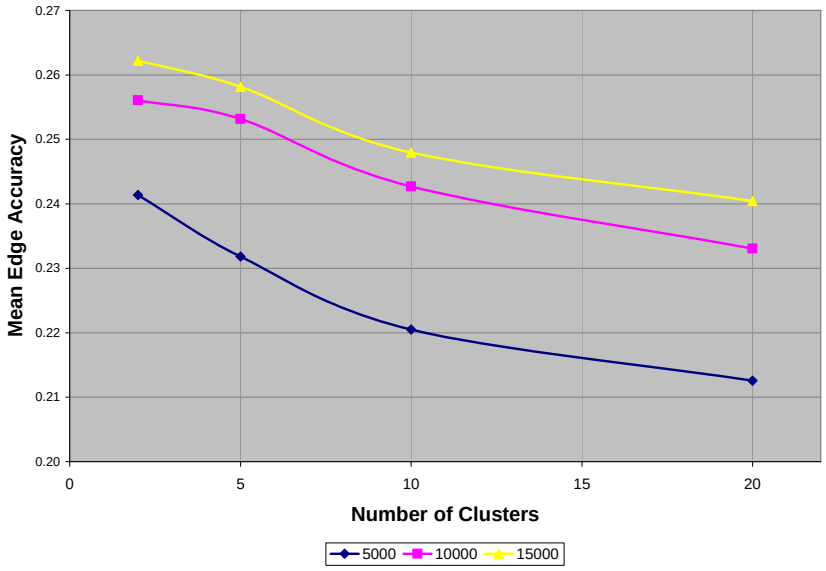

Figure 36: Mean edge score by number of clusters for each image library size.

Processing Time

The use of image library clustering within the MosaiX system was intended to noticeably reduce the required processing time needed to generate a complete image mosaic. By clustering the image library into groups containing only those images that have some degree of similarity among them, the system can identify and search from the cluster that contains images most like its current target. Fewer images to search naturally leads to an decrease in search time. Figure 37, Figure 38, and Figure 39 each clearly show a decrease in the required execution time as the number of clusters is increased regardless of the image library size or the color space used. Interestingly, the rate by which execution time is decreased does not remain constant as the number of clusters is increased. Our experiments revealed an average decrease of approximately one-hundred and twenty seconds when the number of clusters was increased from two to five; this is quite a dramatic increase. However, when the number of clusters quadrupled (from five to twenty), the average decrease in execution time is only around fifty seconds. Resembling an exponential decay, this trend is likely a result of the k-means algorithm used to form the groupings. That is, the number of images contained in a cluster will not necessarily decrease as the number of clusters is increased. For example, if a large grouping of points are very near each other (high density) compared to other points in the clustering space (low density), it is very unlikely that this grouping will ever be broken apart. Instead, other, less densely packed clusters would be first broken apart. This results in searches on the densely-packed clusters remaining relatively constant as the number of clusters is increased, and thus producing results like those seen in Figure 38, and Figure 39.

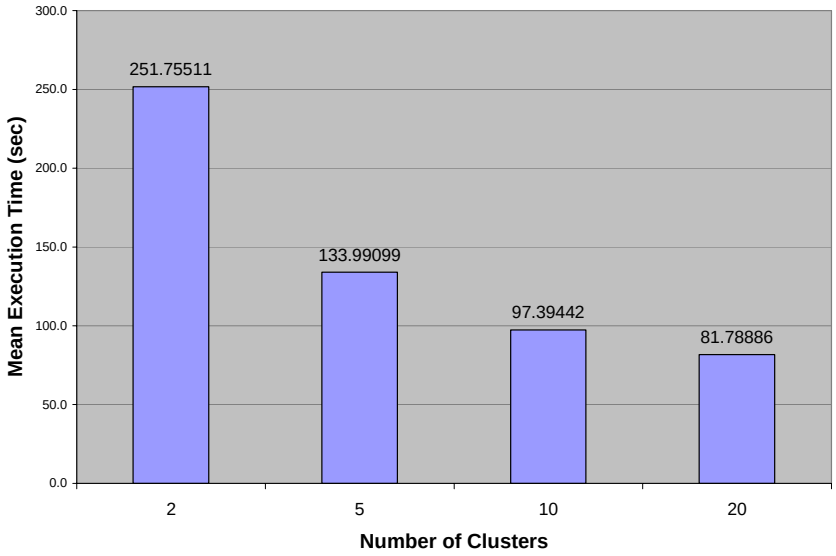

Figure 37: Mean execution time by number of clusters.

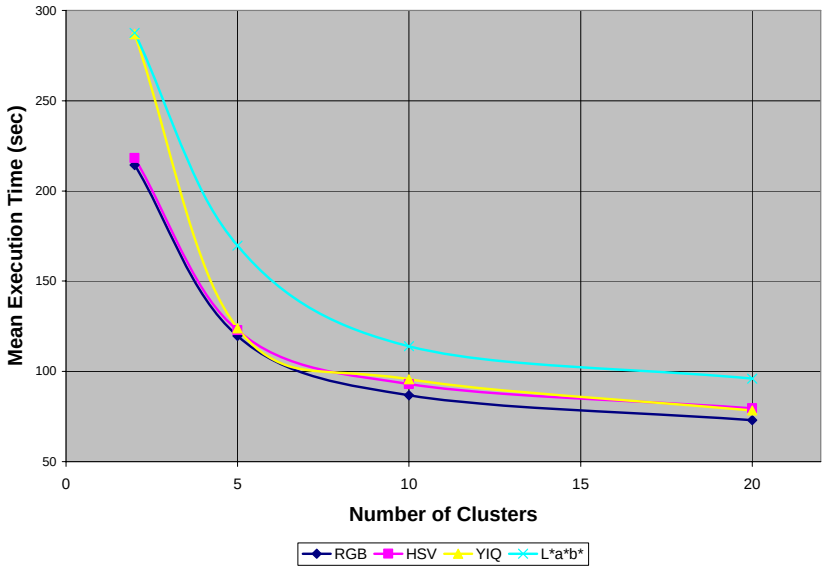

Figure 38: Mean execution time by number of clusters for each color space.

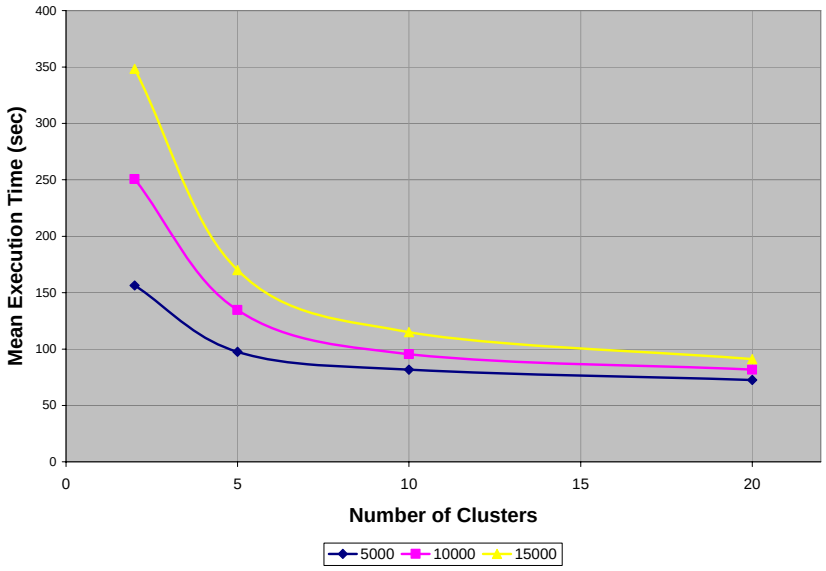

Figure 39: Mean execution time by number of clusters for each image library size.

Impact of Image Library Size

The size of the image library used for finding replacement tesserae for an image mosaic has perhaps the most obvious effects on the resulting reproduction. More images to choose from would no doubt lead to a more accurate imitation of the intended target, simply due to there being a higher probability of finding a closer match within a larger search area. On the other hand, the larger the number of images, the longer it takes to search the entire area. This is a fundamental trade-off that would be found in just about any system designed to generate image mosaics. The effect of this parameter, with respect to our defined measurements for quality and execution time, is explored in the next several subsections.

Color Accuracy

Our experiments generated image mosaics using search libraries of five thousand, ten thousand, and fifteen thousand images. Results for color accuracy, seen as error measures in Figure 40, Figure 41, and Figure 42, confirm the intuition that a larger image library will result in mosaic reproductions that are perceptually closer to their target image. Certainly increasing the number of possible images to select from results in an increased probability of finding a better matching image for placement into the final image mosaic. The observed improvements in accuracy, however, are not overly extreme, as evident by the relatively shallow slopes of the lines in Figure 41, and Figure 42. Improvements averaging only around seven percent were seen as the number of library images was first doubled, and then tripled, from its original size. It should be noted that visual observation of the mosaics generated (see Appendix) shows that even the smallest image library produced aesthetically pleasing results despite the reduced search area.

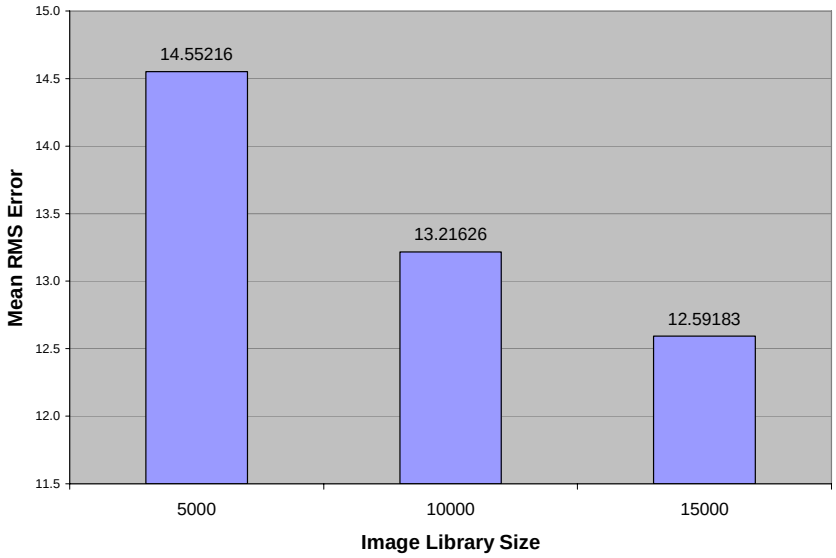

Figure 40: Mean RMS error by size of image library.

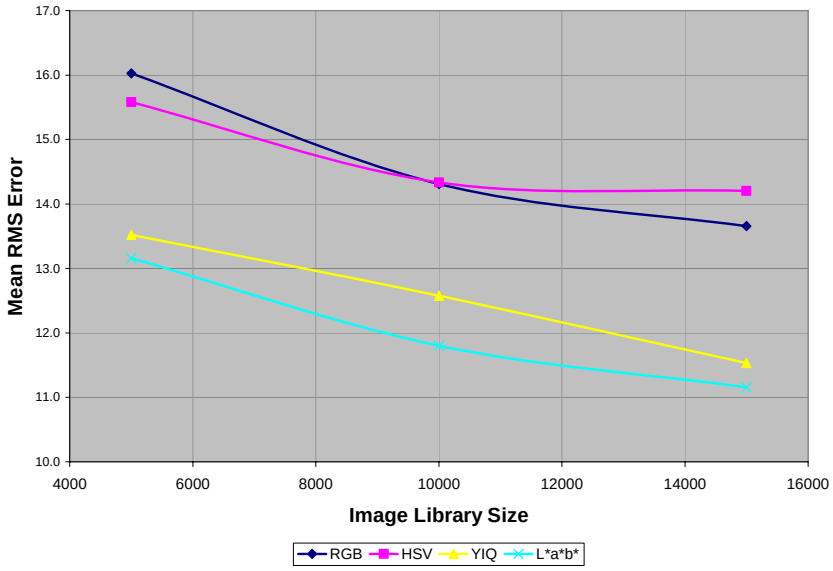

Figure 41: Mean RMS error by size of image library for each color space.

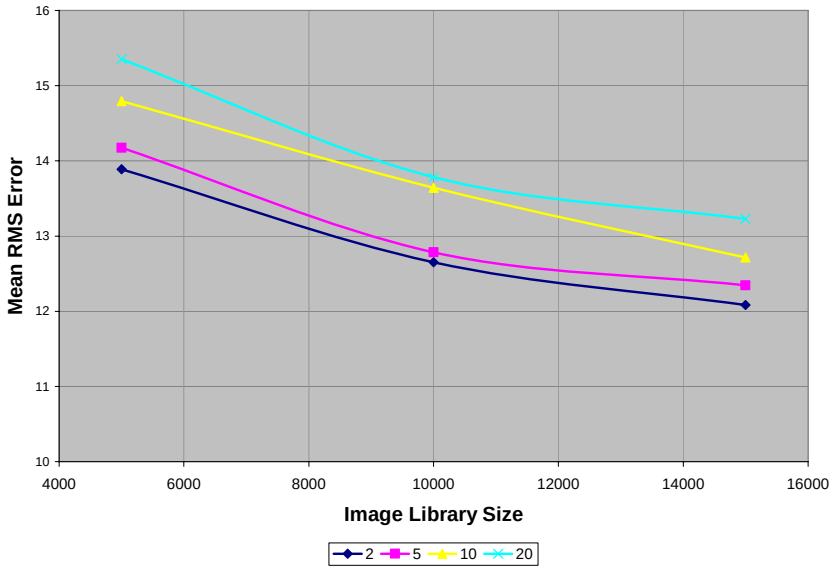

Figure 42: Mean RMS error by size of image library for each number of clusters.

Edge Accuracy

Just as was observed in the investigation of color accuracy with regards to the size of the image library used, an increase in edge accuracy occurs as the number of library images is increased. Measurements, as seen in Figure 43, Figure 44, and Figure 45, clearly show that as the number of library images goes up, so does the edge accuracy of the resulting image mosaic reproduction. Again, this outcome can be attributed to the increased probability of finding a more suitable tile in a larger collection as apposed to a smaller one. Each increase of five thousand images to the original collection produced an average increase in edge accuracy of approximately one percent. However, on average a larger increase in accuracy was seen in the initial jump from five thousand to ten thousand library images than from ten thousand to fifteen thousand library images.

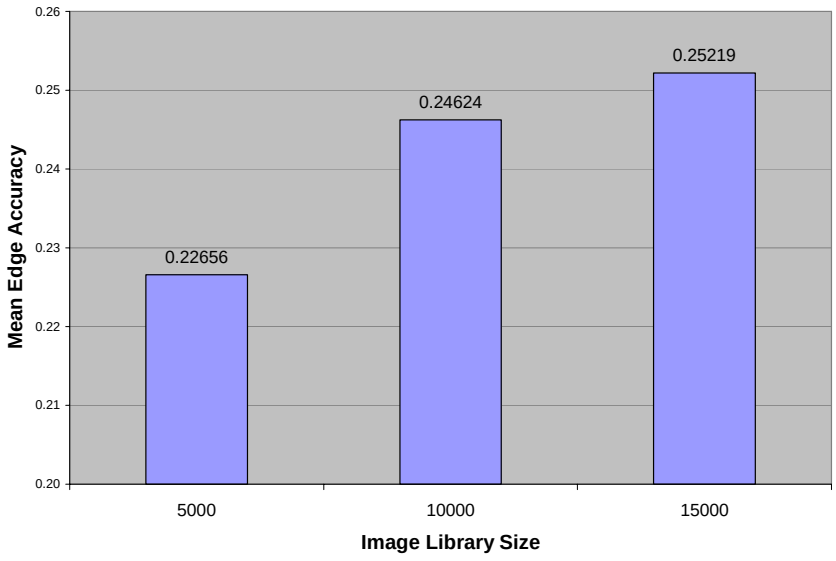

Figure 43: Mean edge accuracy by size of image library.

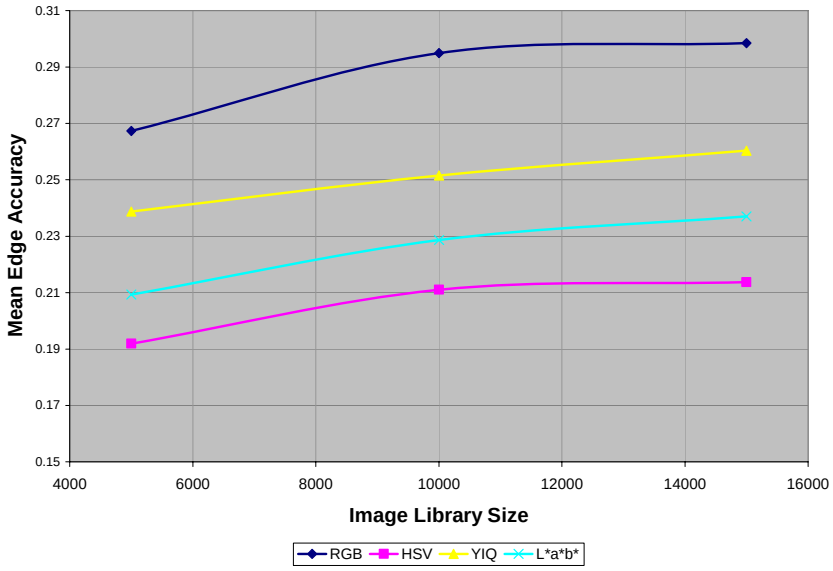

Figure 44: Mean edge accuracy by size of image library for each color space.

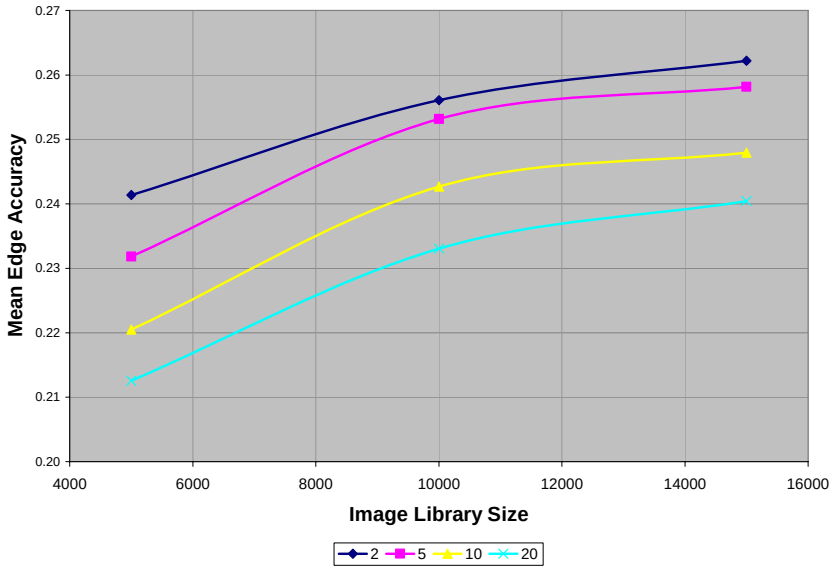

Figure 45: Mean edge accuracy by size of image library for number of clusters.

Processing Time

Simply put, the larger an image collection is, the longer it will take to look through each and every image. Experimental results of generated by our system confirm this effect, as seen in Figure 46, Figure 47, and Figure 48. However, an interesting trend is noticed in Figure 46. When the number of images in the library is doubled, the average time to generate a mosaic does not double; instead, the increase is in the range of only thirty-seven percent. Such a low rate of increase does not seem logical until one observes the trends seen in Figure 48. The rate of increase in processing time is dependent on the number of clusters used to subdivide the image library. With only two clusters used, the difference in processing time is, on average, roughly doubled when the size of the library is tripled from five thousand to fifteen thousand. On the other hand, when a full twenty clusters are employed the processing time is increased by approximately twenty percent.

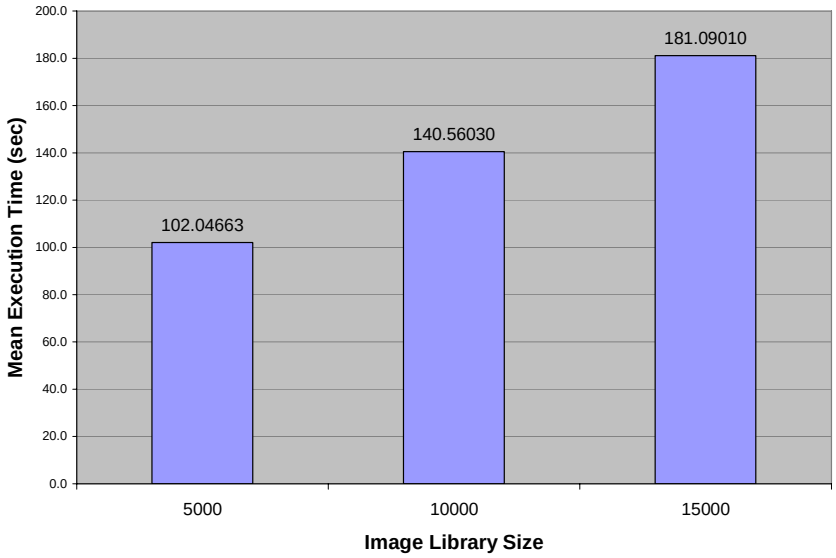

Figure 46: Mean execution time by image library size.

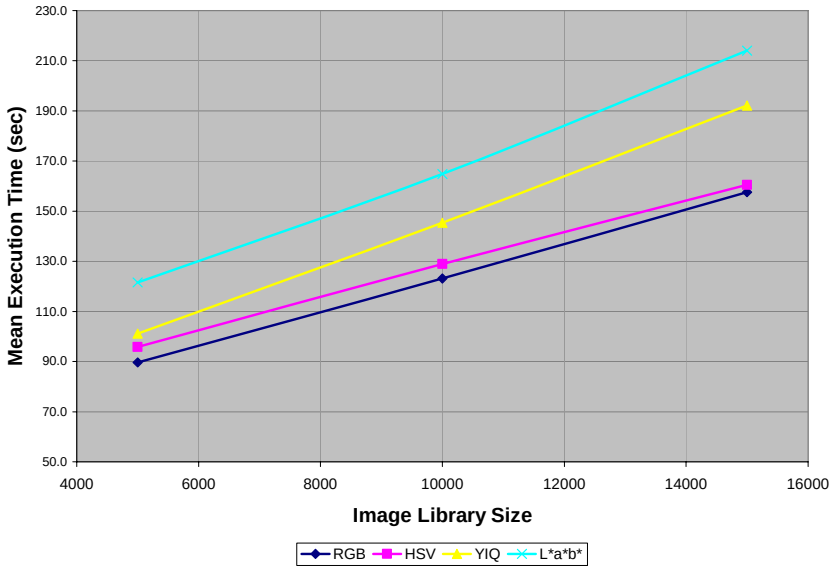

Figure 47: Mean execution time by image library size for each color space.

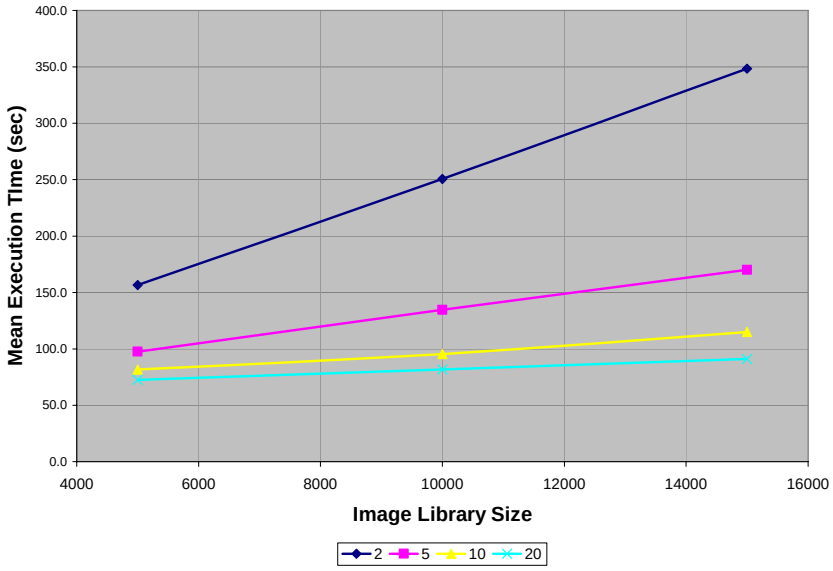

Figure 48: Mean execution time by image library size for each number of clusters.

CHAPTER FIVE: CONCLUSIONS AND FUTURE WORK

Chapter Four of this thesis provided an analysis of the effect various preprocessing input parameters had on the image mosaic generation system, MosaiX, proposed in Chapter Three. This final chapter will draw conclusions from the experimental analysis, as well as summarize the contributions of this thesis and suggest future avenues of work that could take place in the area of image mosaics research.

The Impact of Preprocessing Parameters

The MosaiX image mosaic generation system has been experimented and analyzed with respect to three important preprocessing parameters: the color model used for feature extraction, the size of the image library for which to search for replacement images, and the number of clusters used to subdivide the image library. Analysis showed that all three were able to impact both the quality (in terms of color and edges) and the time required to generate the mosaic reproduction. However, the degree by which each measurement was affected was different for each parameter. Table 2 summarizes the relative degree by which each parameter affected the measurements in terms of *high*, *medium*, or *low* impact.

Table 2: Relative impact of preprocessing parameters on measured quantities.

	Color Model	**Library Size**	**Number of Clusters**
Color Accuracy	High	Medium	Medium
Edge Accuracy	High	High	Medium
Processing Time	Low	Medium	High

In terms of color accuracy, the choice of color model was the most important parameter to consider. Although the number of clusters and the size of the image library did affect this measurement, the color model used provided the most apparent difference in perceptual similarity to the mosaic's target image. The CIE L*a*b* and YIQ color models were able to consistently produce more accurate mosaics than either RGB or HSV. Edge accuracy was also highly dependent on the color model used, with the RGB color space far out performing the others tested. Size of the image library used was, to a slightly lesser extent, able to improve edge accuracy of a resultant mosaic. Figure 36 reveals that mosaics generated with a library of five thousand images produce noticeably less accurate edges (in terms of our measure) than those produced with the larger library sizes. While the number of clusters used to subdivide each library did affect both quality measures, its influence was far less striking.

Despite its dominant control over the quality, the choice of color model failed to significantly influence the time required to generate an image mosaic. In this case, the number of clusters used for library subdivision had the greatest effect on execution times. When the number of clusters was increased from two to five, the average execution time over all tested libraries and color models was reduced by nearly half. Increasing the number of clusters to twenty allowed for the size of image library to have almost no affect on overall processing time (see Figure 39). However, when the number of clusters used was smaller, the size of the image library did have a more significant influence on average generation time.

Thus, a user who wishes to create the most accurate image mosaic and has no time limitations should use either the YIQ or CIE L*a*b* color model for feature extraction, a small number of clusters, and as large an image library as possible. For a user simply wishing to

simply generate mosaics at a high rate, the number of clusters should be increased and the size of the image library reduced.

Future Work

This thesis has focused on the design and implementation of the image mosaic generation system, MosaiX. Several parameters are experimented with and their effects on the quality and processing time of a mosaic reproduction are analyzed. Two methods were introduced which allowed for the quality of a resultant mosaic to be quantitatively measured. Despite these contributions, there are additional areas of the MosaiX system (and image mosaic generation in general) that may be worth further investigation. For example, the weights (seen in Table 1) used in the distance metric used for image retrieval were chosen heuristically based on personal observation. Quality results could be improved if some type of supervised learning technique was used to determine the weights specifically for each color model. Another area of the MosaiX system worth further investigation is the clustering algorithm used to subdivide the image library. Currently, only a basic k-means algorithm is employed. A more sophisticated clustering technique may allow for larger increases in mosaic generation speed with less degradation of mosaic quality.

APPENDIX: EXAMPLE IMAGE MOSAICS

Figure 49: Reference target image for Appendix mosaics.

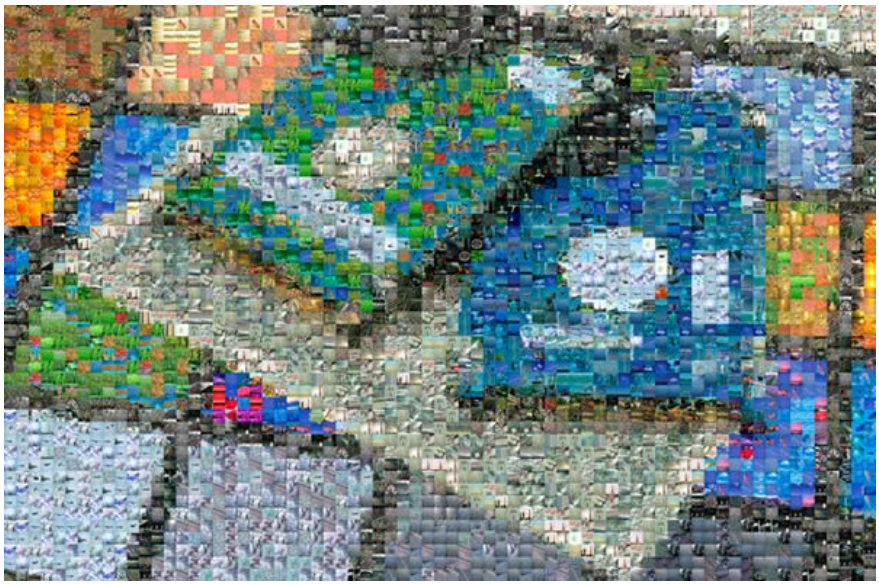

Figure 50: Image mosaic - library size 5000, color model HSV, 2 clusters

Figure 51: Image mosaic - library size 5000, color model HSV, 5 clusters.

Figure 52: Image mosaic - library size 5000, color model HSV, 10 clusters.

Figure 53: Image mosaic - library size 5000, color model HSV, 20 clusters.

*Figure 54: Image mosaic - library size 5000, color model L*a*b*, 2 clusters.*

*Figure 55: Image mosaic - library size 5000, color model L*a*b*, 5 clusters.*

*Figure 56: Image mosaic - library size 5000, color model L*a*b*, 10 clusters.*

*Figure 57: Image mosaic - library size 5000, color model L*a*b*, 20 clusters.*

Figure 58: Image mosaic - library size 5000, color model RGB, 2 clusters.

Figure 59: Image mosaic - library size 5000, color model RGB, 5 clusters.

Figure 60: Image mosaic - library size 5000, color model RGB, 10 clusters.

Figure 61: Image mosaic - library size 5000, color model RGB, 20 clusters.

Figure 62: Image mosaic - library size 5000, color model YIQ, 2 clusters.

Figure 63: Image mosaic - library size 5000, color model YIQ, 5 clusters.

Figure 64: Image mosaic - library size 5000, color model YIQ, 10 clusters.

Figure 65: Image mosaic - library size 5000, color model YIQ, 20 clusters.

Figure 66: Image mosaic - library size 10000, color model HSV, 2 clusters.

Figure 67: Image mosaic - library size 10000, color model HSV, 5 clusters.

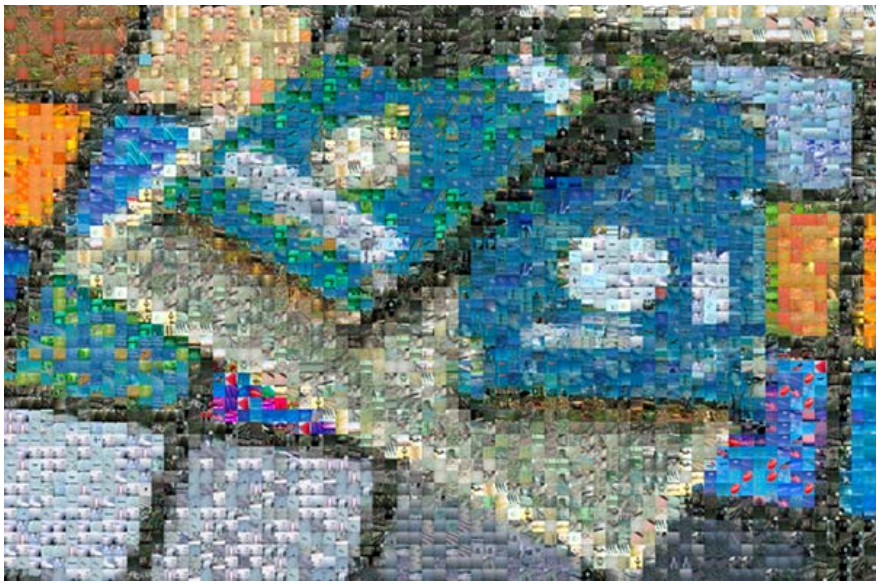

Figure 68: Image mosaic - library size 10000, color model HSV, 10 clusters.

Figure 69: Image mosaic - library size 10000, color model HSV, 20 clusters.

*Figure 70: Image mosaic - library size 10000, color model L*a*b*, 2 clusters.*

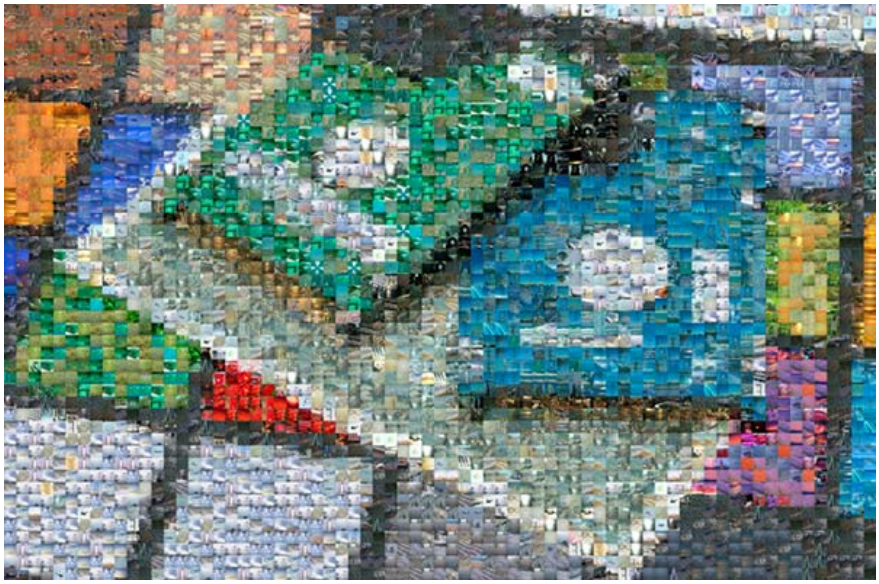

*Figure 71: Image mosaic - library size 10000, color model L*a*b*, 5 clusters.*

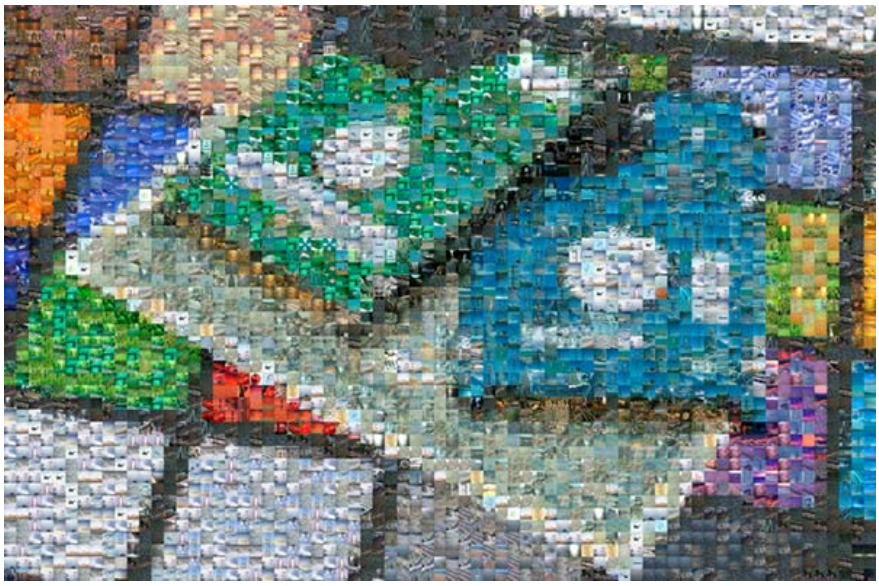

*Figure 72: Image mosaic - library size 10000, color model L*a*b*, 10 clusters.*

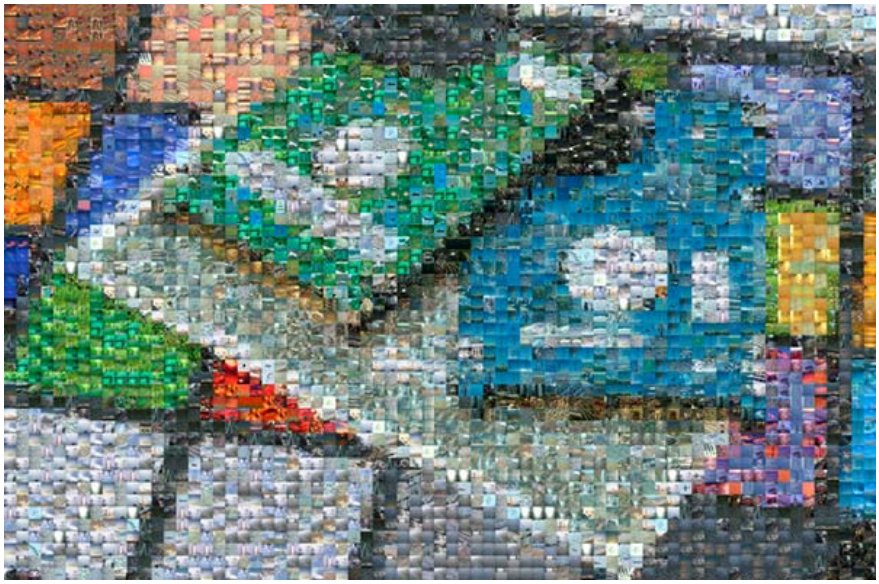

*Figure 73: Image mosaic - library size 10000, color model L*a*b*, 20 clusters.*

Figure 74: Image mosaic - library size 10000, color model RGB, 2 clusters.

Figure 75: Image mosaic - library size 10000, color model RGB, 5 clusters.

Figure 76: Image mosaic - library size 10000, color model RGB, 10 clusters.

Figure 77: Image mosaic - library size 10000, color model RGB, 20 clusters.

Figure 78: Image mosaic - library size 10000, color model YIQ, 2 clusters.

Figure 79: Image mosaic - library size 10000, color model YIQ, 5 clusters.

Figure 80: Image mosaic - library size 10000, color model YIQ, 10 clusters.

Figure 81: Image mosaic - library size 10000, color model YIQ, 20 clusters.

Figure 82: Image mosaic - library size 15000, color model HSV, 2 clusters.

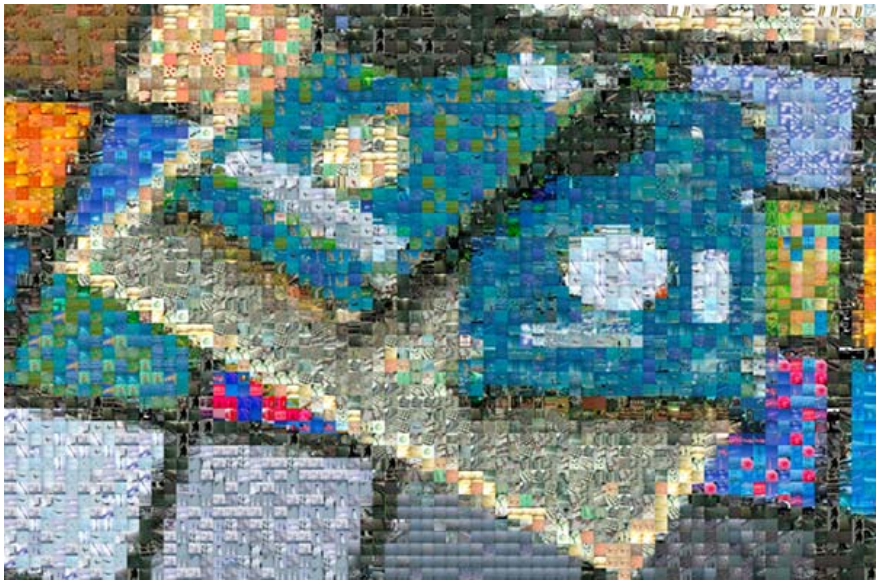

Figure 83: Image mosaic - library size 15000, color model HSV, 5 clusters.

Figure 84: Image mosaic - library size 15000, color model HSV, 10 clusters.

Figure 85: Image mosaic - library size 15000, color model HSV, 20 clusters.

*Figure 86: Image mosaic - library size 15000, color model L*a*b*, 2 clusters.*

*Figure 87: Image mosaic - library size 15000, color model L*a*b*, 5 clusters.*

*Figure 88: Image mosaic - library size 15000, color model L*a*b*, 10 clusters.*

*Figure 89: Image mosaic - library size 15000, color model L*a*b*, 20 clusters.*

Figure 90: Image mosaic - library size 15000, color model RGB, 2 clusters.

Figure 91: Image mosaic - library size 15000, color model RGB, 5 clusters.

Figure 92: Image mosaic - library size 15000, color model RGB, 10 clusters.

Figure 93: Image mosaic - library size 15000, color model RGB, 20 clusters.

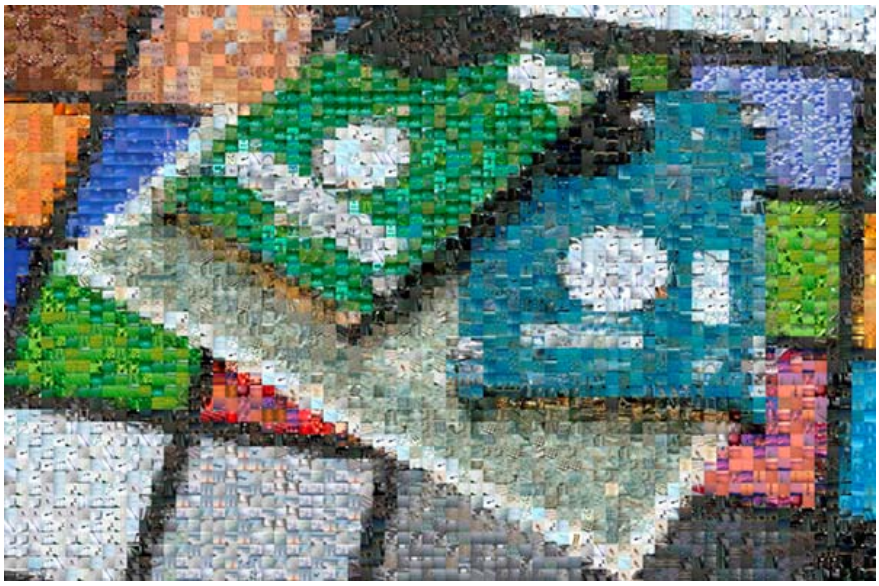

Figure 94: Image mosaic - library size 15000, color model YIQ, 2 clusters.

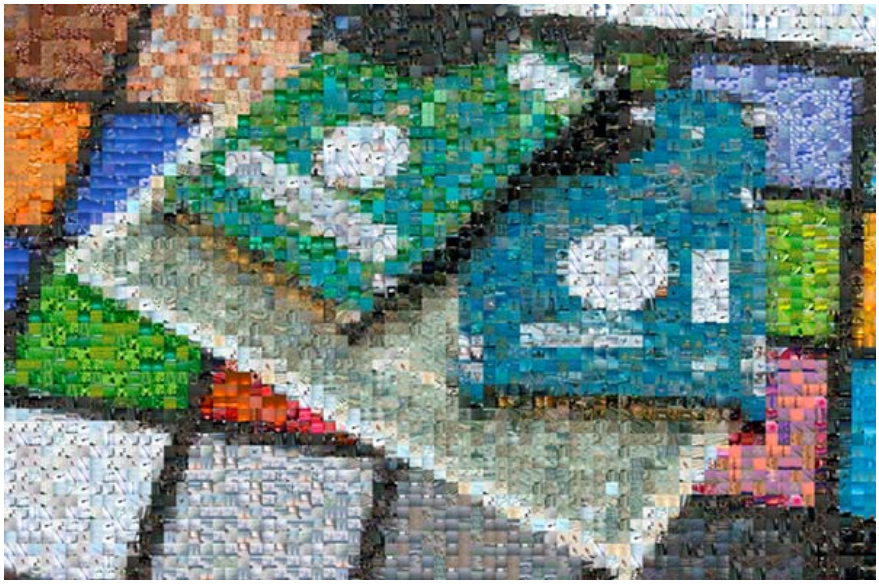

Figure 95: Image mosaic - library size 15000, color model YIQ, 5 clusters.

Figure 96: Image mosaic - library size 15000, color model YIQ, 10 clusters.

Figure 97: Image mosaic - library size 15000, color model YIQ, 20 clusters.

LIST OF REFERENCES

[1] R. Silvers. *Digital Composition of a Mosaic Image,* US Patent and Trademark Office, http://www.uspto.gov, 2000.

[2] K.I. Laws. *Rapid Texture Identification*. Proc. SPIE Conf. Image Processing for Missile Guidance, pp. 376-380, 1980.

[3] L.G. Shapiro and G.C. Stockman. *Computer Vision*. Prentice Hall, 2001.

[4] R.C. Gonzalez and R.E. Woods. *Digital Image Processing*. Prentice Hall, 2002.

[5] X. Tai, C. Wu, F. Ren, and K, Kita, *Image Retrieval Based on Color and Texture*. Proceedings of the Fifth Mexican International Conference on Artificial Intelligence, 2006.

[6] M. Tkalcic and J.F. Tasic, *Color Spaces: perceptual, historical and applicational background EUROCON 2003*, Computer as a Tool, The IEEE Region 8, vol. 1, Sept. 2003, pp. 304-308.

[7] M.J. Swain and D.H. Ballard. *Color Indexing*. International Journal of Computer Vision, 7(1):11-32, 1991.

[8] M. Flickner, H.S. Sawhney, J. Ashley, Q. Huang, D. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steele, and P. Yanker. *Query by Image and Video Content: The QBIC System*. IEEE Computer, 28(9):23-32, September 1995.

[9] M. Stricker and M. Orengo. *Similarity of Color Images*. Proceedings SPIE, vol 2420, pp. 381-392, 1995.

[10] M. Stricker and A. Dimai. *Color Indexing with Weak Spatial Constraints*. Proceedings SPIE, vol 2670, pp. 29-40, 1996.

[11] T. Mostafa, H.M. Abbas, and A.A. Wahdan. *On the use of Hierarchical Color Moments for Image Indexing and Retrieval*. IEEE International Conference on Systems, Man and Cybernetics, 2002.

[12] J.L. Shih and L.H. Chen. *Color Image Retrieval Based on Primitives of Color Moments*. Vision, Image, and Signal Processing, IEEE Proceedings, vol. 149, no. 6, pp. 370-376, 2002.

[13] J. Huang, S.R. Kumar, M. Mitra, W.J. Zhu, and R. Zabih. *Image Indexing Using Color Correlograms*. Proceedings Computer Vision and Pattern Recognition, pp. 762-768, 1997.

[14] J. Huang, S.R. Kumar, and M. Mitra. *Combining Supervised Learning with Color Correlograms for Content-based Image Retrieval*. Proceedings of the Fifth International Conference on Multimedia, pp. 325-334, 1997.

[15] B. Zhang, C.I. Tomai, and A.Zhang. *An Adaptive Texture Image Retrieval System Using Wavelets*. Proceedings of the Seventh International Conference on Control, Automation, Robotics, and Vision, pp. 1210-1215, 2002.

[16] M. Kokare, P.K. Biswas, and B.N. Chatterji. *Texture Image Retrieval Using New Rotated Complex Wavelet Filters*. IEEE Transactions on Systems, Man, and Cybernetics, vol. 35, no. 6, pp. 1168-1178, 2005.

[17] H. Yu, M. Li, H.J. Zhang, J. Feng. *Color Texture Moments for Content-based Image Retrieval*. International Conference on Image Processing, pp. 24-28, 2002.

[18] D.A. Forsyth and M. Fleck. *Finding Naked People*. Proceedings European Conference on Computer Vision, pp. 593-602, 1996.

[19] J. Li, and J.Z. Wang. *Automatic Linguistic Indexing of Pictures by a Statistical Modeling Approach*. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 25, no. 9, pp. 1075-1088, 2003.

[20] A.D. Bimbo, P. Pala, and S. Santini. *Visual Image Retrieval by Elastic Deformation of Object Sketches*. Proceedings IEEE Symposium on Visual Languages, pp. 213-223, 1994.

[21] J. Kim and F. Pellacini. *Jigsaw Image Mosaics*. Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques, pp. 657-664, 2002.

[22] J. Zheng. *Automated Picture Montage Method and Apparatus*, US Patent and Trademark Office, http://www.uspto.gov, 2003.

[23] J.A. Hartigan and M.A. Wong. *A k-means Clustering Algorithm*, Applied Statistics, ch. 28, pp. 100-108, 1979.