THE DESIGN AND IMPLEMENTATION OF AN IMAGE SEGMENTATION SYSTEM FOR FOREST IMAGE ANALYSIS

By

Zhiling Long

A Thesis
Submitted to the Faculty of
Mississippi State University
in Partial Fulfillment of the Requirements
for the Degree of Master of Science
in Electrical Engineering
in the Department of Electrical and Computer Engineering

Mississippi State, Mississippi

August 2001

THE DESIGN AND IMPLEMENTATION OF AN IMAGE SEGMENTATION SYSTEM FOR FOREST IMAGE ANALYSIS

By

Zhiling Long

Approved:

Joseph Picone Professor of Electrical and Computer Engineering (Director of Thesis)	Robert J. Moorhead, II Professor of Electrical and Computer Engineering (Committee Member)
Nicolas H. Younan Associate Professor of Electrical and Computer Engineering (Committee Member and Graduate Director of Electrical Engineering)	A. Wayne Bennett Dean of the College of Engineering

Name: Zhiling Long

Date of Degree: August 4, 2001

Institution: Mississippi State University

Major Field: Electrical Engineering

Major Professor: Dr. Joseph Picone

Title of Study: THE DESIGN AND IMPLEMENTATION OF AN IMAGE SEGMENTATION SYSTEM FOR FOREST IMAGE ANALYSIS

Pages in Study: 76

Candidate for Degree of Master of Science

The United States Forest Service (USFS) is developing software systems to evaluate forest resources with respect to qualities such as scenic beauty and vegetation structure. Such evaluations usually involve a large amount of human labor. In this thesis, I will discuss the design and implementation of a digital image segmentation system, and how to apply it to analyze forest images so that automated forest resource evaluation can be achieved. The first major contribution of the thesis is the evaluation of various feature design schemes for segmenting forest images. The other major contribution of this thesis is the development of a pattern recognition-based image segmentation algorithm. The best system performance was a 61.4% block classification error rate, achieved by combining color histograms with entropy. This performance is better than that obtained by an "intelligent" guess based on prior knowledge about the categories under study, which is 68.0%.

DEDICATION

I would like to dedicate this thesis to my family.

ACKNOWLEDGMENTS

This work was supported by the United States Department of Agriculture (USDA) through the Ecosystem Management research for the Ouachita-Ozark National Forests at the United States Forest Service (USFS) Southern Research Station. The long association with Mr. Vic Rudis of the Southern Research Station has been enjoyable and educational.

Acknowledgement also goes to Dr. Joseph Picone, who directed the project throughout these years. His valuable suggestions were very important to the smooth progress of the project.

Many other people in the Institute for Signal and Information Processing (ISIP) helped me in the process of the project. In particular, I would like to thank Aravind Ganapathiraju, William C. Chapman, Richard J. Duncan, Suresh Balakrishnama, Neeraj Deshmukh, Jonathan Hamaker and Issac Alphonso for all their support. Without their assistance, this project could not have been done successfully. I should also thank Nirmala Kalidindi and Audrey Le for their wonderful work on the image database and the scenic beauty estimation (SBE) system, which laid a sound basis for this project.

Finally, I would like to thank Dr. Nicolas H. Younan and Dr. Robert J. Moorhead, II for serving on my committee. The time they spent reviewing this thesis is highly appreciated.

TABLE OF CONTENTS

CHAPTER Page

LIST OF TABLES

TABLE Page

LIST OF FIGURES

FIGURE Page

FIGURE Page

CHAPTER I

INTRODUCTION

An analog image can be described by a two-dimensional function $f(x, y)$, where (x, y) denotes the spatial coordinate and $f(x, y)$ gives a measurement value according to the coordinate. Analog images need to be digitized before they can be analyzed by a computer. The digitization consists of two steps. First, an analog image is sampled evenly in the spatial domain, with each sample represented by a picture element, or a pixel. Then, the measurement value of each pixel is quantized, converting the image into a digital format. As powerful computational resources have become significantly less expensive, the field of digital image processing has been evolving rapidly. In this thesis, I will discuss the design and implementation of a digital image segmentation system for forest image analysis.

1.1. Image Segmentation

To analyze the content of an image, one needs to first locate and isolate objects in it. This process, which is referred to as segmentation, is the first step in low-level vision [1]. Segmentation is a process of partitioning an image into some disjoint (non-overlapping) regions such that each region is homogeneous and no adjacent regions can be merged without violation of homogeneity [1]. Here, a region means a connected set

of pixels, that is, a set in which all pixels are adjacent. An example of a segmented image is shown in Figure 1.

A formal way to describe this image segmentation process is shown as follows: if F is the set of all pixels and $P(X)$ is a uniformity (homogeneity) predicate defined on X, which stands for a group of connected pixels, then the image segmentation is a partitioning of the set F into several connected subsets (or regions) S_i , $(i = 1, n)$, such

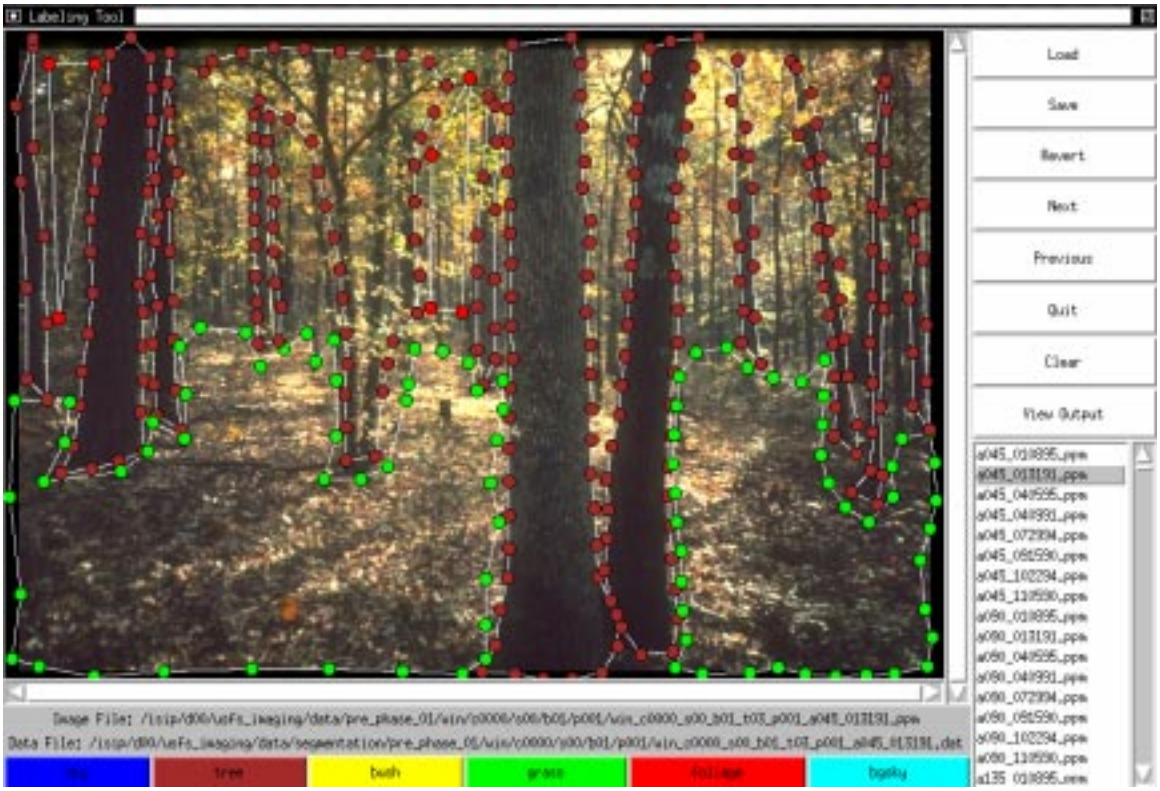


Figure 1. An example of a forest image which was manually segmented into various regions such as “grass” (green color) and “tree” (brown color). In this thesis, six categories of segments were defined for a forest image: tree, foliage, bush, grass, background sky, and sky. See Chapter V for details.

that $\bigcup_{i=1}^{n} S_i = F$ with $S_i \cap S_j = \varnothing, i \neq j$, and the uniformity predicate satisfies

$P(S_i) = true$ for all regions and $P(S_i \cup S_j) = false$ when S_i is adjacent to S_j.

1.2. Techniques for Image Segmentation

There are many segmentation techniques available in the literature [2]. None of these techniques are suitable for all types of images. However, most of the segmentation algorithms can be easily adapted to a variety of applications. Generally, the algorithms are based on such approaches as histogram thresholding [3], edge detection [3], and relaxation [4]. Here, I will give a brief introduction to the most popular techniques used for image segmentation.

Gray Level Thresholding

Thresholding is one of the oldest and simplest techniques for gray-level image segmentation. The basic idea is to assign pixels with a gray level falling between two thresholds to the same region. For the simplest case, if an image consists of regions with different gray level ranges (i.e., the regions are distinct), then the histogram of the pixel gray levels would show peaks and valleys, with each range between valleys corresponding to a certain region. In this case, if the gray levels at the valleys of the histogram are taken as thresholds, the image can be segmented into different regions. Here, the thresholding can be either bilevel (two regions with only one threshold) or of multilevel (multiple regions with several thresholds).

The key to thresholding techniques is the selection of threshold values. This is not a trivial job and various algorithms have been proposed. Ridler and Calvard proposed an iterative thresholding scheme which was based on Bayes' discrimination rule [5]. Ostu maximized a measure of class separability [6]. To overcome the possible drawbacks resulting from considering only the histogram information, Weszka and Rosenfeld developed a "busyness" measure which utilized spatial information, such as the co-occurrence of adjacent pixels in an image, to determine a set of thresholds [7]. Chanda also suggested an average contrast measure on the basis of the co-occurrence matrix [8]. On many occasions, when an image is noisy or the illumination is not good, fixed-value thresholding schemes are not capable of segmenting the image reliably. Adaptive thresholding methods such as the one proposed by Nakagawa and Rosenfeld [9] were developed to deal with such problems.

Iterative Pixel Classification

Approaches of this type can be subdivided into three classes: relaxation approaches, Markov random field-based (or MRF-based) approaches, and neural network-based approaches. Relaxation is an iterative algorithm in which the classification of each pixel can be done in parallel. Decisions made at neighboring pixels in the current iteration are then combined to make a decision in the next iteration. There are two types of relaxation approaches: probabilistic and fuzzy. In probabilistic relaxation, it is assumed that class assignments of pixels are interdependent. Therefore, when a pixel is classified, the category information of its neighboring pixels is also taken into account. The fuzzy

relaxation classification approach will be discussed later. This approach improved the probabilistic relaxation method by utilizing the fuzzy set theory.

MRF-based approaches use a spatial interaction model such as a Markov Random Field (MRF) or a Gibbs Random Field (GRF). Typical examples include the one proposed by Geman [10] which consists of a hierarchical stochastic model for the original image and a restoration algorithm for computing the maximum a posterior estimate of the original scene given a degraded realization. Another good example is the two-dimensional Bayes smoothing algorithm by Derin [11], which computes the optimum Bayes estimate for a scene value at each pixel. In the latter scheme, a scene is modeled as a special type of MRF, that is, the Markov Mesh Random Field which is characterized by causal transition distributions.

Neural networks are massively connected networks of elementary processors. The architecture and dynamics of such networks attempt to resemble information processing in biological neurons. Systems based on neural networks are likely to be robust with respect to random noise and failure of processors because of their massive connection architecture. Moreover, they are capable of generating outputs in real time due to the parallel processing mechanism. A large number of neural network-based image segmentation approaches have been developed in recent years. For example, Blanz and Gish used a three-layer feed forward network for segmentation [12]; Babaguchi developed a multilayer network trained with backpropagation for thresholding an image [13]; and Cortes proposed a neural network to detect potential edges in different orientations [14].

Segmentation Of Color Images

Color is a very important perceptual phenomenon related to human response to different wavelengths in the visible electromagnetic spectrum. There are various color models available for describing a color image. The most popular color models include the Red-Green-Blue (RGB) model, the Hue-Intensity-Saturation (HIS) model, and the Cyan-Magenta-Yellow-Black (CMYK) model.

Some segmentation algorithms attempt to exploit color using the models mentioned above. For example, Ohta et al. [15] attempted to find a set of effective color features by systematic experiments in region segmentation. They claimed that the following three color features, namely $I_1 = (R + G + B)/3$, $I_2 = (R - B)/2$ and $I_3 = (2G - R - B)/4$, constitute the most effective set of features.

Another class of techniques for color image segmentation utilize spectral analysis. These algorithms usually require prior knowledge of the colors of objects. In cases where such prior information is not available, clustering techniques are used. An example typical of this approach is the two-stage color image segmentation which combines thresholding with the Fuzzy C-Means (FCM) method [16].

Edge Detection

Segmentation can also be obtained through detection of edges, defined as narrow regions where there is an abrupt change in gray level intensity. Edges are local features that are computed using pixels in the immediate proximity of the region under

consideration. Generally, edge detection algorithms can be classified into two categories: sequential and parallel. For sequential techniques, the decision whether a pixel is an edge pixel or not is dependent on the detection results at some previously examined pixels. Therefore, the performance of a sequential edge detection method is dependent on the choice of an appropriate starting point and on how the results of previous points influence the selection and result of the next point.

For parallel methods the decision is made based on the point under consideration and its neighboring pixels. For these parallel algorithms usually there are different types of differential operators involved, such as the Roberts gradient [1], the Sobel gradient [1], the Prewitt gradient [1] and the Laplacian operator [4]. These operators are able to detect changes in the gray level of a given image.

A good edge detector should be a filter with the following two features. First, it should be a differential operator, taking either a first or second spatial derivative of images. Second, it should be capable of being tuned to act at any desired scale, so that both blurry edges and sharply focused fine details can be detected.

Methods Based On Fuzzy Set Theory

Zadeh introduced the concept of fuzzy sets in which imprecise knowledge can be used to define an event. The relevance of fuzzy set theory to pattern recognition problems has been adequately addressed in the literature [17][18]. In general, fuzzy set theory may be incorporated to handle uncertainties resulting from deficiencies in various stages of a pattern recognition system. In particular, fuzzy image processing and recognition is under rapid development. The reason for the increasing interest in fuzzy image processing

mainly lies in the realization that many of the basic concepts in image analysis, such as that of an edge or a boundary, are ambiguous in nature and cannot be defined precisely. Fuzzy set theory appears to be a good approach to deal with such uncertainty issues involved in image processing and recognition.

Image segmentation algorithms that exploit fuzzy set theory can be divided into three categories: thresholding, clustering and edge detection. Each of these classes are similar to their non-fuzzy counterparts, with the only difference being that key parameters (such as thresholds) are determined with uncertainty taken into consideration.

1.3. Applications to Forest Image Analysis

Image segmentation is essential to many image analysis applications. In this thesis, I will study how to apply image segmentation to analyze forest images so that automated forest resource evaluation can be achieved.

The United States Forest Service (USFS) is developing software systems to evaluate forest resources with respect to qualities such as scenic beauty and vegetations structure. Such evaluations usually involve a large amount of human labor and are very tedious. However, with the assistance of digital image analysis and pattern recognition techniques, one will be capable of extracting features and creating statistical models which represent scenic beauty and vegetation structure, thus making the evaluation process automatic.

Reliable analysis of forest images would be straightforward if typical forest structures (objects) could be isolated from an image. In this manner, one may quantify various qualities of an image with respect to each category, and then further analyze the

structures for other aspects in which people are interested. This task requires an automatic image segmentation system.

Initial interest in forestry imaging focused on the development of an automatic algorithm to estimate the scenic beauty rating of forest images [19]. Federal and regional agencies have stressed the need for aesthetic evaluation to plan timber harvests in national forests and to assess the impact of public incentive programs for forest management. Scenic beauty estimation (SBE) is a quantitative measure of an individual's preference for the visual attributes of an image. Traditional methods used for SBE include the Public Preference Model [20] and the Descriptive Inventories [21], which involve a large amount of human labor. Thus the initial goal of this project was to extract relevant features from forest images using signal processing techniques for automatic scenic beauty prediction.

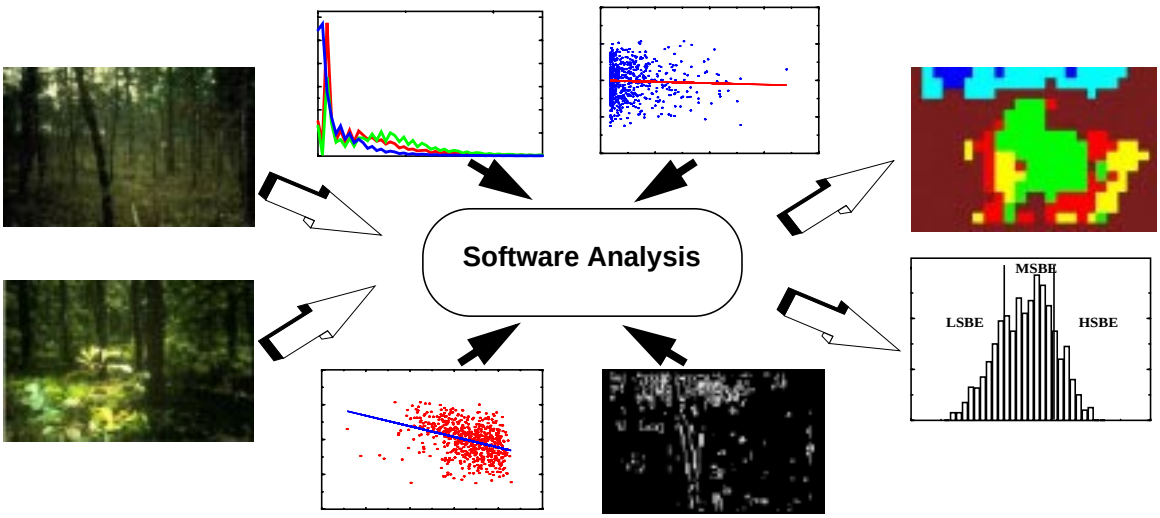


Figure 2. A diagram demonstrating how digital image processing techniques can be used for forestry resources analysis and management.

To assess progress on this task, an extensive database [22] had been developed in conjunction with the USFS. The images included in this database were drawn from a study spanning four years in the Ouachita-Ozark National Forests in Arkansas, USA. Photographs taken under controlled conditions were digitized using an extremely high quality scanning process and converted into computer readable data. The database consists of 637 images, which were taken over two different sessions (1990-91 and 1994-95) and sampled over all the seasons of the year at a number of different angles. Subjective scenic beauty ratings are available for each of the images in the database. Later, an additional 1147 images had been added, bringing the total number of images to 1784. This well-developed database is large enough to provide sufficient data for algorithm training and testing. It is indispensable for the evaluation of system performance.

The scenic beauty of an image is highly dependent on the complexity of that image. Complexity is defined as the degree of variation in the visual qualities of an image [21]. Algorithms had been developed to extract diverse features from the forest image database as measures of image complexity. These features include color histograms with the RGB color model, density of long lines and short lines, the sharpness, the standard deviation of pixel intensities, etc. The features were combined into a variety of feature sets and SBE-based image classification was achieved with them. Regression analysis was also applied to relate the extracted features to the scenic beauty. Examined feature combinations include individual colors (red, green, and blue), colors combined with the density of long and short lines, and colors combined with information theoretic measures such as entropy [19].

An image analysis system had been developed which classifies images into three categories (low, medium, and high) according to their scenic beauty qualities. A system using color, long lines, and entropy [19] yielded the lowest overall classification error rate — 38.5%. This combination of features also had a high correlation with the reference SBEs — 0.59. Most of the errors were observed to be in the regions of overlap between images rated as having medium to low scenic beauty and medium to high scenic beauty. This observation suggests that the SBE value as determined by human perception is somewhat imprecise and needs a more fundamental understanding and calibration. The results also indicate that color and density of the trees play a major role in estimating the scenic beauty of forest images.

My evaluation of image segmentation approaches has been carried out using the above-mentioned research as a testbed. I have augmented the system with a capability to detect and quantify key structures within the image, such as trees, sky, and bushes. I believe that developing an understanding of the composition of the scene will be crucial to improving the ability to analyze images.

1.4. Contribution of the Thesis

The major contributions of this thesis can be summarized as follows. First, I investigated optimization of several feature extraction schemes. Second, I designed features based on the colors yellow and brown, features based on the discrete cosine transform, and explored a feature design based on a Gabor filter bank. These feature extraction schemes were evaluated and their effectiveness in analyzing forest scenery was

verified. Third, I developed an image segmentation algorithm based on block classification using principal component analysis (PCA). A system built on this algorithm achieved an error rate of 61.4% for block classification, using the combination of color histograms and entropy as the feature set. Finally, I investigated possible impact of the segmentation system on the scenic beauty estimation application. A reduction of SBE classification errors for some categories of segments was observed.

1.5. Structure of the Thesis

The thesis is structured as follows. In chapter II, I discuss feature extraction schemes. In chapter III, I will introduce algorithms to classify image segments using a pattern recognition paradigm. Chapter IV contains software design and implementation issues. I will present the experimental results and the corresponding analysis in chapter V. In chapter VI, I will discuss the impact of this image segmentation system on scenic beauty estimation. Finally, in chapter VII, I present conclusions and discuss promising future directions.

CHAPTER II

FEATURE EXTRACTION

To identify a specific target, one always needs features characteristic of that object. In a pattern recognition problem, such features are represented by a multi-dimensional vector called a “feature vector.” Obviously, the separability of feature vectors representing different objects is crucial to achieving a low error rate in pattern recognition. Previously, a variety of features had already been developed for image classification based on their ability to predict scenic beauty. In this work, I developed additional features better suited for image segmentation.

2.1. Baseline Features

In previous work [19], a variety of features had been developed for SBE classification. These features include color histograms, density of lines, entropy, standard deviation, sharpness, fractal dimension and compression ratio. The SBE classification results [19] proved that these features were representative of characteristics of forest images. Therefore, they are still used in the image segmentation system.

Color Histograms

Color is the most visually striking feature of any image and it has a significant bearing on the scenic beauty of an image. For forest image analysis, histograms of color intensities were generated and used as features.

Images in the database use the Portable Pixel Map (PPM) format [23], which assumes the Red-Green-Blue (RGB) color model. The PPM format represents each pixel using 24 bits — one 8 bit byte for each color. Therefore, the minimum intensity value of each color is 0 and the maximum value is 255. To compute the histogram of a color, this range is evenly divided into 10 bins (to generate a reasonably-sized feature vector), with the center value of the first bin being 0 and the center value of the tenth bin being 255. Then distribution of the intensity values is computed with regard to these bins. This computation generates histograms for the three primary colors, respectively. For each histogram, there are 10 feature values, each of which stands for the count of pixels whose intensity value for that specific color falls into the given bin. Sample color histograms extracted from typical forest images are shown in Figure 3.

Line Features

The density of long lines (representative of trees) in an image is a good indication of the density of trees in the image. This density property is not only important in forest resources management (e.g., decision on cutting trees for lumber use), but also a suitable measure of visual penetration, which partly determines the scenic beauty quality of an image. Edge and line detectors were used to locate these line features.

The Canny algorithm [24] was chosen for edge detection because it achieves a minimum localization error with an error rate comparable to other algorithms such as Sobel [1] edge detection and Roberts [1] edge detection algorithms. A block diagram for the Canny edge detector is shown in Figure 4.

To detect lines, the output of the edge detector is postprocessed and edge lengths are compared with a threshold parameter [25]. For an edge to be a line, its length must be above the line threshold. The distinction between short bushes and tall trees is made by assigning another threshold, i.e., the "long line threshold." A detected line whose length falls below the long line threshold is a short line. Since only those vertical lines, which represent tree stems, shrubs and grasses, are of interest in this application, lines in the horizontal direction are not detected. The percentage values of long lines and short lines to the total number of detected lines are used as line features.

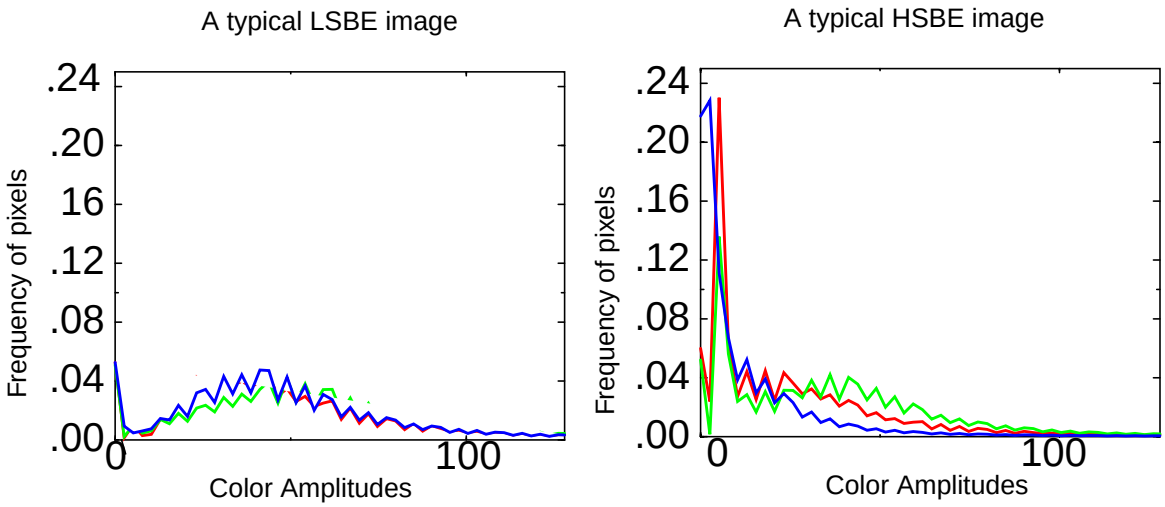


Figure 3. Sample color histograms extracted from typical forest images. The plot on the left is from a Low Scenic Beauty Estimate (LSBE) image, and the one on the right is from a High Scenic Beauty Estimate (HSBE) image.

Sharpness

Sharpness is used to represent the local variation of pixel intensities. It is computed as a frequency-weighted sum of the magnitudes of difference in pixel intensities of an image. Mathematically, it is defined as

$$Sharpness = \sum_{m,\,n} \left(\sum_{i=-1}^{1} \sum_{j=-1}^{1} |x(m,n) - x(m-i,n-j)| \times f \right) \quad . \quad (1)$$

Here, $i,\ j \neq 0$ and f is the cumulative frequency of the bin in the intensity histogram.

The histogram of the difference in pixel intensities gives a measure of amplitude variations in an image and hence a measure of the sharpness of the image. This sharpness

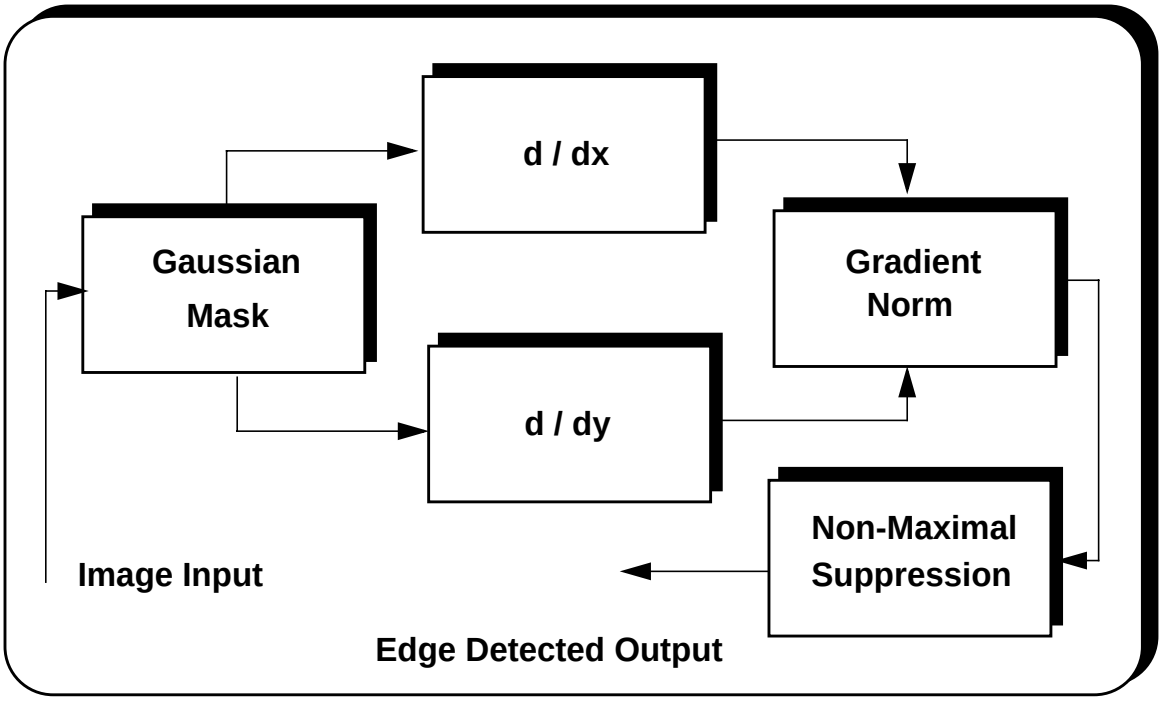


Figure 4. The block diagram for the Canny edge detection algorithm.

is computed for each of the three primary colors in the image. Two example images with significantly different sharpness values are shown in Figure 5.

Standard Deviation

While sharpness is a measure of local variation of pixel intensities, standard deviation is a measure of the global or overall variation of pixel intensities in an image. The standard deviation is computed as

$$STD = \sqrt{\frac{\sum_{i}^{N} (x_i - \bar{x})^2}{N-1}} \quad . \tag{2}$$

Here x_i is the intensity value of a pixel in a primary color channel and $\bar{x}$ is the mean intensity of that primary color in the image. The standard deviation values for each of the

Sharpness = 352.29, SBE = -122.31

Sharpness = 102.91, SBE = 84.63

Figure 5. Example images with different sharpness features. Note that the SBE is a signed real number. The greater the value is, the better the scenic beauty quality is.

three primary colors are computed individually and summed to obtain the total standard deviation.

Example images with different standard deviation values from the database are shown in Figure 6. Typically an image which is bright due to penetrating sunlight has a higher standard deviation. On the other hand, an image which has less sunlight penetration usually has a lower standard deviation.

Entropy

Entropy is a measure of randomness in an image. It is represented mathematically as

$$Entropy = -\sum_{x} p(x)\log p(x) \ . \qquad (3)$$

Here x is the serial number of a bin within a color histogram and $p(x)$ is the probability for a pixel intensity value to fall into that bin. As described earlier, the dynamic range of intensity values for each color is divided into 10 bins. For each bin, the pixel intensity

Standard Deviation = 33.47, SBE = -4.61

Standard Deviation = 2.40, SBE = 54.07

Figure 6. Example images with different standard deviation values.

distribution is calculated. Then entropy is computed according to the above equation. The total entropy is computed as the sum of the three entropy values for the primary colors.

Example images demonstrating a variety of entropy values are shown in Figure 7. Typically, an image with more randomness has a lower scenic beauty quality. This is demonstrated in the scatter plot shown in Figure 8.

Compression Ratio

The compression ratio of an image is another measure of the complexity of the image. Image compression is a technique which seeks to replace original pixel-related information with more compact mathematical representations. The compression ratio is the ratio of the original image size to the compressed image size. Images with high complexity are less susceptible to compression, hence are associated with a low compression ratio. Here, compression ratio of an image is computed with JPEG coding [26], which is a widely used lossy compression technique. Sample images with high and low compression ratios are shown in Figure 9.

Fractal Dimensions

Fractal geometry is used to describe, model and analyze the complex forms of nature and fractal dimension is a measure of the texture of an image. A Triangular Prism Surface [27] approach was used to compute the fractal dimension, which is illustrated graphically in Figure 10. First, a square region $ABCD$ is chosen and divided into four triangles with the center pixel P as the common vertex for all the four triangles. The distance r is variable, with a minimum value being 3 pixels. The sum of the areas of all

Entropy = 22.57, SBE = -122.31 Entropy = 15.77, SBE = -47.20

Figure 7. Example images with different entropy values.

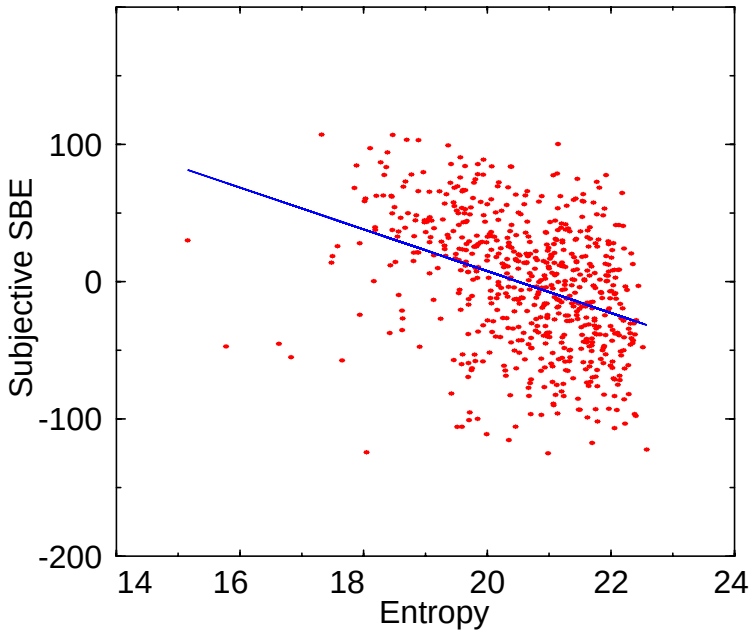


Figure 8. A scatter plot of entropy vs. SBE. The blue line was generated by a linear regression on the data.

Compression ratio = 7.77, SBE = -122.31 Compression ratio = 3.81, SBE = 84.63

Figure 9. Example images with high and low compression ratios.

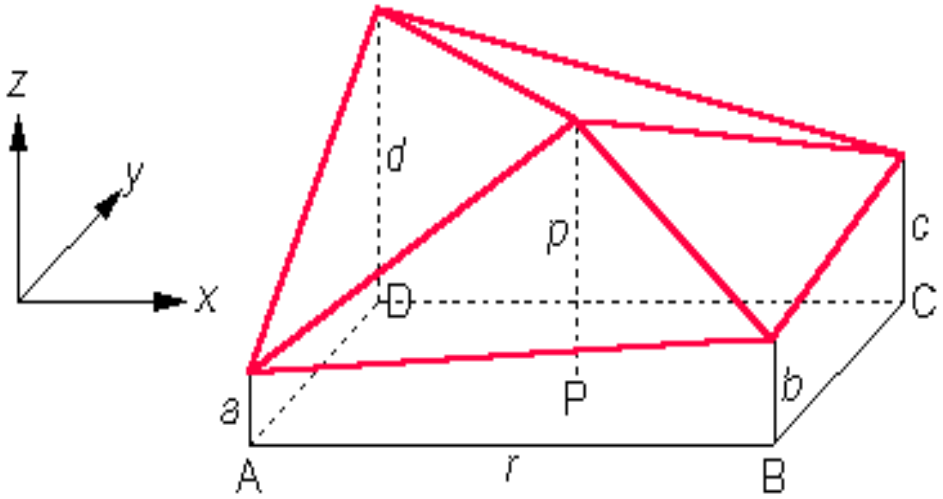


Figure 10. The triangular prism surface approach.

the triangles is computed. This procedure is repeated for each pixel in the image and the total sum of areas is acquired. Subsequently, different values of r are used for this area computation and an area-distance plot is made in log scale, as shown in Figure 11. The slope of the line is denoted as s. Then the fractal dimension is related to s as $D = 2 - s$. For color images, fractal dimensions for each color are computed separately.

Images with higher fractal dimensions are considered to be more complex than images with lower fractal dimensions. Sample images with high and low fractal dimensions are shown in Figure 12.

2.2. Development of New Color Features

Colors have an important effect on human perception of the scenic beauty of an image. Rudis and others [28] noted that blue was significantly associated with the degree of recent cutting. In [29] and [30] the following rules have been demonstrated:

- the color green is most likely to appear in a summer scene, yellow in fall, but blue and brown have the highest chance to show in winter.
- in non-winter seasons, the presence of green and yellow colors enhances the scenic beauty of a forest scene, while blue and brown do the opposite.

From these results, it is evident that brown and yellow are also key color features in evaluating the scenic beauty quality of an image. Moreover, brown distinguishes tree stems from other forest scenes, and yellow helps to identify leaves and grasses in a forest image taken in the fall. Therefore, it is necessary to add the brown and yellow features into

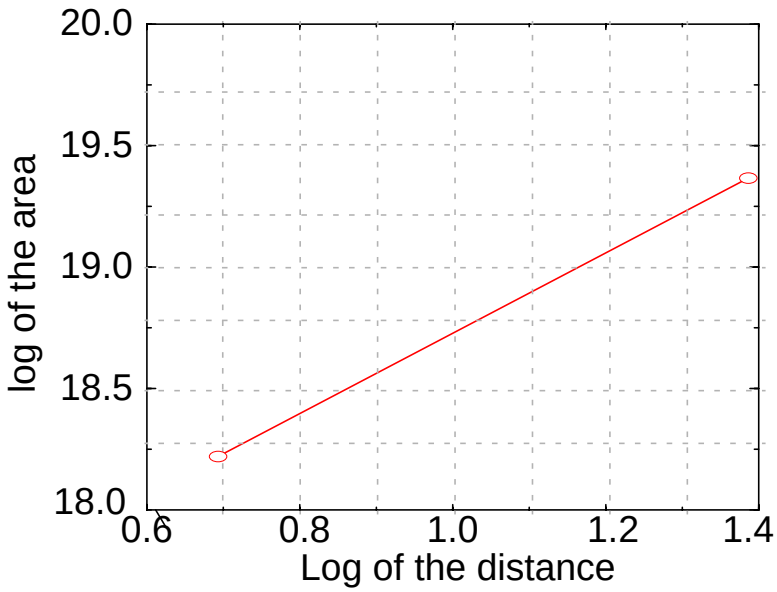
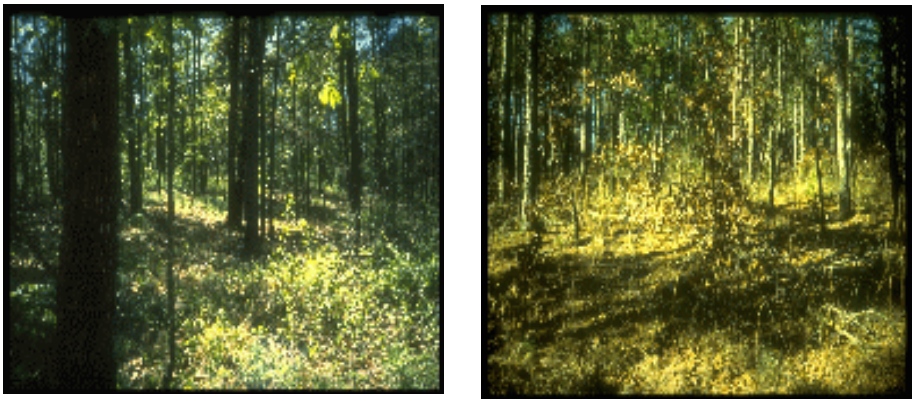


Figure 11. The area-distance plot. Note both the distance and the area are in log scale.



Fractal Dimension = 2.32, SBE = 31.20 Fractal Dimension = 0.14, SBE = -85.57

Figure 12. Example images with high and low fractal dimensions.

the color measurement scheme and evaluate their effectiveness in analyzing forest images for various purposes.

The histograms for the brown and yellow colors were computed as feature vectors, as they were for red, green and blue, previously. To calculate the intensity of brown and yellow for a given pixel, one takes advantage of the theory of additive color synthesis [31], which makes it possible to create any color by mixing the three primary colors, red, green, and blue, in various proportions.

Particularly, the color brown is generated by mixing one part blue, one part green, and four parts red. Similarly, yellow can be obtained by mixing equal parts of red and green. A normalized version of the mixing formulae were implemented as:

$$brown = \frac{1}{6} \times blue + \frac{1}{6} \times green + \frac{4}{6} \times red \quad (4)$$

$$yellow = \frac{1}{2} \times red + \frac{1}{2} \times green \quad (5)$$

Here red, green and blue stand for the corresponding intensity values of the given pixel.

2.3. Development of DCT-based Frequency Features

So far, none of the features discussed presents spectral information of forest images. Given the fact that most forest structures display remarkable variations in the spatial domain, and that the patterns of those variations change with each specific kind of structure, I believe that there should exist some frequency features helpful for distinguishing between those structures. Therefore, I investigated widely-applied frequency domain analysis techniques such as the Discrete Cosine Transform (DCT) [26] and the Gabor filter bank [32].

DCT Analysis

The DCT is a widely used frequency analysis method in image processing [33][34][35]. It has many advantages over other transform techniques. For example, no complex computation is involved in the DCT; fast DCT implementations are available; and most importantly, energy is packed efficiently into a small number of DCT coefficients [26].

A two-dimensional forward DCT of an $n \times n$ data block is defined as

$$F(u,v) = \frac{4C(u)C(v)}{n^2}\sum_{j=0}^{n-1}\sum_{k=0}^{n-1} f(j,k)\cos\left[\frac{(2j+1)u\pi}{2n}\right]\cos\left[\frac{(2k+1)v\pi}{2n}\right] \quad (6)$$

Here the constants are computed as

$$C(w) = \begin{cases} \frac{1}{\sqrt{2}} & w = 0 \\ 1 & otherwise \end{cases}.$$

On the other hand, the inverse 2-D DCT (IDCT) is defined as

$$f(j,k) = \sum_{u=0}^{n-1}\sum_{v=0}^{n-1} C(u)C(v)F(u,v)\cos\left[\frac{(2j+1)u\pi}{2n}\right]\cos\left[\frac{(2k+1)v\pi}{2n}\right] \quad (7)$$

Here $C(w)$ is calculated the same way as it is in the forward DCT.

To better understand the idea of DCT analysis, a forward DCT was first computed on the forest image data. One typical region was chosen from the image database for each of the six categories under study, i.e., tree, foliage, bush, grass, background sky and sky (see Chapter V for definitions). Then the DCT coefficients were generated for the

selected data with the block size ranging from 8 to 128. There appeared to be a unique pattern of the coefficients' amplitude distribution for each kind of structure. To illustrate this more intuitively, all the original data and the outcome DCT coefficients were plotted in MATLAB. Some example plots are shown in Figure 13 and 14. The plots shown in Figure 13 describe amplitude distributions of the DCT coefficients calculated on a typical tree region and a foliage region with the image block size set to 64 x 64, while the plots in Figure 14 show the DCT coefficients computed on a foliage region with the block size set to 8 x 8 and 32 x 32, respectively.

From these plots, it is noted that the DCT coefficients' amplitude distribution varies significantly between the structures. This variation may indicate a potential for DCT coefficients to distinguish between the forest structures. Another observation is that the block size for the DCT computation has a noticeable impact on the pattern of the amplitude distribution. The reason for this sensitivity to the block size may be that small sizes are not sufficient for the data to contain enough texture information, resulting in patterns inconsistent with those generated on larger blocks. Also, an interesting phenomenon can be observed that whatever the block size is, the energy tends to be packed into the lower frequency components by the transformation. Therefore, the lower frequency components are much more important than the higher components, since they collect the majority of the energy. Based on these observations, I believe that if a suitable block size is chosen and the lower DCT coefficients are used to generate features, I would be able to achieve good segmentation performance.

DCT-based Feature Design

I designed a few features based on what had been observed in the DCT analysis. The basic idea behind the design is: first compute DCT coefficients on an image block, then filter them in the frequency domain. After that choose the lower frequency components as the feature vector. To be more specific, I designed the baseline scheme in the following way: perform a DCT on the green pixels, average every four of them for the filtering stage, then choose the first 16 filtered outputs to create the feature vector.

With this scheme, a feature vector is built from the first 64 DCT coefficients. This choice is reasonable since it has already been verified that the energy is packed into the

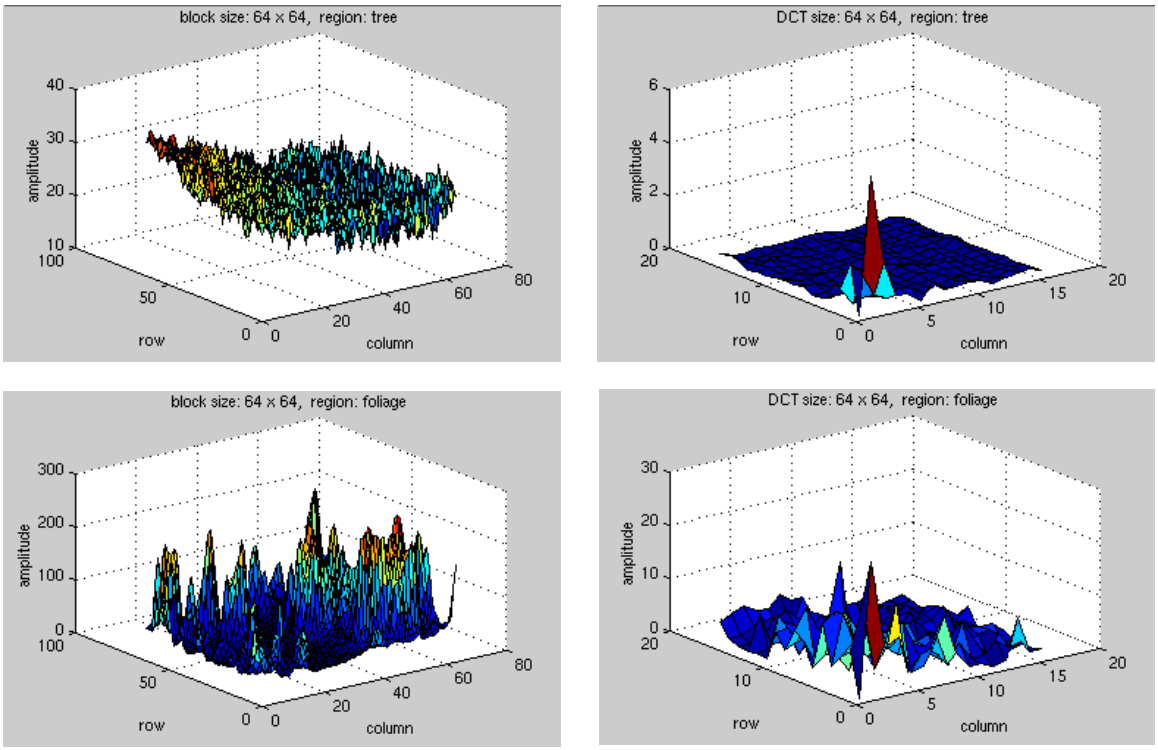


Figure 13. The amplitude distribution plots of DCT coefficients computed on some typical forest structures (trees on the top; foliage on the bottom). Note the left column plots are for the original data, and the right ones are for the DCT coefficients.

lower frequency components, and for those typical regions, the first 64 DCT coefficients contain the majority of the energy. If all DCT coefficients are included in the feature computation, there would be problems with large DCT block sizes. For example, computing a DCT on a 64 x 64 block and averaging every four of the DCT coefficients to generate a feature vector produces a feature vector with a dimension of 64 x 64 / 4 = 1024. This is too large to be practical for most applications and would result in an unnecessarily large amount of computations in the overall system.

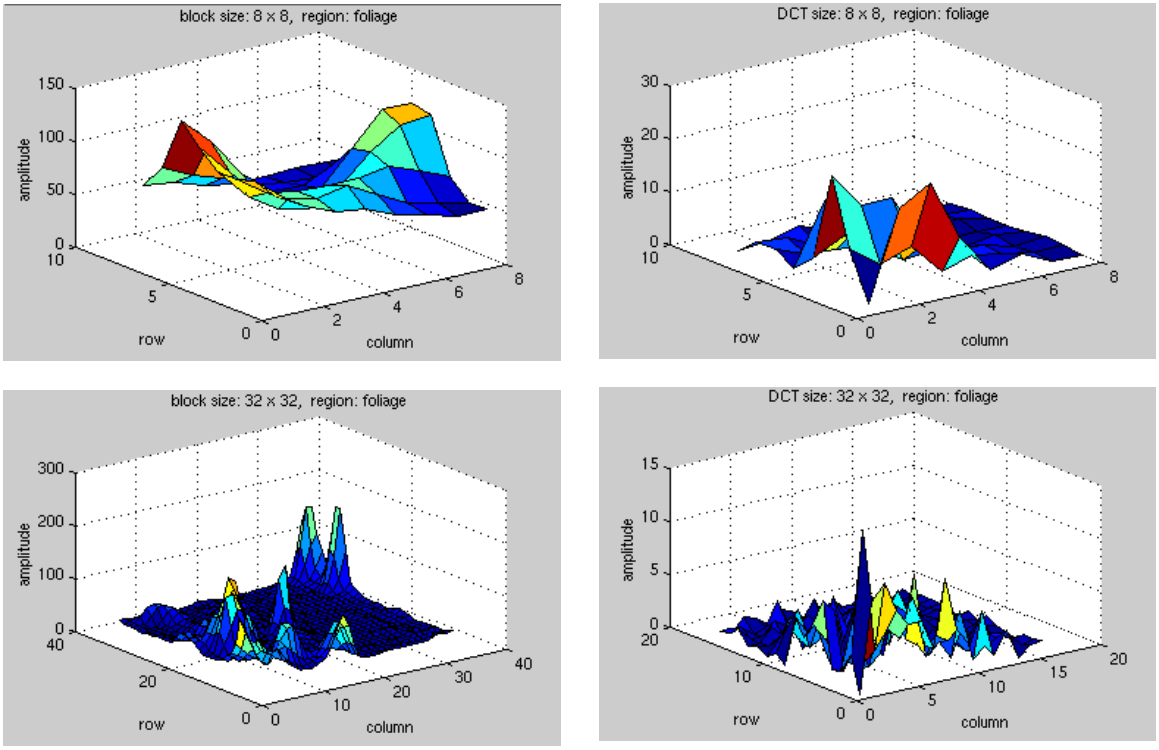


Figure 14. The amplitude distribution plots of DCT coefficients computed on a typical foliage region with the block size set to 8 x 8 and 32 x 32, respectively.

2.4. Gabor Filter Bank-based Features

Another frequency analysis technique I investigated was Gabor filters. This filtering technique was justified based on a physiological study of the human vision system [32]. An unsupervised texture image segmentation algorithm based on this technique was developed by Jain and Farrokhnia and proved very effective in classification of a small number of texture categories [32]. First, I duplicated experiments described in the original paper and verified the performance of this algorithm. Next, I applied this algorithm to images representing different forest structures of interest in the study. I analyzed the results and identified problems with this feature design for the forestry application.

Gabor Filters

Gabor filters are important in visual image analysis. The impulse response of an even-symmetric Gabor filter is given by:

$$h(x, y) = \exp\left\{-\frac{1}{2}\left[\frac{x^2}{\sigma_x^2} + \frac{y^2}{\sigma_y^2}\right]\right\}\cos(2\pi u_0 x). \qquad (8)$$

Here, u_0 is the frequency of a sinusoidal plane wave along the x axis (or the 0^o orientation), σ_x is the space constant of the Gaussian envelope along the x axis, and σ_y is the other space constant along the y axis. To obtain a Gabor filter with an arbitrary orientation, one needs to rotate the $x - y$ coordinate system accordingly.

The Fourier domain representation of (8) is given by

$$H(u, v) = A\exp\left\{-\frac{1}{2}\left[\frac{(u-u_0)^2}{\sigma_u^2} + \frac{v^2}{\sigma_v^2}\right]\right\} + A\exp\left\{-\frac{1}{2}\left[\frac{(u+u_0)^2}{\sigma_u^2} + \frac{v^2}{\sigma_v^2}\right]\right\}. \qquad (9)$$

Here $\sigma_u = \frac{1}{2}\pi\sigma_x$, $\sigma_v = \frac{1}{2}\pi\sigma_y$, and $A = 2\pi\sigma_x\sigma_y$. This equation is also referred to as a *modulation transfer function* (MTF) since it specifies the amount by which the filter modifies each frequency component of the input image.

Gabor filters are able to keep an optimal balance between both the resolution in the spatial domain and that in the frequency domain [32]. This is a significant property for the texture segmentation problem, where high resolution in the spatial domain is necessary for locating texture boundaries, and high resolution in the frequency domain is desirable for distinguishing between different kinds of textures. The usefulness of Gabor filters in texture segmentation has been demonstrated by the research work described in [32].

Gabor Feature Design

The Gabor features were extracted on the basis of the design described in [32]. First, a bank of Gabor filters are convolved with an input texture image. For these Gabor filters, the orientation θ is set to be 0°, 45°, 90°, and 135°, respectively, and the following values are used for the radial frequency u_0: $1\sqrt{2}$, $2\sqrt{2}$, $4\sqrt{2}$, ..., and $(N_c/4)\sqrt{2}$. Here N_c is the number of columns of the input image. This set of Gabor filters are sufficient for detecting textures with either high or low frequency components, of any arbitrary orientation.

Second, the filtered outputs, denoted as $r_k(x, y)$, are transformed nonlinearly using the equation:

$$\psi(t) = \tanh(\alpha t) = \frac{1 - e^{-2\alpha t}}{1 + e^{-2\alpha t}}. \qquad (10)$$

The curve for this function is shown in Figure 15. From the curve, one can see that the function actually represents a rapidly saturating, threshold-like transformation. The constant in the equation, α, determines how fast the function saturates.

After the nonlinear transformation, feature images are generated by averaging over windowed regions centered at each pixel in the transformed images. The feature computation for a filtered image $r_k(x, y)$ is given as

$$e_k(x, y) = \frac{1}{M^2} \sum_{(a, b) \in W_{xy}} \left|\psi(r_k(a, b))\right|. \tag{11}$$

Here W_{xy} is an $M \times M$ window centered at the position of (x, y). Actually, the nonlinear transformation is incorporated into this equation.

In equation (11), the features are calculated with rectangular windows. To keep a balance between both a reliable measurement of texture features and an accurate localization of region boundaries, a Gaussian window, instead of the rectangular one, has been adopted in Jain's algorithm. Therefore, equation (11) is modified to:

$$e_k(x, y) = \frac{1}{M^2} \sum_{(a, b) \in W_{xy}} g(a - x, b - y)\left|\psi(r_k(a, b))\right|. \tag{12}$$

Here,

$$g(x, y) = \exp\left\{-\frac{1}{2} \cdot \frac{x^2 + y^2}{\sigma^2}\right\}. \tag{13}$$

The constant σ in (13) is proportional to the average size of the intensity variations in the image, which is denoted as T. That is, $\sigma = 0.5T$, where $T = N_c / u_0$. The scale was empirically determined to be 0.5. The window size M is supposed to be

positively proportional to the value of σ, and the optimal scale also needs to be determined empirically.

When feature computation is performed for all filtered images, feature values of the same position are assembled into a feature vector. This feature vector is then used for classifying the pixel at that position. Evaluation details about this feature design are described in Chapter V.

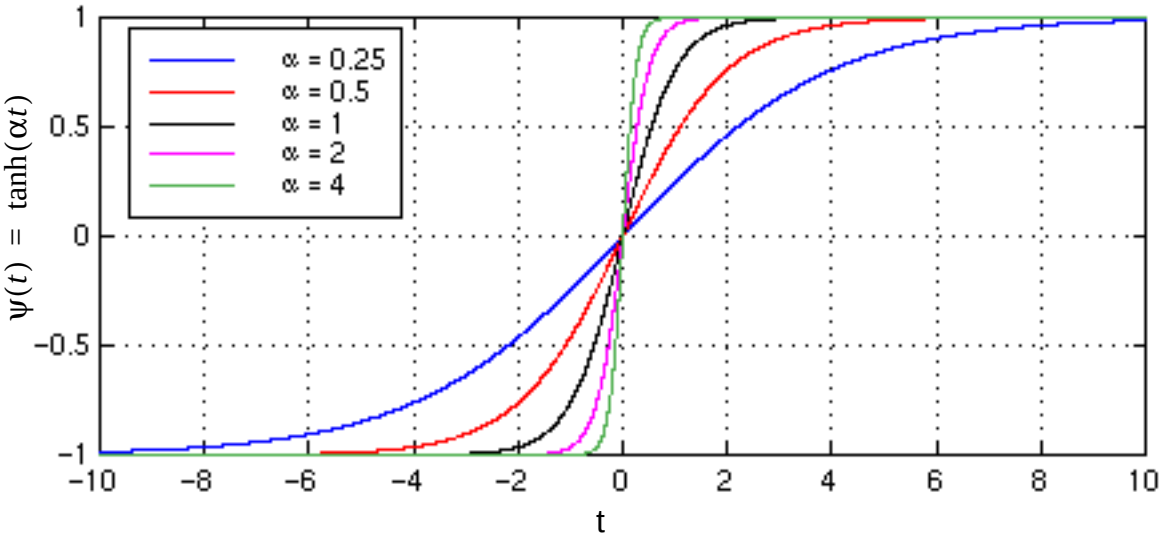


Figure 15. The nonlinear transformation used in the Gabor filter approach.

CHAPTER III

ALGORITHMS

The core of the block-level classification scheme for the image segmentation system is a well-established statistical pattern recognition algorithm known as Principal Component Analysis (PCA) [36]. Fundamentals of PCA as well as the details of the algorithm design are described in this chapter.

3.1. Principal Component Analysis

Principal Component Analysis (PCA) is a statistical normalization technique. It is probably the most straightforward transformation that can be used with a linear classifier [36]. PCA seeks a projection that represents a set of data with the least square error.

Suppose one has a set of *d*-dimensional samples $\vec{x}_1$, $\vec{x}_2$,..., $\vec{x}_n$, of which the sample mean is $\vec{\mu}$ and the covariance matrix is Σ. One may represent a vector in the sample space as

$$\vec{x} = \vec{\mu} + \sum_{i=1}^{d} a_i \cdot \vec{e}_i \quad . \tag{14}$$

The corresponding squared-error criterion function is

$$J = \sum_{k=1}^{n} \left\| \left(\hat{\mu} + \sum_{i=1}^{d} a_{ki} \cdot \hat{e}_i \right) - \hat{x}_k \right\|^2 \quad . \tag{15}$$

As shown in [36], to minimize the criterion function, the vectors $\hat{e}_1$, $\hat{e}_2$,..., $\hat{e}_d$ must be the eigenvectors associated with the largest eigenvalues of a scatter matrix S, where

$$S = \sum_{k=1}^{n} (\hat{x}_k - \hat{\mu}) \cdot (\hat{x}_k - \hat{\mu})^t = (n-1) \cdot \Sigma \quad . \tag{16}$$

Given that S is real and symmetric, the eigenvectors $\hat{e}_1$, $\hat{e}_2$, ..., $\hat{e}_d$ are orthogonal. Therefore, they consist of a set of basis vectors for representing the space formed by the sample vectors. If the eigenvectors are combined to form a transformation matrix Φ with each vector as one of its column vectors, then a linear transformation is obtained:

$$\hat{y} = \Phi^T \hat{x} \quad , \tag{17}$$

which transforms a vector into the specific sample space.

3.2. Class-specific PCA

To apply principal component analysis to a practical pattern recognition problem, one may choose to do a class-specific PCA. Suppose one needs to distinguish between n categories (classes). First of all, PCA is applied to each class respectively. That is, a group of training data (feature vectors) is collected for each class, then mean vectors and covariance matrices are computed within each class. This process (also known as supervised learning) builds up statistical models for all the classes.

To determine the category a given feature vector belongs to, the feature vector is linearly transformed, as shown in equation (17), into feature spaces corresponding to each category, and is classified by using a nearest-neighbor rule [37]. For the nearest-neighbor estimation, Euclidean distance is used, which can be computed directly using the following equation,

$$d^2(\vec{x}) = (\vec{x} - \vec{\mu})^T \Sigma^{-1} (\vec{x} - \vec{\mu}) \quad . \tag{18}$$

Here $\vec{x}$ is the vector to be classified, $\vec{\mu}$ and Σ are the mean vector and covariance matrix of a certain class.

Obviously, the choice of the training data set is vital. To be specific, the amount of the training data should be sufficient, and the chosen data should be typical, that is, they should be capable of representing the nature of the associated categories.

3.3. Block-level Classification

The objective of this thesis is to build up an image segmentation system, which is capable of segmenting a given forest scene into its various constituents. To simplify the problem, six categories of forest structures were defined. These categories are trees, bushes, grasses, foliage, sky, and background sky. They are sufficient to represent typical forest scenes dealt with in this application. The definitions of these categories will be given in Chapter V, where the data preparation and experimentation will be described.

To implement such a segmentation system, I proposed a block-level classification scheme. According to this scheme, an image is divided evenly into small blocks. Then it is processed block by block. For each block, a pattern recognition algorithm is applied to

classify the block into one of the predefined categories. After all blocks have been classified, a segmentation of the image is obtained, in which each segment is assigned to a category. I used class-specific principal component analysis for block classification, which had been successfully employed in the previous image classification systems [19].

The block-level classification scheme is simple and efficient with regard to the forestry application. Classifying groups of pixels is much more efficient than repeating the process for each pixel, reducing the computational complexity significantly.

On the other hand, classification on the basis of blocks also results in some disadvantages. The most significant one is that the segmentation would not be as accurate as those accomplished by pixel-based algorithms. Errors occur at the borders between adjacent regions belonging to different categories. However, such errors are dependent on the block size which is being used, and hence are somewhat under control.

Another issue related to the block classification scheme is windowing. For each block (or frame), the feature vector is computed on a larger area obtained by applying a square window to the frame. If necessary, such as at image boundaries, zero-padding is used in the windowing. Different window sizes create different amounts of overlap between adjacent frames [38]. An optimal windowing scheme must be determined empirically.

CHAPTER IV

SOFTWARE DESIGN AND IMPLEMENTATION

So far, I have discussed feature extraction schemes and pattern recognition algorithms that have been developed for image segmentation. In this chapter, I will describe the design and implementation of the system, and expose important implementation details. The system was implemented in *C/C++* on a UNIX platform. The structure of the entire software package is illustrated in Figure 16.

4.1. Class Design

Several classes were designed to handle different issues involved in segmenting an image. Among them, the most important ones are: *Image_analysis*, *Image_segmentation*, *Ppm*, *Polygon*, and *Image_eval*.

Image_analysis

This class computes various features on a specific region separated from an image. The region may belong to one of the six predefined categories, i.e., tree, foliage, grass, bush, sky, and background sky (definitions of these categories will be given in Chapter V). To be compatible with the previous SBE classification capabilities, feature computation can also be done on an entire image.

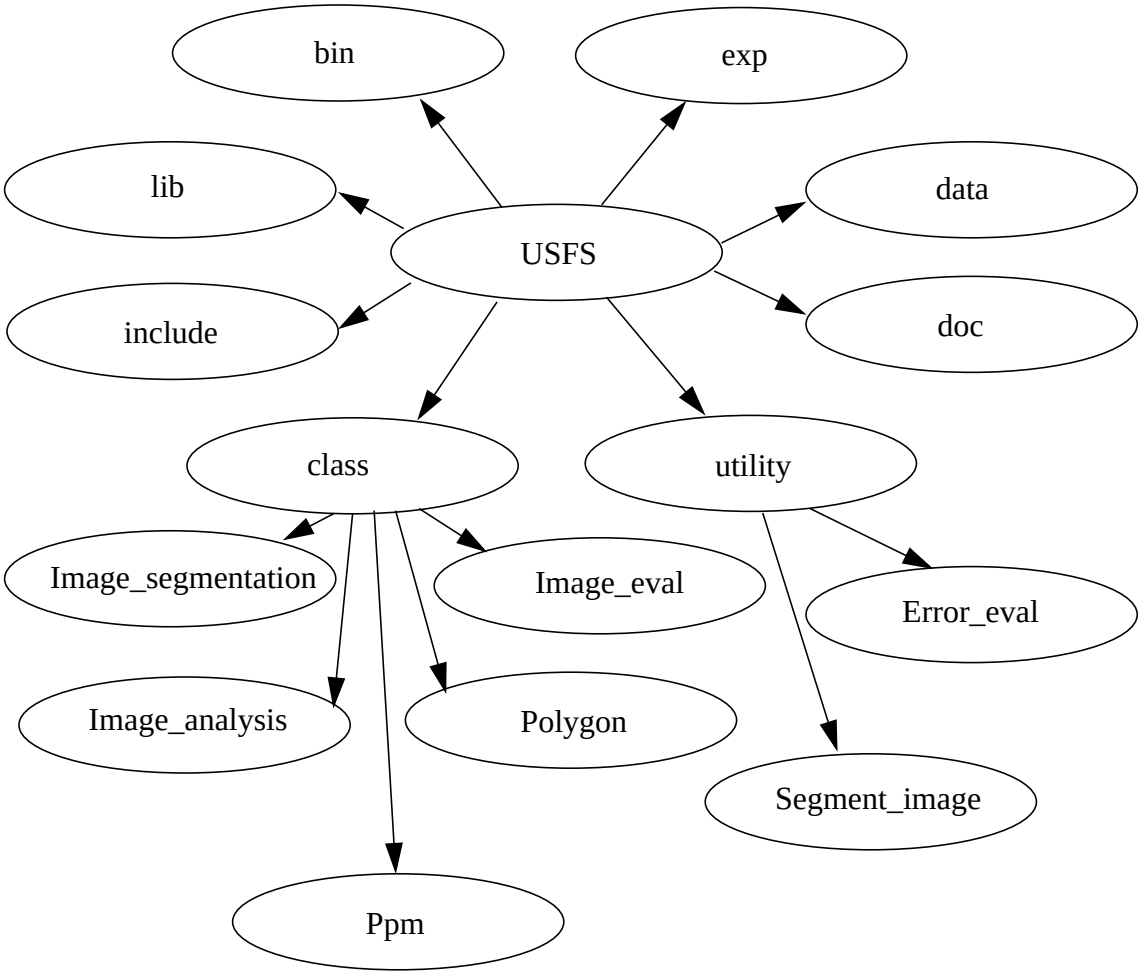


Figure 16. The structure of the software package.

For this class, some protected data members were defined. These include both general data representing information of parameter files, model files, what features to calculate, what windowing scheme to use, etc., and data specific to each feature extraction algorithm.

Public methods can be roughly grouped into the following categories: loading and writing methods, processing methods, and debugging methods. Loading and writing methods deal with the reading of parameter files and the writing of model or result files. Processing methods handle all the computation issues involved in training and testing, and hence are the most important. Debugging methods are used for debugging purposes.

All the algorithmic details for feature computation are hidden from the users in the private methods. These methods are grouped according to each kind of feature computation, and constitute the majority of the source code for this class.

Image_segmentation

This class segments an image into the six prescribed categories using the block-based segmentation algorithm described in Chapter III. It presents a programming interface for executing image segmentation. Details of the processing are handled by the *Image_analysis* class.

Protected data includes the desired processing mode (training or testing), the frame size, the window size, a pointer to the associated *Image_analysis* object, etc. Public methods include an initiation method, some loading and writing methods, and all the necessary processing methods. As I mentioned earlier, to apply windowing schemes, one

needs to deal with padding windows. Zero padding is used for this application. The related methods are declared as private methods of the *Image_segmentation* class.

Ppm

This class handles lower-level manipulation of portable pixel images (PPM). PPM is a common file format for digital images. In the *Ppm* class, some protected data members were designed to represent the unique information required for a PPM image, such as a magic number which identifies the associated image as a PPM image, width and height of the image, and the maximum color level of the image.

Important methods of the *Ppm* class are open and close methods, which handle the opening and closing of a PPM image, read and write methods, which read or write either the header information or the image data, assign methods, which set the format for an image, and comparison methods, which differentiate two PPM images. All these methods are public.

Polygon

To segment images using the block-level classification algorithm, training data needs to be prepared. This data was obtained by manually segmenting typical images from the database. Manual segmentation data was stored in data files using a format that consisted of the coordinates of the vertices of each polygon segments. The *Polygon* class was designed to deal with this reference segmentation data. It reads the coordinate data for the polygons from data files and determines whether a given point is inside a polygon.

Image_eval

This class was designed for the evaluation of the system performance in terms of error classification rate. An error evaluation component complements the design of the image segmentation system.

The relationship between all major classes is shown in Figure 17. As one can see, the *Image_analysis* class plays a key role in the system.

4.2. Utility Design

The classes are the core of the overall system. To perform a segmentation, one also needs some command line tools which pass all necessary parameters prescribed by the user to these classes and make the segmentation a reality. The following utilities were designed as the interface between the user and the system: *segment_image*, *error_eval*, and *labeler.*

Segment_image

This utility performs segmentation of an image. It operates either in a training mode or in a testing mode. In the training mode, model files about all prescribed

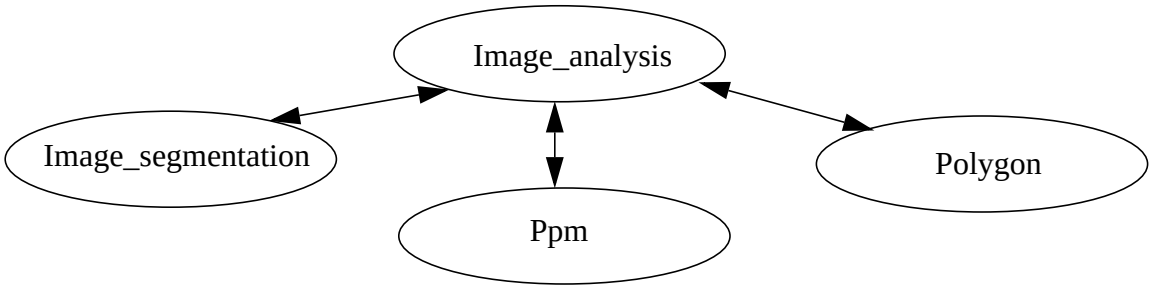


Figure 17. Relationship between the major classes.

categories are built. In the testing mode, a given image is segmented into regions belonging to various categories. The flow chart in Figure 18 demonstrates how these utilities work.

Error_eval

This utility evaluates the performance of a segmentation in terms of an error classification rate at the block level. To do the error evaluation, the block classification results are compared with the associated reference segmentation data, which was obtained previously by manual segmentation of the same image. Blocks which are incorrectly classified are counted, and the ratio of the number of classification errors to the total number of blocks being classified is computed as the error rate. The flow chart for this program is shown in Figure 19.

Labeler

This utility is designed specifically for manual segmentation. It was developed in *Tcl/Tk*. More detailed information about this tool will be presented in the next chapter.

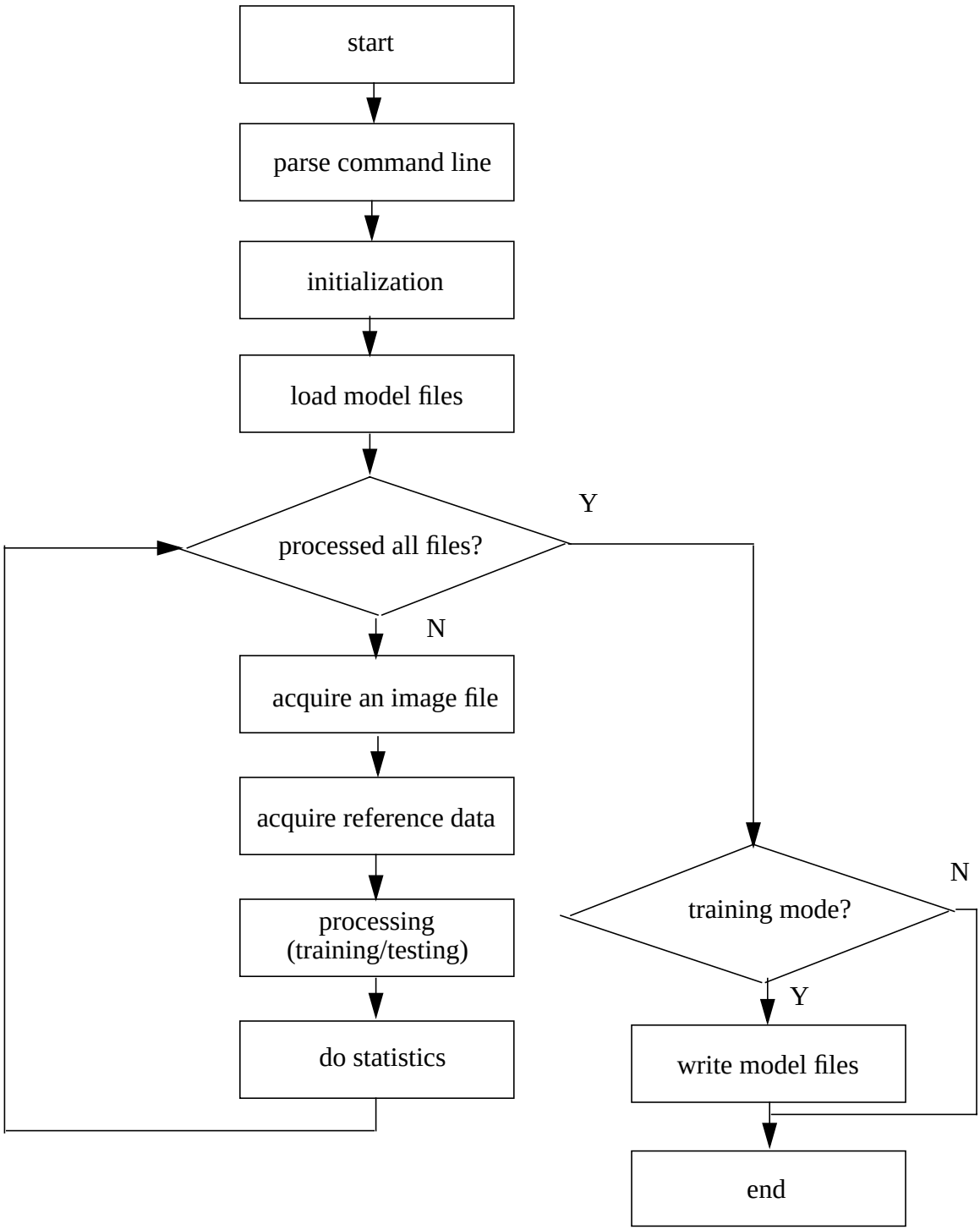


Figure 18. The flow chart for the shell program of the *segment_image* utility.

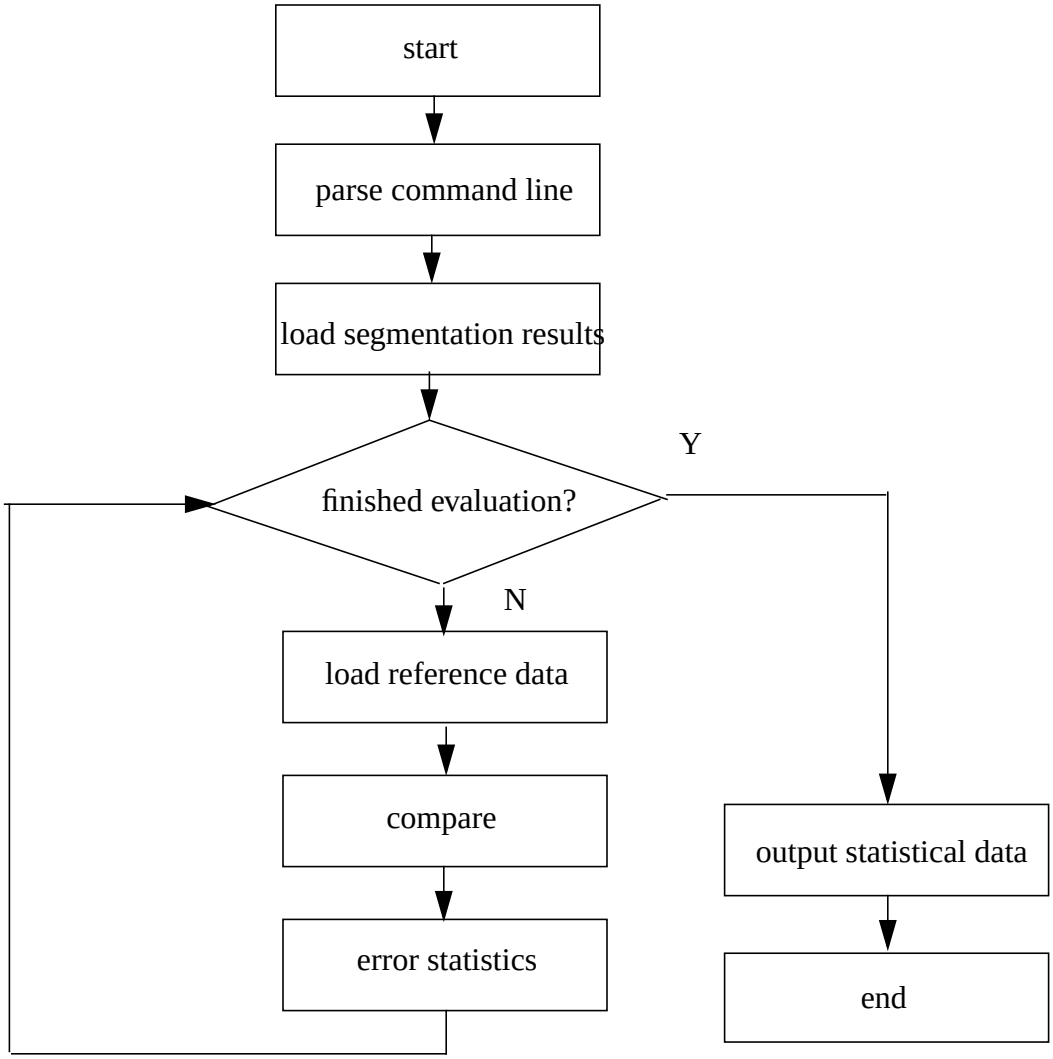


Figure 19. The flow chart for the shell program of the *error_eval* utility.

CHAPTER V

EXPERIMENTATION

I designed a variety of experiments to evaluate the segmentation system. These experiments are discussed in this chapter. First, I will describe the image database on which these experiments were carried out. Data preparation played an important role in these experiments. Next, I will introduce experiments designed to evaluate feature extraction algorithms. Finally, I will discuss experiments aimed at evaluating the overall system performance.

5.1. Image Database

Previously, an extensive database had been developed in conjunction with the United States Forest Service (USFS) to support the development of algorithms that will

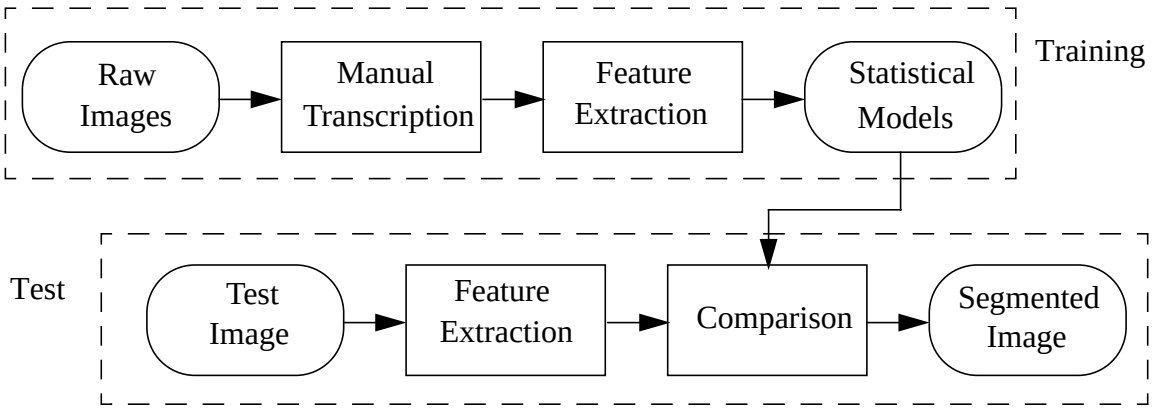


Figure 20. A block diagram of the overall system.

automate scenic beauty evaluation of forest images. The images included in this database were drawn from a study [39] spanning four years, which was carried out in the Ouachita-Ozark National Forests in Arkansas, USA. Photographs taken under controlled conditions had been digitized with an extremely high quality scanning process, and converted into Portable Pixel Map (PPM) format. A unique feature of the database is that each image was assigned a scenic beauty rating obtained by subjective assessment involving human subjects.

There are a total of 637 images in the database. These images are grouped into three categories [22] according to the associated subjective scenic beauty ratings. The three categories are: Low Scenic Beauty Estimate (LSBE), Medium Scenic Beauty Estimate (MSBE) and High Scenic Beauty Estimate (HSBE). The corresponding scenic beauty rating is distributed as shown in Figure 21. More details about the design and development of this database are available in a separate document [22].

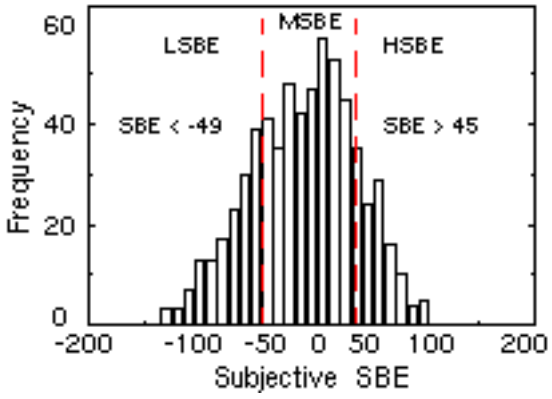


Figure 21. The distribution of the subjective scenic beauty rating across the database.

5.2. Manual Transcription

According to the system design, one needs transcribed training data to build statistical models. For this purpose, the digital images were segmented into different polygon regions, which were labeled with appropriate category names. A software tool, named Image Object Labeler (IOL), was developed to facilitate this task.

The IOL tool was rapidly prototyped in *Tcl/Tk*, an interpreted script language useful for graphical user interface design. It was designed to allow a user to select a certain type of object, and to draw a closed polygon around the region defining that object. Each image can be labeled multiple times. A list of images is traversed using a scrolling dialog box. This allows the user to iterate on a set of images until all segmentations are consistent. An example of a labeled image is shown in Figure 22.

For now, six categories have been defined for the image transcription. These categories are necessary for describing a forestry scene. The definitions of these categories are given in Table 1. The transcription results are saved as ASCII data files. A data file begins with the name of the associated image file, followed by an object name, then the coordinates of all vertices of the polygon used to mark this object. There can be multiple instances of any object in a file, and the order of the objects is arbitrary (each object is written in the order it was created). The format of the data file is shown in Figure 23. With the tool introduced above, all 637 images in the USFS database were manually transcribed. These transcriptions established the basis of the image segmentation system.

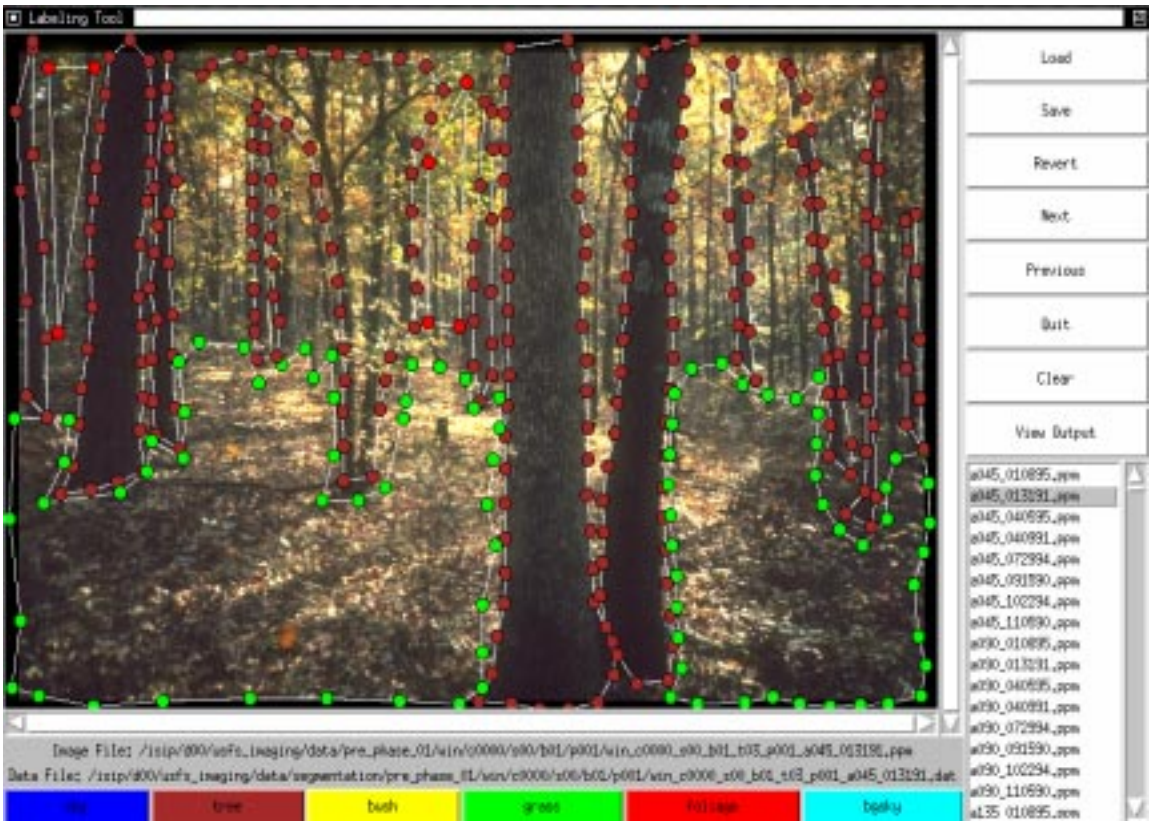


Figure 22. An example image labeled by using the manual segmentation tool.

Table 1. The six categories used in manual transcription of the forest images.

Category	Description
tree	a woody perennial plant having a single, usually elongated, main stem, generally with few or no branches on its lower part
foliage	a cluster of leaves, flowers, and branches
bush	a low densely branched shrub or a close thicket of shrubs suggesting a single plant
grass	grass or land covered with growing grass, or other green-colored ground cover
sky	a region typically containing sky and few obstructions such as trees; strictly limited to the top of the image
background sky	a region with sky in the background and other objects in the foreground, but still clearly identified as sky

5.3. Experiments with DCT-based Features

As explained in Chapter II, I designed several new feature vectors for the image segmentation system. Since the feature extraction scheme based on discrete cosine transform involved several parameters, I executed a series of experiments to optimize the parameters.

The evaluation experiments were run on a pilot data set. The training set is shown in Table 2. Note that different images were chosen to train different models. The reason for this choice was to make the training process more efficient. Since the six categories of structures are distributed unevenly across the database, if the same training set had been used for all the models, the training set would have had to be increased to a much larger size to guarantee a sufficient amount of training data (image blocks) for each model. The test data set consisted of 5 random images with a total of 1406 64 x 64 image blocks.

As a first step, I evaluated the effects of the windowing scheme on the baseline design discussed in Chapter II. The results indicate that system performance is sensitive to the windowing parameters. As one can see from Table 3, when the frame and window

```
/isip/d00/usfs_imaging/data/phase_01/cd_0012/img0002.ppm
Class tree
954  32
954  386
...
1106  30
Class foliage
186  32
...
```

Figure 23. An example of the output file format used by the transcription tool to store manual transcribed data. A simple ASCII file format is used here.

sizes were set to be 64 x 64 and 96 x 96 respectively, the system performed best on green pixels. Next, I investigated the performance with different color features, i.e., I computed DCT coefficients on the red, green and blue pixel components respectively. For these evaluations, the frame size was set to 64 x 64, and the window size was 96 x 96. As shown in Table 4, generally the error rates with all three colors were comparable. This similarity is reasonable because DCT coefficients are supposed to describe the spatial variation, or the texture pattern an image block displays. These texture characteristics should not vary remarkably for different color components.

In the schemes discussed so far, I chose the frequency filter size to be 4. However, that filter parameter may not necessarily be the optimal choice. Therefore, I tested a smaller filter size of 2. In this scenario, all other conditions for feature generation were kept the same as the baseline design. The windowing parameters also remained unchanged, that is, I used 64 x 64 frames and 96 x 96 windows. It is observed in Table 5 that the system with a larger filter worked slightly better. However, the difference in the error performance was not significant.

Table 2. The training data set for DCT-based feature evaluation.

Structure	Number Of Images	Number Of Blocks	Block Size
tree	5	523	64 x 64
foliage	6	632	64 x 64
bush	5	520	64 x 64
grass	6	523	64 x 64
background sky	10	675	64 x 64
sky	10	61	64 x 64

Table 3. Performance for a variety of windowing schemes.

Experiment No.	Frame Size	Window Size	Percentage Error Rate
1	32 x 32	32 x 32	72.6
2	32 x 32	64 x 64	67.6
3	64 x 64	64 x 64	67.9
4	64 x 64	96 x 96	**66.6**
5	64 x 64	128 x 128	67.6

Table 4. Performance across different color components.

Experiment No.	Color Component	Percentage Error Rate
1	red	**65.7**
2	green	66.6
3	blue	67.3

Table 5. Performance as a function of the frequency filter sizes.

Experiment No.	Filter Size	Percentage Error Rate
1	4	66.6
2	2	69.2

Finally, I evaluated the impact of the number of DCT coefficients. The results are shown in Table 6. I used the first 64 and 128 coefficients to generate feature vectors respectively. For the first experiment, I had a 16-dimensional feature vector, while for the second experiment, I used a 32-dimensional vector. The results verified the idea that using more DCT coefficients is not necessarily better — with 128 DCT coefficients, the system performance was 5% worse than that of the system with the features based on only 64 coefficients.

5.4. Experiments with Gabor Features

To evaluate Gabor filter-based feature design, I selected several typical forest structures from the database, and carried out pair-wise discrimination experiments on these regions. Some example images and segmentations are shown in Figures 24 and 25. In general, with these Gabor features, discrimination between structures was good. Most misclassification errors occurred in shaded areas — textures there were difficult to distinguish because of poor lighting conditions. It is interesting to note that in the experiment with the “bgsky-bush” pair, although the error rate was 22.0%, the segmenter was able to figure out the exact contour of the background sky area. The surrounding

Table 6. The performance evaluation results with different amounts of DCT coefficients involved in the feature computation.

Experiment No.	Amount Of DCT Coefficients Involved	Percentage Error Rate
1	64	66.6
2	128	72.5

branches and leaves in the transcribed background sky region look more like the bushes on the right half of the image, resulting in their classification as bushes.

On the other hand, the computational requirements for the algorithm are comparatively high. With this algorithm, computing feature vectors for all pixels within a 128 x 128 image block took around 120 minutes on a 333 MHz processor with 512 Mbytes of memory. In this application, the dimension of an image is 1536 x 1024. Consequently, if one computes the Gabor features for each pixel, computation on only one

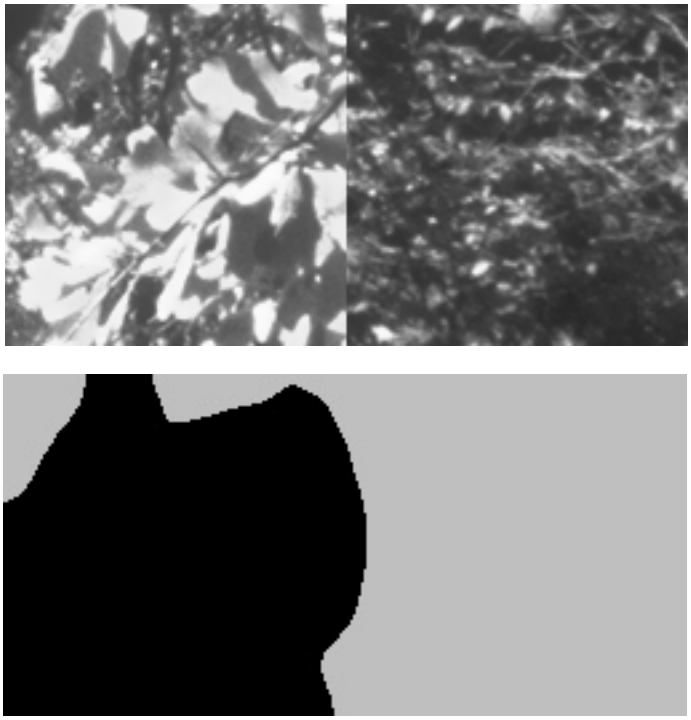


Figure 24. An example segmentation generated by applying the Gabor feature design to typical forest structures. The image consists of two categories. The left half is a typical “foliage” block, and the right half is a “bush” block. In the segmentation image, the black region stands for hypotheses for the “foliage” category, while the grey color is used for “bush.” The pixel classification error rate was 7.3%.

image would be approximately 192 hours. Hence, I investigated ways to speed up the algorithm without sacrificing performance.

Since the system is actually doing block-level classification, only one feature vector is needed for an image block. As observed in the previous exploration, the Gabor feature values computed for pixels of the same category did not vary dramatically. Hence it would be feasible to compute a feature vector for an image block by averaging over a small amount of neighboring pixels. In the experiments, it was assumed that a feature vector computed at the central pixel of an image block represented all characteristics

Figure 25. Another example segmentation with the Gabor feature design. The left half image is a typical “bgsky” (background sky) block, and the right half is a “bush” block. In the segmentation image, the black region stands for hypotheses for the “bgsky” category, while the grey color is used for “bush.” The pixel classification error rate was 22.0%.

necessary to distinguish this block from blocks of other categories. With this assumption, the feature vector for a block center was used as the vector for the whole block. Hence the feature computation time for one 128 x 128 block was reduced from 120 minutes to 12 minutes. In accordance with this improved feature design, to run a training job on 1000 image blocks with a size of 128 x 128, the required computation time is approximately 200 hours or 8.3 days.

5.5. Evaluating the Performance of the System

The performance of the image segmentation system was evaluated with three different feature vectors on set 1 of the USFS Pre-Phase 01 data, which consists of 478 training images and 159 test images formatted as 1536 x 1024 PPM images [22]. The three feature sets were chosen:

- **blue + brown**: These are the histograms for colors blue and brown. Blue is an important color for classification of sky and background sky, while brown distinguishes tree stems from other specified categories.
- **blue + brown + sl**: Here, “sl” stands for the density of short lines of an image. Short lines are characteristic of bushes and grasses in a forest scene.
- **rgb + ent**: Here, “rgb” denotes the histograms for red, green and blue. “Ent” denotes entropy. “Rgb” combines the three primary colors together. It describes to some extent general luminance information, as well as the lighting condition in the forest. Entropy is a measure of the randomness of an image.

For the experiments described below, the frame size was set to 64 x 64, which presented an appropriate sampling rate of the original images sized at 1536 x 1024. The

window size was set to be 128 x 128, resulting in a 50% overlap between neighboring windows.

The evaluation results are shown in Table 7 and the associated confusion matrices are given in Tables 8 to 10. Also, some example segmentations are shown in Figure 26. A confusion matrix is interpreted as follows. Numbers in the first column show how tree blocks were classified by the software. As an example, the matrix in Table 8 shows $[2619, 122, 855, 985, 697, 49]^T$ in the first column, which means out of a total of 5327 tree blocks, only 2619 blocks were classified correctly. Others were classified to incorrect categories (122 blocks to foliage, 855 blocks to bush, and so on). The other columns are organized for the remaining five categories in the same manner.

5.6. Analysis

The best performance of the segmentation system was a 61.4% block classification error rate, obtained by combining color histograms ("rgb") with entropy as the feature set. Although the error rate was comparatively high, the blocks were generally classified in a reasonable manner — with sky and background sky in the top, grass at the bottom, etc., as shown in Figure 26. This is a promising trend toward a successful system.

I explored further to see what were the major causes for the classification errors. I examined the confusion matrices from the above experiments and noticed that a significant portion of the errors occurred within the foliage blocks. With the best system, of which "rgb + entropy" was used as the feature set, the respective error rates for each category are summarized in Table 11.

Table 7. Evaluation results for the block-level image segmentation system.

Feature Sets	Frame Size	Window Size	Percentage Error Rate
blue + brown	64 x 64	128 x 128	70.8
blue + brown + sl	64 x 64	128 x 128	70.2
rgb + ent	64 x 64	128 x 128	61.4

Table 8. The confusion matrix with the feature set as “blue + brown.” The corresponding error rate was 70.8%.

Category	tree	foliage	bush	grass	bgsky	sky
tree	2619	5100	2567	3332	179	0
foliage	122	1788	528	384	35	0
bush	855	5024	4595	3096	1	0
grass	985	1962	2818	3001	14	0
bgsky	697	3299	508	420	1464	1
sky	49	77	167	140	266	9

Table 9. The confusion matrix with the feature set as “blue + brown + short lines.” The corresponding error rate was 70.2%.

Category	tree	foliage	bush	grass	bgsky	sky
tree	2437	4913	2381	2978	182	0
foliage	170	1952	575	457	40	0
bush	822	4796	4428	2971	3	0
grass	1156	2315	3167	3441	14	0
bgsky	695	3201	475	388	1464	1
sky	47	73	157	138	256	9

Obviously, a better foliage classifier would be a key to improving the system. To this end, I studied the foliage classifier from a statistical viewpoint. In the PCA system, distances between a test feature vector and all the reference vectors were computed. A block was assigned to the label associated with the minimum distance. I computed the distribution of the distances, or scores, for the foliage classifier in terms of false-alarms and accuracy. The distribution of the scores are shown in Figures 27 and 28. From these plots, it is clear that the foliage classifiers used in these experiments were not capable of discriminating foliage blocks from non-foliage blocks. This explains the majority of errors which occurred. Further, I studied all other features not used in the experiments with regard to the foliage classification in the same manner and obtained similar results.

It is interesting to compare the performance of the segmentation system to a system which makes intelligent guesses based on prior knowledge about the categories under study. According to Bayes' decision rule [36], we can minimize the overall errors by weighting our estimate of the probability of each class by the prior probabilities of each class. In this application, the prior probabilities of the categories being studied are unknown. However, one can estimate these probabilities using a frequency of occurrence computed over the training set [36]. For the data set used in the evaluation experiments, these frequencies are shown in Table 12. We can construct a simple system that always guesses the most probable class. In this case, the system will always classify an image block as "foliage" since this class has the highest frequency of occurrence: 32.0%.

Obviously, the classification error rate of this system is 68.0%. The best performance of the segmentation system (61.4% classification error) is better than this

baseline guessing system. With further improvement of the foliage classifier, the segmentation system is expected to yield much more reliable results. Such improvements will eventually make the segmentation system suitable for practical use.

Table 10.The confusion matrix with the feature set as “rgb + entropy.” The corresponding error rate was 61.4%.

Category	tree	foliage	bush	grass	bgsky	sky
tree	2807	4494	1717	2454	181	0
foliage	260	2402	609	259	98	0
bush	911	6412	6252	2578	16	0
grass	831	1479	2208	4888	7	0
bgsky	504	2450	391	169	1437	1
sky	14	13	6	25	220	9

Table 11. Statistics of classification errors for the experiment which used “rgb + entropy” as the feature set and yielded the error rate of 61.4%.

Category	tree	foliage	bush	grass	bgsky	sky
Block Total	5327	17250	11183	10373	1959	10
Errors	2520	14848	4931	5485	522	1
Error Rate (%)	47.3	86.1	44.1	52.9	26.6	10.0

Table 12. Computed frequencies of occurrence for all six categories under study. Note the total number of blocks is 136513.

Category	tree	foliage	bush	grass	bgsky	sky
Number of Occurrence	23526	43714	37647	24040	7450	136
Frequency of Occurrence	17.2%	32.0%	27.6%	17.6%	5.5%	0.1%

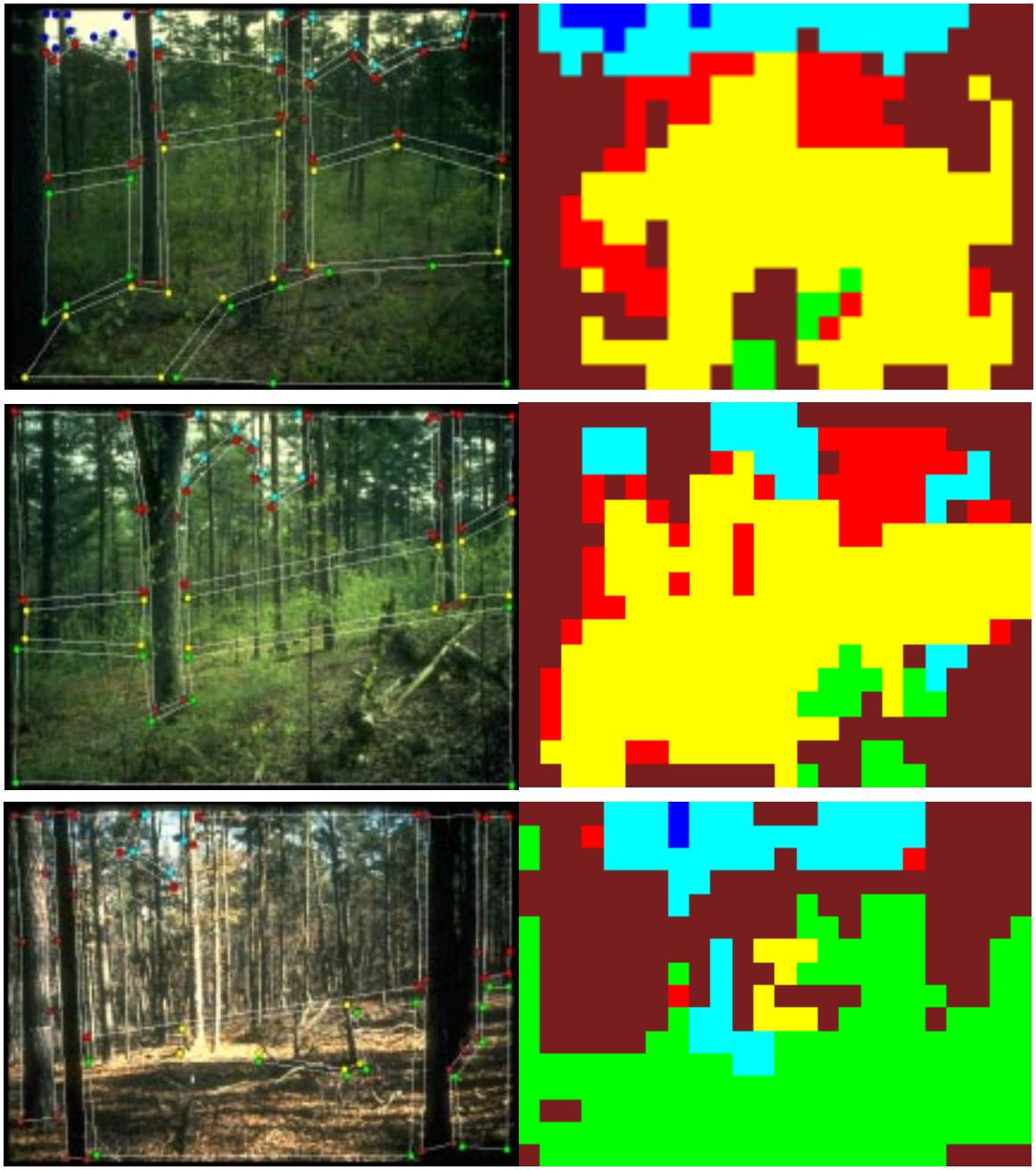


Figure 26. Example segmentations produced by the software on some typical forest images. Images on the left are the original ones with manual transcriptions added. For both manual transcriptions and automatic segmentations, different colors are used to distinguish between categories. The mapping relationship is: brown - tree, red - foliage, yellow - bush, green - grass, dark blue - sky, and light blue - background sky. Note most segmentations for sky and background sky were generated accurately.

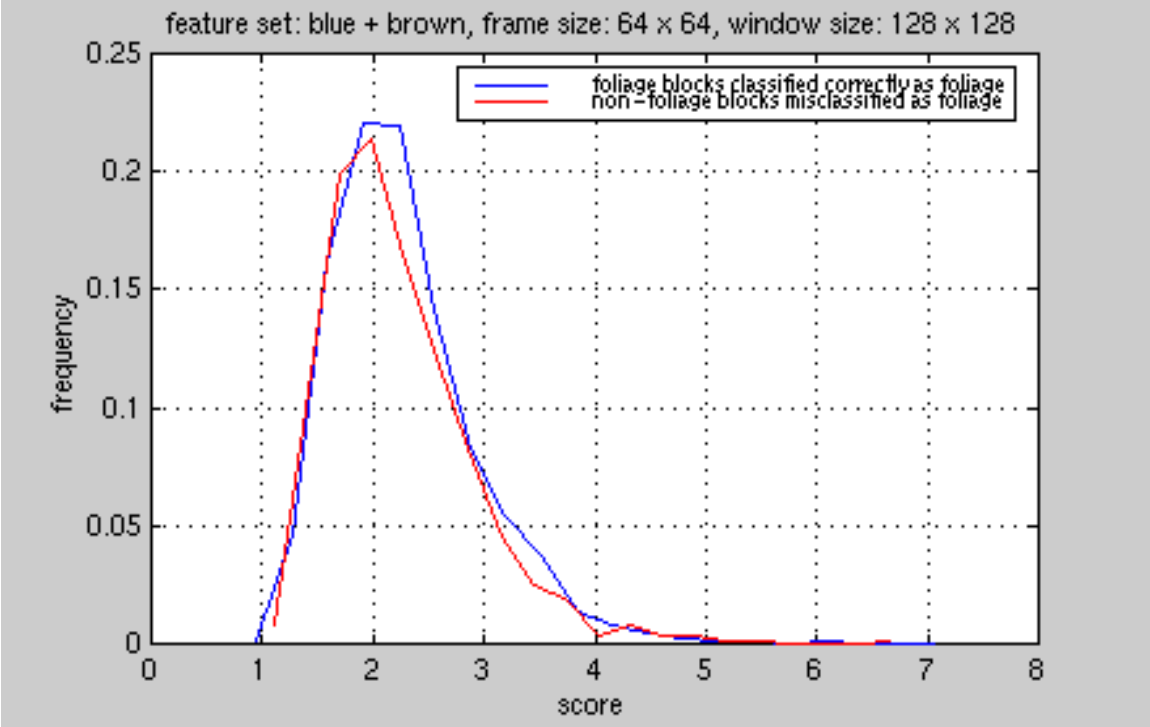


Figure 27. Distribution plot of scores (distances) generated by the foliage block classifier of the image segmentation system. The conditions for this experiment are listed in the title of the plot.

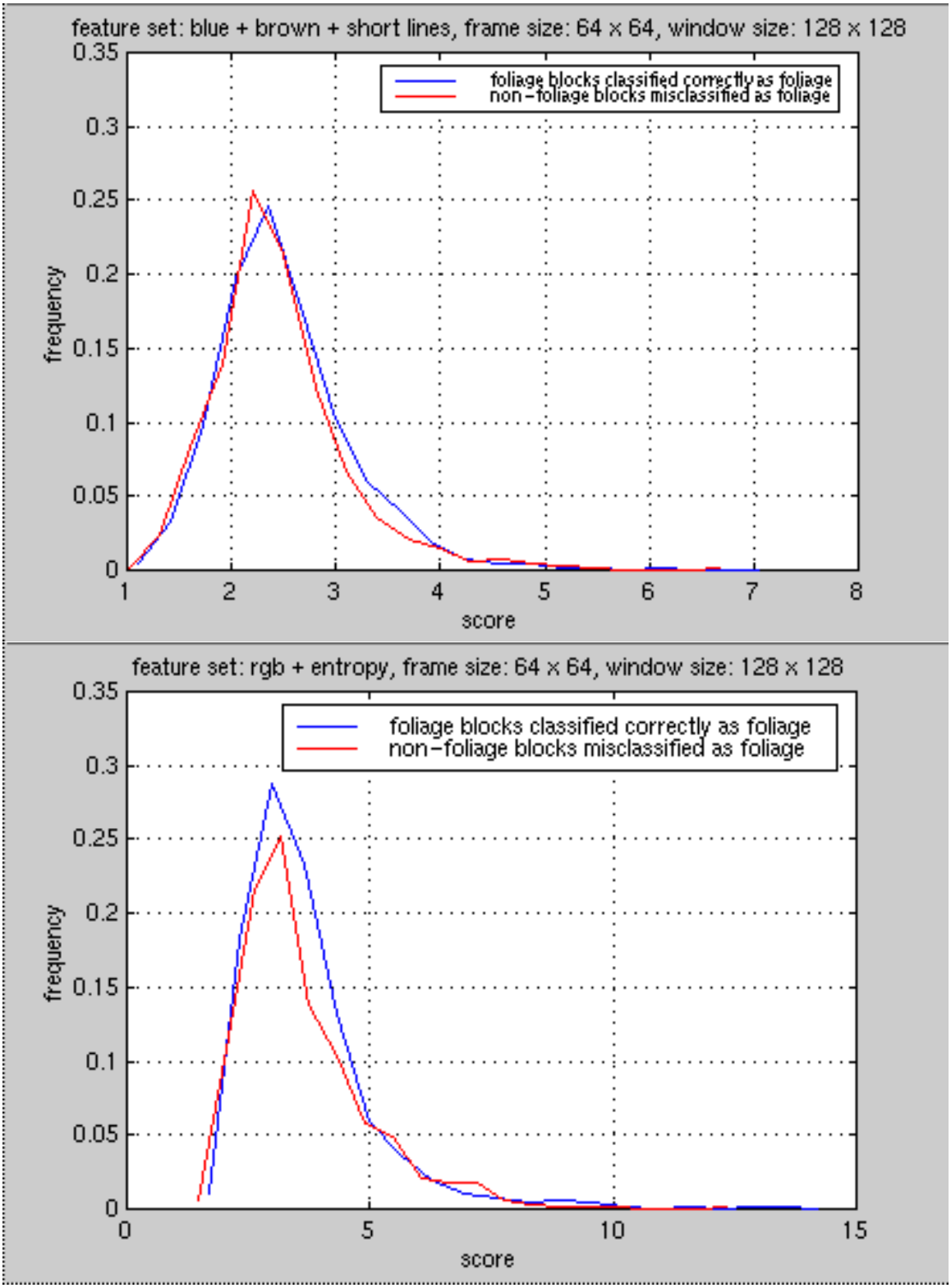


Figure 28. Score distribution plots for the other two experiments.

CHAPTER VI

IMPACT ON THE SBE PROBLEM

As discussed in the previous chapters, the image segmentation system was developed in an attempt to facilitate automated evaluation of forest resources. An important aspect of this task is scenic beauty estimation of forest scenes, which is referred to as the SBE problem. The previous SBE software produced good results on scenic beauty quality-based classification [19], and I attempted to assess the impact a good image segmentation system could have on this application.

6.1. Cheating Experiments

As a first step, I designed some "cheating experiments" to facilitate the investigation. For these experiments, it is assumed that image segments have already been accurately generated. This assumption is easy to satisfy given that the manual transcriptions of all images in the database are available. Then images are classified into different SBE categories (HSBE, MSBE, and LSBE) on the basis of feature vectors computed on a specific kind of segment.

As an example, suppose an SBE classification of "grass" segments only is performed using "rgb" (histograms of red, green and blue) as the feature set. First of all, PCA models need to be trained on the specified segments. To do this, all pixels of a

training image which fall into grass segments are located by checking the manual transcription. Then the “rgb” histograms for the image are computed only on these “grass pixels.” With feature data extracted in the same manner for all training images, the trained statistical models are obtained. Similarly, when a given image is tested, only the feature vector for “grass pixels” of the image is computed.

I carried out such experiments with several combinations of features. Again, the experiments were based on data set 1 of USFS Pre-Phase 01 images. The feature sets used were “rgb + entropy,” “rgb + entropy + long lines,” and “blue + brown.” As discussed earlier in Chapter V, “rgb” histograms may describe the lighting condition in a forest, and “entropy” quantifies the randomness of an image. Both are very important in scenic beauty estimation. I also included in the experiments the feature of “long lines,” which would help to identify tree segments. The error rates and confusion matrices are shown in Tables 13 to 15.

6.2. Analysis

Overall, SBE classification on a particular type of image, such as background sky, produced worse results than doing the classification on an entire image. This is clearly revealed by the error rates in Table 13. This comparison indicated that the feature sets I tested were not doing as good a job of discrimination. However, there are some interesting observations with the confusion matrices.

As shown in Table 14 and 15, when “blue + brown” was used as the feature set, classification of segments transcribed as background sky (bgsky) decreased errors for HSBE image classification by half. According to research publications on scenic beauty

estimation [21], an HSBE forest image usually features little visual penetration through the foreground foliage and twigs. Since blue is a characteristic color for sky, by examining this feature on the “background sky” segments, I was able to do much better in separating HSBE images from other categories. Similarly, with the same settings, classification errors for LSBE images decreased by one-third on the segments labeled as trees. Such a reduction in errors for individual classes was also found for the other conditions presented in Table 13. These observations demonstrate a likelihood that one could do a much better job in characterizing images with regard to their scenic beauty quality if the analysis focuses on a certain type of segments, rather than the image as a whole. However, further investigation is necessary to verify this hypothesis.

Table 13. Results of the cheating experiments in terms of percentage error rate. Note the last column gives the classification error rate with the previous SBE system, which performs classification on an entire image, not a specific category of segments.

Features	Tree	Foliage	Bush	Grass	Bgsky	Entire Image
rgb + ent	47.8	46.5	52.8	47.2	55.3	**39.6**
rgb + ent + ll	48.4	49.1	51.6	46.5	57.2	**42.1**
blue + brown	45.9	45.3	54.7	42.1	56.6	**31.4**

Table 14. Example confusion matrices. The left matrix is for the SBE classification on background sky segments with “blue + brown” as the feature set. The right matrix was generated from the corresponding experiment on entire images. Note the errors for the HSBE image classification decreased by half.

Automatic	LSBE	MSBE	HSBE
LSBE	5	12	0
MSBE	19	45	9
HSBE	4	46	**19**

Automatic	LSBE	MSBE	HSBE
LSBE	2	2	0
MSBE	24	97	18
HSBE	2	4	**10**

Table 15. Another example. The left matrix is for the SBE classification on tree segments with “blue + brown” as the feature set. The right matrix is for the corresponding experiment on entire images. The errors for the LSBE image classification decreased by one-third.

Automatic	LSBE	MSBE	HSBE
LSBE	**9**	27	2
MSBE	16	66	15
HSBE	3	10	11

Automatic	LSBE	MSBE	HSBE
LSBE	**2**	2	0
MSBE	24	97	18
HSBE	2	4	10

CHAPTER VII

CONCLUSIONS AND FUTURE WORK

Image segmentation is essential to analyzing forest images for automated forest resource evaluation. The success of an image segmentation system relies on appropriate designs of feature extraction schemes and pattern recognition algorithms. In this chapter, I will conclude the thesis with a discussion of my major results on these designs and suggestions for future work.

7.1. Conclusions

The first major contribution of the thesis is the evaluation of various feature design schemes for segmenting forest images. Color histograms, entropy, and density of lines, which are important in evaluating the scenic beauty quality of forest images, are still very useful in segmenting the images. The combination of color histograms (“rgb”) with entropy produced 61.4% block classification error rate, which was the best segmentation performance achieved with the system. This performance is better than guessing based on prior knowledge, which yielded an error rate of 68%. The discrete cosine transform (DCT) is an efficient technique for representing images in the spatial frequency domain. With a feature design based on filtering DCT coefficients computed from green pixels, I obtained a 66.6% block classification error rate, which was comparable to the performance

generated with features such as color histograms and entropy. There was no significant difference between DCT features computed with different colors in terms of the error rate.

Gabor filters are also a very promising technique for image segmentation. The Gabor filter bank-based feature extraction approach yielded an error rate of 7.3% in a pair-wise discrimination task. The computational inefficiency of this approach precludes evaluation on a large-scale task. However, this level of performance is very encouraging compared to other conventional techniques.

The other major contribution of this thesis is the development of a pattern recognition-based image segmentation algorithm. I presented an algorithm that divides an image evenly into small blocks, and then classifies each block into one of the six prescribed categories (namely, tree, bush, grass, foliage, background sky and sky) using class-specific principal component analysis. This algorithm is simple and efficient with regard to the forestry application. It significantly reduced the computational complexity at the expense of sacrificing the accuracy of block classification at borders between regions of different categories.

I evaluated this image segmentation system in terms of error rate using a standard pattern recognition approach to classification. As mentioned above, the best performance was achieved by combining color histograms (“rgb”) with entropy, resulting in a 61.4% block classification error rate. I analyzed the classification errors and determined that the classifier for foliage blocks caused the majority of the errors.

A by-product of this work was a large database of manually segmented images. Six categories of image segments typical of a forest scene were defined. A transcription tool

was developed to facilitate the manual work, and all 1784 images in the database were transcribed. This database will be a useful tool for supporting future research on these problems, and complements other extensive resources available for this data [22].

Finally, I investigated the impact an image segmentation system could have on the scenic beauty estimation problem. With carefully designed “cheating experiments,” I determined that SBE classification errors decreased on some segments, but overall performance was lacking. This fact indicated that one might be able to characterize the scenic beauty quality of an image more accurately by analyzing different categories of segments separately, rather than mixing them together.

7.2. Future Work

First, the feature design based on the DCT coefficients needs to be improved. There are still parameters, such as the frequency filter size, and the number of DCT coefficients involved in the computation, which need to be optimized. Moreover, to combine the DCT coefficients more effectively, one will have to understand thoroughly the relationship between the physics of the visual system and the information present in the discrete cosine transform representation.

The Gabor filter bank-based features generated very encouraging results in the pair-wise discrimination experiments. To further evaluate their effectiveness in the segmentation task, it is necessary to improve the computational efficiency of this approach. Once this barrier is overcome, one will be able to explore the feature design within the PCA segmentation system.

Lighting conditions are a very important factor in scenic beauty evaluation [21]. People tend to consider a bright forest scene to be of low scenic beauty. However, most popular feature extraction approaches, such as those presented in this work, do not do a good job of representing this information. The contrast [40] of an image helps to describe the lighting condition, hence is worthy of investigation.

Another unique aspect of natural images is the strong correlations between neighboring pixels [41] since nearby regions in a natural scene usually have similar intensities and colors. Syntactic constraints [42] can be imposed on an image that model these correlations. This is part of a class of techniques known as Bayesian belief networks [36]. With Bayesian belief networks, dependencies between variables involved in a system are represented graphically using a network structure and Bayes rule is used to calculate probabilities. Incorporation of such techniques into the segmentation system is expected to enhance the system's performance significantly.

Finally, decision tree based approaches to block classification [36] are also worth exploring. Decision trees have the potential for developing extremely powerful context-sensitive statistical models through the integration of many diverse knowledge sources. They are suitable for tasks where there is large overlap in the absolute feature space, or discrimination amongst classes in the feature space requires nonlinear decision surfaces. This overlap can often be reduced by integrating other knowledge sources. However, a large engineering effort is often required to determine the best architecture for such systems. Identification of good knowledge sources and the most appropriate way to integrate this knowledge often requires extensive training databases and significant

amounts of experimentation. Nevertheless, such approaches can often produce impressive performance under the proper conditions.

REFERENCES

[1] K. R. Castleman, *Digital Image Processing*, Prentice Hall, Upper Saddle River, New Jersey, USA, 1996.

[2] N. R. Pal and S. K. Pal, "A Review on Image Segmentation Techniques," *Pattern Recognition*, vol. 26, no. 9, pp. 1277-1294, September 1993.

[3] K. S. Fu and J. K. Mui, "A Survey on Image Segmentation," *Pattern Recognition*, vol. 13, no. 1, pp. 3-16, January 1981.

[4] A. Rosenfeld and A. C. Kak, *Digital Picture Processing*, Academic Press, New York City, New York, USA, 1982.

[5] T. W. Ridler and S. Calvard, "Picture Thresholding Using an Iterative Selection Method," *IEEE Transactions on Systems, Man and Cybernetics*, vol. SMC-8, no. 8, pp. 630-632, August 1978.

[6] N. Ostu, "A Threshold Selection Method from Gray Level Histograms," *IEEE Transactions on Systems, Man and Cybernetics*, vol. SMC-9, no. 1, pp. 62-66, January 1979.

[7] J. S. Weszka and A. Rosenfeld, "Threshold Evaluation Techniques," *IEEE Transactions on Systems, Man and Cybernetics*, vol. SMC-8, no. 8, pp. 622-629, August 1978.

[8] B. Chanda, B. B. Chaudhuri and D. D. Majumder, "On Image Enhancement and Threshold Selection Using the Gray Level Co-occurrence Matrix," *Pattern Recognition Letters*, vol. 3, no. 4, pp. 243-251, July 1985.

[9] Y. Nakagawa and A. Rosenfeld, "Some Experiments on Variable Thresholding," *Pattern Recognition*, vol. 11, no. 3, pp. 191-204, May 1979.

[10] S. Geman and D. Geman, "Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-6, no. 6, pp. 721-741, November 1984.

[11] H. Derin, H. Elliott, R. Cristi and D. Geman, “Bayes Smoothing Algorithms for Segmentation of Binary Images Modeled by Markov Random Fields,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-6, no. 6, pp. 707-720, November 1984.

[12] W. E. Blanz and S. L. Gish, “A Connectionist Classifier Architecture Applied to Image Segmentation,” *Proceedings of the 10th International Conference on Pattern Recognition*, vol. 1, pp. 272-277, Atlantic City, New Jersey, USA, June 1990.

[13] N. Babaguchi, K. Yamada, K. Kise and T. Tezuka, “Connectionist Model Binarization,” *Proceedings of the 10th International Conference on Pattern Recognition*, vol. 1, pp. 51-56, Atlantic City, New Jersey, USA, June 1990.

[14] C. Cortes and J. A. Hertz, “A Network System for Image Segmentation,” *Proceedings of the International Joint Conference on Neural Networks*, vol. 1, pp. 121-125, Washington DC, USA, June 1989.

[15] Y. Ohta, T. Kanade and T. Sakai, “Color Information for Region Segmentation,” *Computer Graphics and Image Processing*, vol. 13, no. 3, pp. 222-241, July 1980.

[16] Y. W. Lim and S. U. Lee, “On the Color Image Segmentation Algorithm Based on the Thresholding and Fuzzy C-means Techniques,” *Pattern Recognition*, vol. 23, no. 9, pp. 935-952, September 1990.

[17] J. C. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms*, Plenum Press, New York City, New York, USA, 1981.

[18] J. C. Bezdek and S. K. Pal, *Fuzzy Models for Pattern Recognition: Methods That Search for Structures in Data*, IEEE Press, New York City, New York, USA, 1992.

[19] N. Kalidindi, V. Ramani and J. Picone, “Scenic Beauty Estimation of Forestry Images,” *Final Project Report for the Southern Forest Experiment Station, United States Forest Service*, available at *http://www.isip.msstate.edu/publications/reports/usfs_imaging/1998/scenic_beauty_estimation/*, Institute for Signal and Information Processing, Mississippi State University, December 1998.

[20] T. C. Brown and T. C. Daniel, “Scaling of Ratings: Concepts and Methods,” *Research Paper RM-293*, United States Department of Agriculture, Forest Service, Rocky Mountain Forest and Range Experiment Station, Fort Collins, Colorado, USA, September 1990.

[21] L. M. Arthur, T. C. Daniel, and R. S. Boster, “Scenic Assessment: an Overview,” *Landscape Planning*, vol. 4, no. 2, pp. 109-129, February 1977.

[22] N. Kalidindi, A. Le, and J. Picone, “Scenic Beauty Estimation Database,” *Quarterly Project Report for the Southern Research Station, United States Forest Service*, available at *http://www.isip.msstate.edu/publications/reports/usfs_imaging/1996/database/*, Institute for Signal and Information Processing, Mississippi State University, December 1996.

[23] J. D. Murray and W. vanRyper, *Encyclopedia of Graphics File Formats*, O’Reilly and Associates, Inc., Sebastopol, California, USA, 1994.

[24] J. Canny, “A Computational Approach to Edge Detection,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 8, no. 6, pp. 679-698, November 1986.

[25] Z. Long, J. Picone, and V. Rudis, “The Optimization of Edge and Line Detectors for Forest Image Analysis,” *Proceedings of the 4th World Multiconference on Systemics, Cybernetics and Informatics*, vol. 5, pp. 171-176, Orlando, Florida, USA, July 2000.

[26] M. Rabani and P. W. Jones, *Digital Image Compression Techniques*, SPIE Optical Engineering Press, Bellingham, Washington, USA, 1991.

[27] K. C. Clarke, “Computation of the Fractal Dimension of Topographic Surfaces Using the Triangular Prism Surface Area Method,” *Computers and Geosciences*, vol. 12, no. 5, pp. 713-720, September 1986.

[28] V. A. Rudis, R. E. Thill, J. H. Gramann, J. Picone, N. Kalidindi and P. Tappe, “Understory Structure by Season Following Uneven-aged Reproduction Cutting: a Comparison of Selected Measures Two and Six Years after Treatment,” *Forest Ecology and Management*, vol. 114, no. 2-3, pp. 309-320, January 1999.

[29] W. Yhang, J. H. Gramann and V. A. Rudis, *Color Visibility by Season and Silvicultural Treatment: Effects on the Scenic Beauty of Southern U.S. Pine-Oak Forests*, Texas A&M University, College Station, Texas, USA, and USDA-FS Southern Research Station, Forest Inventory and Analysis Unit, Starkville, MS, USA, October 1998.

[30] W. Yhang, *The Effect of Color on the Perceived Scenic Beauty of Pine-Oak Plots in the Ouachita National Forest, Arkansas*, Ph. D. Thesis, Department of Recreation, Park, and Tourism Sciences, Texas A&M University, College Station, Texas, USA, December 1994.

[31] D. B. Judd and G. Wyszecki, *Color in Business, Science, and Industry*, John Wiley and Sons, Inc., New York City, New York, USA, 1963.

[32] A. K. Jain and F. Farrokhnia, “Unsupervised Texture Segmentation Using Gabor Filters,” *Pattern Recognition*, vol. 24, no. 12, pp. 1167-1186, December 1991.

[33] J. Lay and L. Guan, “Image Retrieval Based on Energy Histograms of the Low Frequency DCT Coefficients,” *ICASSP Proceedings*, vol. 6, pp. 3009-3012, Phoenix, Arizona, USA, March 1999.

[34] Y. Xiao, N. P. Chandrasiri, Y. Tadokoro, and M. Oda, “Recognition of Facial Expressions Using 2D DCT and Neural Network,” *Electronics and Communications in Japan*, vol. 82, no. 7, pp. 1-11, July 1999.

[35] Y. Huang and R. Chang, “Texture Features for DCT-Coded Image Retrieval and Classification,” *ICASSP Proceedings,* vol. 6, pp. 3013-3016, Phoenix, Arizona, USA, March 1999.

[36] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*, John Wiley and Sons, Inc., New York City, New York, USA, 2001.

[37] R. J. Schalkoff, *Pattern Recognition: Statistical, Structural and Neural Approaches*, John Wiley and Sons, Inc., New York City, New York, USA, 1992.

[38] J. G. Proakis and D. G. Manolakis, *Digital Signal Processing: Principles, Algorithms, and Applications*, Macmillan Publishing Company, New York City, New York, USA, 1992.

[39] T. A. Herrick and V. A. Rudis, “Visitor Preference for Forest Scenery in the Ouachita National Forest,” *Proceedings of the Symposium on Ecosystem Management Research in the Ouachita Mountains: Pretreatment Conditions and Preliminary Findings*, pp. 212-222, Hot Springs, Arkansas, USA, October 1993.

[40] H. Mo, S. Satoh, and M. Sakauchi, “Video Scene Annotation by Classification Based on Typical Scene Images,” *Proceedings of World Multiconference on Systemics, Cybernetics and Informatics*, vol. 5, pp. 223-228, Orlando, Florida, USA, July 2000.

[41] B. A. Olshausen and D. J. Field, “Vision and the Coding of Natural Images,” *American Scientist*, vol. 88, no. 3, pp. 238-245, May 2000.

[42] G. C. Lai and R. J. Figueiredo, “Image Interpretation Using Contextual Feedback,” *Proceedings of IEEE International Conference on Image Processing*, vol. 2, pp. 623-626, Washington DC, USA, October 1995.