ABSTRACT

Title of Document: IMAGE GEOLOCATION THROUGH HIERARICHAL CLASSIFICATION AND DICTIONARY-BASED RECOGNTION.

Michael William Jones, M.S., 2012

Directed By: Professor Rama Chellappa, Department of Electrical and Computer Engineering

Image geolocation, estimating GPS coordinates from an image, is a relatively new endeavor in the field of computer vision. This thesis presents two approaches to obtain the coordinates: hierarchical and dictionary-based. The hierarchical approach uses SVMs to first determine the general environment of the image and then estimates the exact location within that environment. The dictionary-based approaches are performed with linear and non-linear dictionaries using K-SVD and KK-SVD. Both methods are performed on the image feature gist and histograms of the image's color, SIFT descriptors, textons, and lines. Both the hierarchical and dictionary-based approaches build upon and combine existing systems to provide improved accuracy on a data set of twelve locations belonging to four environmental types.

IMAGE GEOLOCATION THROUGH HEIRARCHICAL CLASSIFICATION AND DICTIONARY-BASED RECOGNTION

By

Michael William Jones

Thesis submitted to the Faculty of the Graduate School of the
University of Maryland, College Park, in partial fulfillment
of the requirements for the degree of
Masters of Science
2012

Advisory Committee:
Professor Rama Chellappa, Chair
Professor Larry Davis
Associate Professor Jonathan Simon

Acknowledgements

I would like to thank my advisor, Dr. Rama Chellappa for his guidance and support throughout my time at the University of Maryland. I am also grateful to Dr. Jonathan Simon and Dr. Larry Davis for their advice and for being on my defense committee. For her assistance with technical issues that arose in the course of this project, I would like to thank Priyanka Vageeswaren. Finally, I would like to express my gratitude towards my parents for their love and support. Thank you for giving me a chance to improve myself throughout all my walks of life.

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

Image geolocation, estimating a global position system (GPS) coordinates from a single image, is a relatively new endeavor in the field of computer vision. Solutions have recently become possible due to the availability of large data sets from photo sharing websites like Flickr, where users can upload personal pictures to the internet and tag the image with the latitude and longitude where it was taken. These websites give permission to developers to download images, which provides millions of images to train and test different solutions.

In this introduction the problem of image geolocation is first formulated. Next, two previously existing systems are examined. An overview of the designed system that was constructed is discussed. Lastly, an outline of the remainder of this thesis is given.

1.1 Problem Formulation

Image geolocation seeks to provide an estimate of latitude-longitude coordinates of a digital image based only on the image content. In the past, these coordinates have been estimated by gathering thousands to millions of images and training a system by comparing test images to images with known GPS coordinates. Additionally, because there are an infinite number of ways a photograph can be taken in any location on Earth, there must be constraints on the desired precision to make the size of the problem reasonable.

The desired precision of the estimate of the photographs' locations can be attained at different levels. For example, the exact coordinates of where an image is

taken can be estimated down to a very small range, such as a specific landmark, or to a wider range, the scale of a city or national park. Because there are many different ways a photo can be taken of a single landmark due to varying conditions (daylight, night), numerous pictures are required to provide a system with a complete representation of the landmark. Similarly, a city or national park can be thought of as a collection of landmarks, requiring even more images to gain a complete representation of the larger area.

For this thesis, the city/national park level of accuracy was estimated. This estimation allowed testing of a variety of locations, ensuring the system worked in different environments. It will be shown later that a finer estimate of location can be added to provide landmark scale accuracy. In addition to the city/national park level of accuracy, only twelve different locations in the United States were chosen.

Several constraints were placed on the images. The first constraint was that only outdoor images could be used. The second was that blurry or out of focus images were eliminated. The third was that images showing very little of the surroundings, like close-ups of individuals, were not included. The fourth was that images with incorrectly tagged GPS coordinates were not included. These constraints are similar to those imposed on previous experiments and are necessary to ensure that only usable images were included in the experiment [1, 2].

Another aspect of some of the photos is the presence of textual tags. These textual tags are added by the photographer to describe the object in the image, like where the image was taken or what type of camera was used. For this experiment,

these textual tags were ignored so that only non-textual image data influenced the system.

1.2 Previous Work

Two previous systems have been created to perform image geolocation.

1.2.1 IM2GPS

One of the earliest systems that attempted to perform image geolocation was developed by Hayes and Effros [1]. Their system began with an initial gathering of 6 million GPS-tagged images from the website Flickr. From each image they extracted the following set of features:

1. 16 x 16 pixel color images
2. Color Histogram
3. Texton Histogram [3, 4, 5]
4. Line Histogram [6, 7]
5. Gist Descriptor [8]
6. Geometric Context

These features, which will be discussed in Chapter 2, were concatenated together to form a long vector and stored with the GPS coordinates of the image in a database.

Following the extraction of features for each test image, K-nearest neighbors was used to find K images in the database. Different distance metrics were used for each feature. Finally, mean-shift was performed on the GPS data of the K images to estimate a GPS coordinate for the image.

Overall, this approach yielded an accuracy of about 15%, where an accurate estimate is a test image whose GPS coordinate has been estimated to be within 200

km of its true location. When compared to random chance, guessing a random latitude and longitude point on the globe, their estimate performed around 30 times better. In addition to their experiments, they determined that the 16 x 16 pixel color images and geometric context features were not geographically discriminative. Furthermore, they determined that the size of their database greatly impacted the accuracy of their system, with an accuracy of around 1% for a database of 900 images and 16% for a database of 6.3 million images [1].

1.2.2 Mapping the World's Photos

The second system developed in 2009 by Crandel et al. Developed after the system described in "IM2GPS," their system took an alternate approach and utilized support vector machines (SVM) instead of the nearest neighbors. Their system provided a useful backbone to inspire the work done in this thesis.

Their system begins with mean shift being performed on image GPS data to find locations with many images. They used a set of 35 million pictures downloaded from Flickr. From their paper it is unclear whether any manual processing was done to remove useless pictures from their dataset. After mean shift was performed, they selected the K largest clusters to become the classes used in SVM and ignored the pictures not belonging to these clusters. Following class selection, the shift invariant feature transform (SIFT) was performed on each image [9]. The outputs of SIFT were quantized into 1000 keywords using K-means. Next, a 1000-dimensional histogram was created to indicate how often each of the keywords appeared. Finally, a SVM was created on the SIFT histograms for each of the K clusters. The SVM indicated whether a test image did or did not belong to the cluster associated with that SVM.

It is important to know that in addition to using the visual features from SIFT, textual tags were also used in their algorithm. However, despite their use, they also included results using SIFT features alone. In addition, it is significant that this system was only implemented on city pictures.

Their system was implemented on both the landmark and city scale. They found that on the city scale, recognition was around 40% for the ten largest clusters. In a 25-way landmark test in the top ten clusters, the recognition was 23.56% or 5.9 times better than chance. In a 50-way landmark test, the recognition was 14.40% or 7.2 times better than chance. Additionally, they performed a city scale test and found that recognition was 12.72%. It is important to note that their experiment does not indicate how many cities were used, so it is not known how their recognition rate compared to chance [2].

1.3 Proposed Systems

All of the systems proposed by this thesis follow the same overall structure but differ in how classification is performed. They both begin by performing mean-shift on the GPS coordinates to group the images into clusters and provide labels for the images. While labeling occurs, a set of image features are extracted from the images and then aggregated, normalized, and combined. The image features and their labels are used to form classifiers or dictionaries that are used to predict the GPS coordinates of a test image based upon its image features. This process is shown in Figure 1.1.

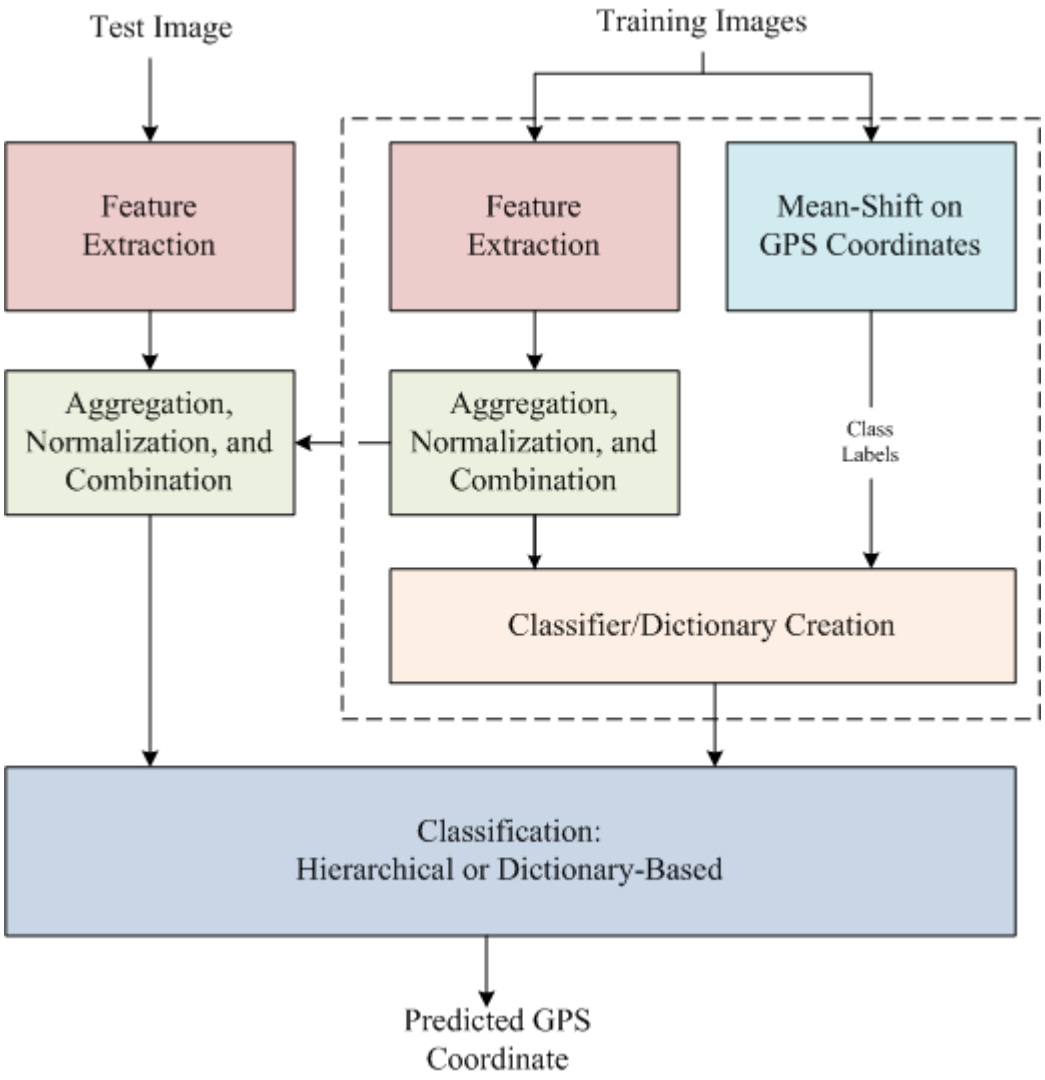

Figure 1.1 High level diagram illustrating the overall structure of the hierarchical classification and dictionary-based recognition methods.

1.3.1 Hierarchical Classification Based System

The basic idea behind the first system proposed in this thesis is relatively simple: the pictures are first sorted by their environmental type and then sorted again by locations within that environmental type. For example, if the system encountered an image from Death Valley National Park, a set of classifiers would classify the image as a desert image. After being classified as a desert image, another set of

classifiers representing desert locations would then determine that the image was an image from Death Valley. It is possible to create many different environmental classifiers, such as desert, coastal, city, forest, mountaintop, river, and plain. For this experiment, the environmental classifiers chosen were desert, coastal, forest, and city. For each environmental classifier, three locations were chosen. These locations are shown in Table 1.1.

Table 1.1 List of environmental classifiers and locations.

Environmental Classifiers	Locations
Desert	Grand Canyon National Park
	Death Valley National Park
	Bryce Canyon National Park
Coast	Acadia National Park
	Cannon Beach, Oregon
	Outer Banks, North Carolina
Forest	Yellowstone National Park
	Olympic National Park
	Shenandoah National Park
City	New York
	San Francisco
	New Orleans

The hierarchical structure created by classifying the images by their environmental type allows for finer differences in appearance to be determined between locations of the same environmental type. This occurs for two separate reasons. The first reason is that when the features for the images are obtained, vector quantization is often necessary to create a histogram of the information. Through K-means, K vectors are created to represent the vectors. When K-means is only

performed on feature vectors from desert images, a set of K vectors representing desert features are created. This allows the system to exclude vectors from the city, forest and coast when determining which vectors best represent the desert features. The second reason is that when the location classifiers are created, they are trained on only locations of that type. This allows for more focused classifiers that can ignore the influence of images from locations not belonging to their environment.

The disadvantage of using a hierarchical structure is that if the environmental classifiers perform poorly, the entire system will suffer. This poor performance is due to errors propagating through the system attributable to wrong initial choices. For example, if a forest image is incorrectly labeled as a desert image, nothing can be done to estimate its true location. Therefore, it is imperative that the initial classifiers are very accurate.

The hierarchical classification based system's classifiers were created using SVMs. The specifics of this method are explained in Chapter 3.

1.3.2 Dictionary-Based Recognition System

The second proposed system was built upon the dictionary-based recognition system proposed by Ngyuen et al. in [10]. Instead of using a hierarchical approach, pictures were only sorted by their location. Using the same twelve locations specified in Table 1.1, a location specific dictionary was created. The location specific dictionaries were then used to create an approximation of a test image using a sparse combination of atoms. The location specific dictionary producing the approximation with the lowest reconstruction error was then chosen as the predicted location.

While this system could be extended to follow a hierarchical structure, preliminary evaluations determined that the same hierarchical structure used in Section 1.3.1 did not significantly improve accuracy. Additionally, the long computation times required to create dictionaries consisting of many locations using cross-validation and a parameter search rendered a hierarchical structure impractical. However, despite not using a hierarchical structure, by combining results across locations of the same environment, the environmental classification of the dictionary-based system was comparable to the hierarchical classification system.

The specifics of the location specific dictionaries are explained in Chapter 4.

1.3.3 Image Features

For the both systems, the following features were extracted from the images:

- Gist Descriptor
- SIFT Histogram
- RGB Histogram
- Texton Histogram
- Line Histogram

These features, which will be discussed in Chapter 2, were concatenated together to form a long vector for each image. Following the extraction of features, classifiers were trained for the environmental classifiers and location classifiers using the SVMs and non-linear class specific dictionaries.

1.4 Outline of Thesis

The remainder of this thesis is organized as follows. Chapter 2 discusses the features and processing done on the images. Chapter 3 covers the hierarchical-based approach. Chapter 4 describes the dictionary-based approach. Chapter 5 details the image dataset that was collected. Chapter 6 covers the experimental results. Lastly, Chapter 7 presents the conclusions and Chapter 8 suggests future work.

Chapter 2: Image Features

2.1 Pre-Processing

All images were initially processed to reduce their dimensions. Images were resized so that their minimum dimension, either the height or width, was a maximum of 700 pixels. This step was done to reduce the amount of computations required to compute the features and reduce the storage space. This resizing is justifiable because both of the previous systems, "IM2GPS" and "Mapping the World's Photos," performed dimensionality reduction and found no significant reductions in accuracy [1, 2].

Dimensionality reduction was performed in MATLAB, and after pre-processing the images were stored together. Their file information and GPS locations were stored in a binary file for quick access.

2.1 RGB

The simplest features that can be extracted are red-green-blue (RGB) histograms. Since images are typically stored in three channels, either red, green, or blue, a histogram of the information could be easily taken. For each image, three, sixteen bin histograms were created, one histogram for each color. The three histograms were then concatenated together to form a 48-dimensional vector. The histograms were precomputed and stored in individual files so that a histogram of an image was computed only once.

The code for the RGB histograms was written in C++ using OpenCV.

2.2 Gist

In the same way as done in "IM2GPS", the gist descriptor developed by Olvia and Torralba was used as a feature [1, 8]. The gist descriptor is essentially what its name implies, the general gist of the scene one might understand when squinting at the image. More technically, instead of looking at individual objects that compose the image, the information describing the global shape of the image is sought. The descriptor is built by first dividing the image into a grid. Next, each section of the grid is filtered by a bank of Gabor filters at different orientations and scales. From the result of the filter bank the energy is used to represent that particular grid location, filter orientation, and scale.

The gist descriptors used in the experiment were constructed by first resizing the image to a 256 by 256 pixel image. Then, the image was divided into a six by six grid. In each section of the grid, the locations were filtered by a bank of Gabor filters with eight orientations and four scales, producing a 1152-dimensional vector.

The gist descriptor was implemented through a combination of C++ and MATLAB. The MATLAB code used to compute the descriptor was taken from Olivia's website and compiled into an executable file to be accessed by C++ [8]. The gist descriptor for each image was stored in a binary file so it only needed to be computed once.

2.3 SIFT

In the same way as the system Crandel et al. designed a histogram of scale-invariant feature transform (SIFT) descriptors was created for each image [2, 9]. This image feature takes the opposite approach of gist. Instead of taking a global scene-level look at the image like gist, a histogram of SIFT descriptors represents the image as a collection of important points.

SIFT was developed in 1999 by Lowe [9]. In the algorithm, keypoints in an image are extracted, and a 128-dimensional feature vector is used to represent the information at that keypoint. The keypoints are found by filtering the image through a bank of Laplacian of Gaussian filters at different scales. From these filtered versions, extrema are located in both the spatial and scale dimensions and identified as points of interest. Next, points in areas of low contrast and along edges are removed. The remaining points are labeled as keypoints, and a descriptor is built for each keypoint that is invariant to location, scale, and rotation.

A histogram was created for each image by first representing its particular descriptors by a collection of 1000 representative descriptors. This was accomplished by sampling 40,000 descriptors from the training images involved in a particular classifier. Then, K-means++ was used to find 1000 representative descriptors for the entire set of 40,000 descriptors [11]. Next, a 1000-dimensional vector was created to represent the image where each bin represent one of the 1000 descriptors computed through K-means. The bin's value corresponds to how many descriptors in the image are closest to the bin's descriptor. Following this step, the histogram was normalized

by its L^1 norm so that images of smaller dimensions with less keypoints were not underrepresented.

The SIFT descriptors were implemented in C++ using OpenCV. The descriptors for each image were stored in a binary file so the individual SIFT descriptors only needed to be computed once.

2.4 Texton

Using the same method as in"IM2GPS," a texton histogram was used as an additional image feature. This image feature is intended to provide a representation of the kinds of different textures present in an image. Potentially, this feature is useful for distinguishing between images with different types of foliage, building material, and types of rocks.

The term texton was first introduced by Julesz and later used as a vector by Leung and Malik [3, 4, 5]. To compute the textons of an image, the image is filtered by two banks of filters: Gaussian second derivative filters and Hilbert transforms of the Gaussian second derivative filters. Each of the filters had six orientations, two scales, and two elongations. Next, each pixel in the image was represented by a 48-dimensional vector where each dimension represented the output of a particular filter, orientation, scale, and elongation. Once the textons were computed, a histogram was created in the same way as the SIFT histogram. 100,000 textons were randomly sampled from the collection of training images associated with a particular classifier or set of dictionaries, and K-means++ was performed to select a set of 512 representative textures that were used to represent the textons in an individual image [11].

The texton descriptors were implemented in C++ and MATLAB. The MATLAB code to compute the individual textons was taken from the publicly available code on David Martin's website and was compiled into an executable file to be accessed by C++ [5]. To reduce computations, a maximum of 20,000 randomly sampled textons were created and stored in a binary file.

2.5 Lines

The final image features used to represent the images were histograms of line lengths and angles. The relative lengths of lines between manmade and natural images were helpful in distinguishing between these two types of images. These histograms were created in the same way as done in "IM2GPS:" through the use of the method described in "Video Compass" [1, 6].

To compute the line lengths and angles in an image, the image derivatives are calculated, followed by Canny Edge detection for non-maximum suppression. Next, the gradient direction is quantized into eight ranges and all edge pixels are labeled according to these ranges. Then, connected edges with the same label are grouped together to form a line support region. For each line support region, the eigenvalues and eigenvectors are calculated from the scatter matrix of the pixel coordinates. Finally, the line length and angle are determined by Eq. 2.1 and Eq. 2.2.

$$\theta = \text{atan}\big(e_1(2), e_1(1)\big) \tag{2.1}$$

$$\rho = \bar{x}\cos\theta + \bar{y}\sin\ \theta \tag{2.2}$$

In these equations, $\bar{x}$ and $\bar{y}$ are the mid-points in the line segment. Once all of the line parameters for an image were computed, 4000 line lengths were randomly sampled from the collection of training images associated with a particular classifier

or set of dictionaries. K-means++ was then performed on the 4000 line lengths to select 50 representative line lengths that were used to represent the line lengths in an individual image [11]. The same process was performed for the line angles so that two 50-dimensional vectors were created for each image.

The line descriptors were implemented in C++ and MATLAB. The MATLAB code used to compute the individual line lengths and angles was taken from the publically available code on Li's website and was compiled into an executable file to be accessed by C++ [7]. To reduce computations, the lines were calculated and stored in a binary file.

Chapter 3: Support Vector Machines Method

3.1 Support Vector Machines

Support vector machines (SVM) are commonly used to create a linear discriminant function to classify data. The major difference between SVM and other discriminant functions, like Fisher's linear discriminant analysis (LDA), is that SVM has the capacity to map the data into a higher dimension and construct a hyper plane for this dimension to separate the data. In other words, SVM is a linear classifier in a higher dimension than the original data. This difference allows for SVM to create curved decision boundaries if they are realized in the lower dimension, giving SVM more flexibility and improved accuracy.

3.1.1 Problem Formulation

SVMs follow a common pattern recognition problem formulation:

- Images are organized into a set of N_1 training vectors and N_2 testing vectors, of dimension d.
- The training vectors are written as $\{x_1, x_2, x_3, \dots, x_{N_1}\}$ and the set of testing vectors are written as $\{t_1, t_2, t_3, \dots, t_{N_2}\}$.
- The vectors belong to one of two classes, D_{-1} or D_{+1}.

With the N_1 training vectors, a hyperplane is created to separate the classes. Figure 3.1 is an example.

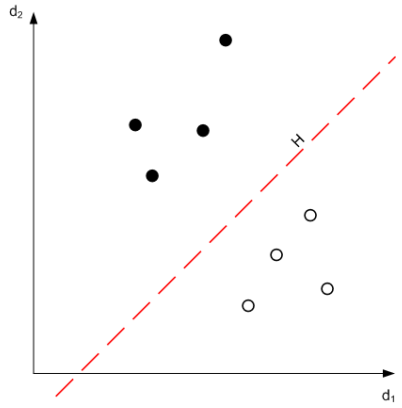

Figure 3.1 Depiction of hyperplane separating two classes in two dimensions. The classes are marked as either filled or empty circles, and the hyperplane is marked as a dotted line.

3.1.2 Linear Support Vector Machines

SVM seeks to find a solution to Eq. 3.1:

$$D_i(\boldsymbol{x}_i \cdot \boldsymbol{w} - \boldsymbol{b}) - 1 \geq 0 \ \ \forall i \qquad (3.1)$$

Where $D_i \in \{-1,1\}$, $\boldsymbol{w}$ is the normal vector to the hyperplane, and $\frac{\boldsymbol{b}}{\|\boldsymbol{w}\|}$ determines the offset of the hyperplane from the origin. When the data is linearly separable, SVM seeks to maximize the distance of the values $\boldsymbol{x}_i$ to the hyperplane, which is equivalent to maximizing $1/\|\boldsymbol{w}\|$ and minimizing $\|w\|^2$.

Next, the problem can be formulated with Lagrange multipliers α_i, with the constraint $\alpha_i \geq 0$.

$$L_P = \frac{1}{2}\|\boldsymbol{w}\|^2 - \sum_{i=1}^{N_1} \alpha_i D_i(x_i \cdot \boldsymbol{w} - \boldsymbol{b}) + \sum_{i=1}^{N_1} \alpha_i \qquad (3.2)$$

Since the objective function is a quadratic function, L_P can be maximized with the constraint that the gradient of L_P, with respect to $\boldsymbol{w}$ and $\boldsymbol{b}$, vanish and $\alpha_i \geq 0$. This maximization produces the dual problem with the new constraints:

$$\boldsymbol{w} = \sum_i \alpha_i D_i \boldsymbol{x}_i \qquad (3.3)$$

$$\sum_i \alpha_i D_i = 0 \qquad (3.4)$$

Substituting Eq. 3.3 and 3.4 into Eq. 3.2 gives Eq. 3.5:

$$L_p = \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j D_i D_j x_i \cdot x_j \tag{3.5}$$

Once solved, the set of Lagrange operators α_i will be either positive numbers or zero. Operators not equal to zero will correspond to data vectors that rest on the margin of the hyperplane, termed support vectors. From Eq. 3.3 the vector $\boldsymbol{w}$ can be found. The vector $\boldsymbol{b}$ can be found through the realization $D_k(\boldsymbol{x}_i \cdot \boldsymbol{w} - \boldsymbol{b}) = 1$, for all $\boldsymbol{x}_i$ with non-zero α_i, the support vectors.

Once $\boldsymbol{w}$ and $\boldsymbol{b}$ are known, it is possible to classify the vectors $\{t_1, t_2, t_3, \ldots, t_{N_2}\}$ through Eq. 3.6 and Eq. 3.7:

$$(\boldsymbol{t}_i \cdot \boldsymbol{w} - \boldsymbol{b}) \geq 1 \tag{3.6}$$

$$(\boldsymbol{t}_i \cdot \boldsymbol{w} - \boldsymbol{b}) < 1 \tag{3.7}$$

If Eq. 3.6 is true, $\boldsymbol{t}_i$ is assigned to class +1, and if Eq. 3.7 is true, it is assigned to class -1. If more than two classes exist, it is possible to build multiple SVMs where each SVM indicates whether the $\boldsymbol{t}_i$ belongs to a particular class. If there is a conflict and multiple SVMs indicate that $\boldsymbol{t}_i$ belongs to multiple classes, the SVM producing the largest distance from the margin is chosen.

It is important to realize that the data vectors $\boldsymbol{x}_i$ are not always separable. When this is the case, it is possible to create a soft margin by assigning a regulation parameter C to allow for some vectors to be misclassified. This new formulation is shown in Eq. 3.8 and Eq. 3.9.

$$D_i(\boldsymbol{x}_i \cdot \boldsymbol{w} - \boldsymbol{b}) - 1 \geq \xi_i \quad \forall i \tag{3.8}$$

$$\min_{w,\xi,b} \left\{ \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{N_1} \xi_i \right\} \tag{3.9}$$

Where $\xi_i \geq 0$ are slack variables representing the degree of misclassification of each $\boldsymbol{x}_i$.

3.1.3 Non-Linear Support Vector Machines

SVM can be extended to allow for non-linear decision boundaries. Eq. 3.5 shows that a dot product between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ is performed. Therefore, if the original data is mapped to some higher dimensional Euclidean space H, through the mapping $\Phi: \mathbb{R}^d \rightarrow H$, the dot product in Eq. 3.5 can be replaced with the kernel given by Eq. 3.10.

$$K(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j) \tag{3.10}$$

This kernel is limited to functions that satisfy Mercer's Condition. Some commonly used kernels include polynomial, radial, and hyperbolic kernels. In this thesis, the radial basis function (RBF) was used as a kernel.

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\gamma \left\|\mathbf{x}_i - \mathbf{x}_j\right\|^2\right) \tag{3.11}$$

In the case of non-separable variables, this leaves two parameters that must be found, γ and C. To find the parameters suited to the experiment, the training data was divided into K non-overlapping subsets. Each subset was used for testing and the remaining were used for training. For each test, a quadratic grid search can be performed to find the parameters suited for the data: γ and C.

3.2 Classifier Creation

To create the data vectors $\boldsymbol{x}_i$, many different possible image feature combinations are available. Additionally, it is possible to aggregate and normalize the features. With five features, each with two to three forms of aggregation and three

forms of normalization, there are over twenty-four thousand combinations. Furthermore, performing a grid search over 30 parameter possibilities with 12-fold validation would require the training of around 8.5 million SVMs for each classifier. With the long computation times involved in creating SVMs, it is not feasible to test all of the combinations. Therefore, to construct a suitable combination of features, aggregations, and normalizations, a method similar to the method proposed by Peter Belhumeur for face recognition was implemented [11, 12]

To construct the vectors, the followings steps were taken. First, initial vectors were selected by using all aggregation and normalization combinations of a single image feature. The initial image feature chosen was the gist feature, with a total of three aggregation possibilities and three normalization possibilities. With these vectors, SVMs were trained through grid searches and cross-validation and the vector producing the highest cross-validation was selected as the initial vector. Next, the following iterations were performed.

1. An untested image feature was selected and all aggregation and normalization combinations were created. Next, these vectors were concatenated with the previous iteration or initialization, using the highest cross validation accuracy.
2. For each of the new lengthened vectors, SVMs were created through grid searches and cross-validation, and the vector that produced the highest cross-validation was chosen. If none of the current vectors produced SVMs with higher cross-validation compared to the previous iteration or initialization, the current image feature was rejected.

This iterative method reduced the number of combinations to a total of 37 or 11,880 SVMs with grid searches and cross validation for each classifier.

3.2.1 Aggregation

The features were aggregated three ways if feasible: no aggregation, a histogram of the data, and the sample mean and variance of the data. For all of the features, except for gist, performing no aggregation was impractical because the dimension of the data was too high. The histograms were created through vector quantization, and the specifics for each feature are outlined in Chapter 2. The sample mean and variance were calculated for each image feature using Eq. 3.12 and Eq. 3.13.

$$\boldsymbol{\mu} = \frac{1}{N_1}\sum_{i=1}^{N}\mathbf{y}_i \tag{3.12}$$

$$\boldsymbol{\theta} = \sqrt{\frac{1}{N_1-1}\sum_{i=1}^{N}(\mathbf{y}_i - \boldsymbol{\mu})^2} \tag{3.13}$$

Where $\mathbf{y}_i$ is an individual element of an image feature, such as a single SIFT keypoint descriptor or the output of the set of filters for one location in gist.

3.2.1 Normalization

The features were normalized three ways after aggregation: no normalization, mean normalization, and energy normalization. The mean normalization was performed according to Eq. 3.14.

$$\widehat{\boldsymbol{x}}_i = \frac{\boldsymbol{x}_i}{\boldsymbol{\mu}} \tag{3.14}$$

Where μ is calculated according to Eq. 3.12, except over all $\boldsymbol{x}_i, i = 1, \ldots, N_1$. Energy normalization is performed according to Eq. 3.15.

$$\tilde{\boldsymbol{x}}_i = \frac{\mathbf{x}_i - \mu}{\theta} \tag{3.15}$$

Where μ and θ were calculated according to Eq. 3.12 and Eq. 3.13 except over all $\boldsymbol{x}_i, i = 1, \ldots, N_1$.

3.3 Alternative Classifier Creation

An alternate method of classifier creation was used to combine the different image features. Individual SVMs were created for each image feature to make new vectors $\boldsymbol{v_i}$, where $\boldsymbol{v_i} = [v_i^1, v_i^2, \ldots, v_i^5]$. v_i^1 is the distance from the margin for the SVM created from the gist image features, and v_i^2 is the distance from the margin for the SVM created from the RGB histogram, etc. Using a separate portion of the training images, an additional SVM was trained to classify $\boldsymbol{v_i}$ and estimate the image's location or environmental type [12]. A structural diagram illustrating how this process is performed is shown in Figure 3.2.

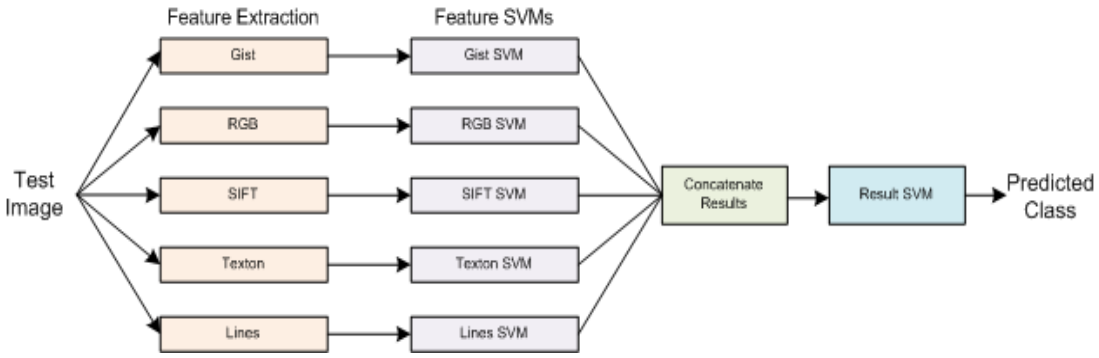

Figure 3.2 Structural diagram for an alternative classifier creation structure. Individual SVMs for each feature were created, and the results from each SVM were concatenated together and filtered through a final SVM to predict the class.

Combining the image features using this method has advantages and potential disadvantages. An advantage is that this method allows for SVM to determine the appropriate weightings for the different image features. Additionally, this method allows for different kernels for each image feature. However, a potential disadvantage

is that the dimensionality reduction enacted by this method can remove structural properties that exist between image features that can be useful in classifying the images.

3.4 Hierarchical Structure

A hierarchical structure was implemented to categorize the images. Many different categories could be used, such as time of day, items included in the image, etc. However, in this research environmental types were chosen to separate the images since the environmental type is easily recognized and correlates highly with the physical location. Four environmental types were used to divide the images: desert, forest, coast, and city. While there are many other obvious choices of images, such as mountains, suburbs, and plains, the four aforementioned types provide variety.

The system was assembled by creating an SVM for each environmental type. Each SVM was constructed from the image features from an equal number of images representing the environmental type and an equal number representing all other environmental types. Early in the testing it was observed that increasing the amount of training images increased accuracy. Therefore, the maximum numbers of training images were used for each SVM. Additionally, for each SVM, new histograms were created for the SIFT, texton, RGB, and line features, and these histograms extended to the testing data. The environmental type was predicted for each test feature by selecting the environmental type corresponding to the largest distance from the margin. A structural diagram showing how a test images environment was predicted is shown in Figure 3.3.

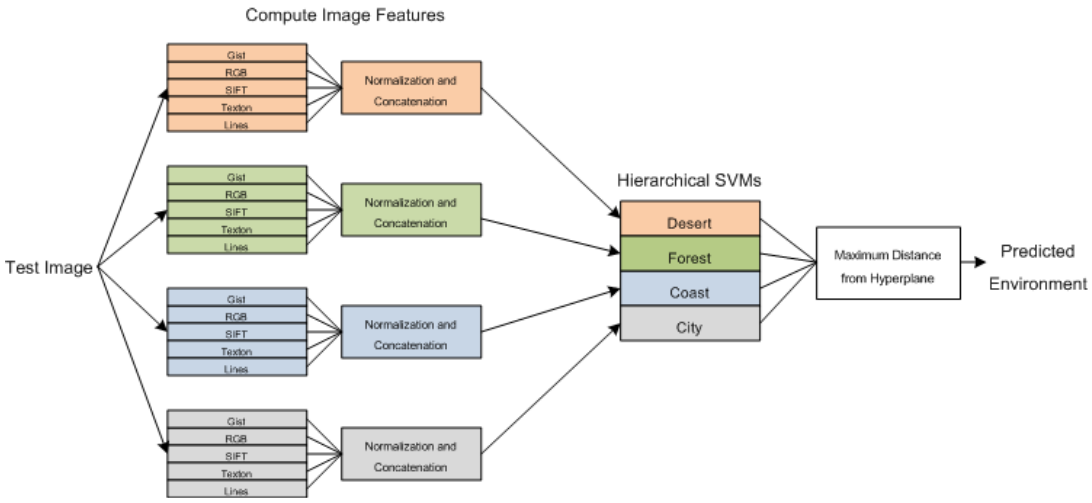

Figure 3.3 Structural diagram showing how a test images environment was predicted using the hierarchical system. Individual image features (Gist, SIFT, etc.) were only computed once; however, their histograms were computed for each environment.

Following the creation of the SVMs for the environmental types, SVMs for the locations of each environmental type were created. For example, a set of three SVMs representing three distinct desert locations were computed. Like the hierarchical SVMs, new histograms were computed for SIFT, texton, RGB, and line features. This division allowed for greater variation across the histograms of locations belonging to the same environment. In addition, new histograms were computed for the testing data for each SVM, resulting in 12 histograms each corresponding to each location's SVM. After the environmental type was predicted, the test data was then sent to the location's SVM of the predicted environmental type. The predicted location was chosen by the SVM producing the largest distance from the margin. A

structural diagram of the hierarchical structure is shown in Figure 3.4.

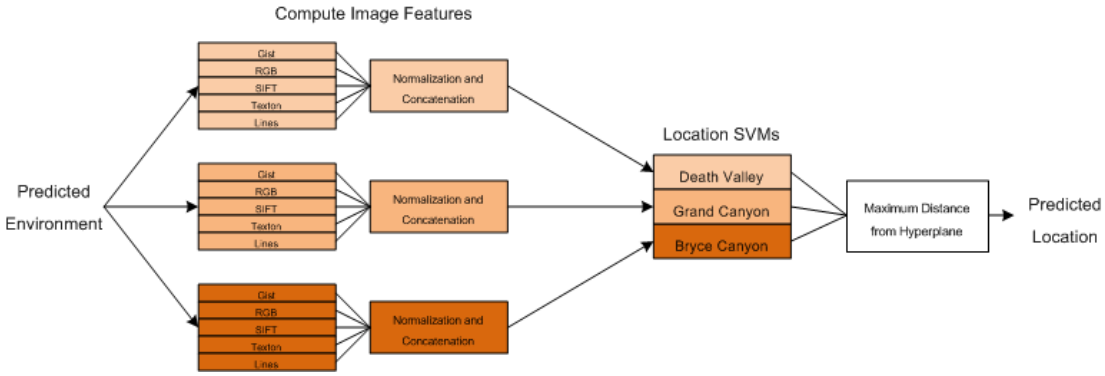

Figure 3.4 Structural diagram showing how a test image's location was predicted for the desert locations using the hierarchical system. The individual image features (Gist, SIFT, etc.) were only computed once, but their histograms were computed for each environment.

Chapter 4: Dictionary-Based Method

Dictionary-based recognition is another commonly used method to perform recognition in computer vision. In this method, a representative dictionary is learned from training data. Later, when new data is encountered, the dictionaries are used for classification by choosing the dictionary that can reconstruct the new data with the lowest reconstruction error [13]. These dictionaries can be built from the images themselves or, if there is too much variation between the images, from features extracted from the images. Additionally, the algorithm seeks to find create dictionaries that create low reconstruction error from a sparse representation vector. Spare representation vectors are sought because if an image belongs to a particular class, it should be possible to reconstruct it with only a few atoms. Like SVM, methods exist to reformulate the algorithms to use a kernel and create non-linear dictionaries which can improve classification accuracy [10].

4.1 Linear Dictionary Learning

The first dictionary-based method used for recognition was a system similar to the dictionary-based method proposed by Patel et al. in their paper "Dictionary-based Recognition Under Variable Lighting and Pose" [13]. In their paper they propose an algorithm to create class specific dictionaries for face recognition using the algorithm K-single value decomposition (K-SVD) [14].

4.1.1 Problem Formulation

The problem is formulated as follows:

- We have *C* distinct classes each with a set of m_i training images, where $i \in \{1, \cdots, C\}$.
- Each training image can have a vector of image features extracted from it to produce a *N*-dimensional vector $\boldsymbol{x}$.
- A matrix can be obtained by concatenating the vectors to produce $\mathbf{B}_i$ where:

$$\mathbf{B}_i = \left[\boldsymbol{x}_i^1, \cdots, \boldsymbol{x}_i^{m_i}\right] \in \mathbb{R}^{N \times m_i} \tag{4.1}$$

- Then, we have a test image whose true class is unknown and whose image features can be extracted to produce an *N*-dimensional vector $\boldsymbol{y}$.

4.1.2 Class Specific Dictionaries

For each matrix $\mathbf{B}_i$, we seek to find a dictionary of *K* atoms $\mathbf{D}_i \in \mathbb{R}^{N \times K}$ that can create an accurate representation of B_i using a set of sparse representation vectors $\boldsymbol{\gamma}_i^k \in \mathbb{R}^K, k \in \{1, \cdots, m_i\}$. The representation vectors can be combined to form $\boldsymbol{\Gamma}_i$ as shown in Eq. 4.2.

$$\boldsymbol{\Gamma}_i = \left[\boldsymbol{\gamma}_i^1, \cdots, \boldsymbol{\gamma}_i^{m_i}\right] \in \mathbb{R}^{K \times m_i} \tag{4.2}$$

These two matrixes are found by solving the following optimization problem seen in Eq. 4.3.

$$\left(\acute{\mathbf{D}}_i, \acute{\mathbf{\Gamma}}_i\right) = \arg\min_{D_i, \Gamma_i} \|\mathbf{B}_i - \mathbf{D}_i \mathbf{\Gamma}_i\|_F^2 \text{ s.t. } \forall k \left\|\boldsymbol{\gamma}_i^k\right\|_0 \leq T_0 \tag{4.3}$$

In Eq. 4.3, the sparsity of the representative vectors $\boldsymbol{\gamma}_i^k$ is capped through the second half of the equation where $\|.\|_0$ counts the number of non-zero elements and T_0 is the maximum amount of non-zero elements allowed.

4.1.2 K-Singular Value Decomposition (K-SVD)

An algorithm that solves Eq. 4.3 is K-Singular Value Decomposition (K-SVD) [14]. K-SVD is an iterative algorithm based upon the popular clustering algorithm K-means. The algorithm begins by initializing the dictionary with K randomly selected l_2-normalized vectors $\boldsymbol{x}$ from the set of training vectors. Next, the algorithm alternates between a sparse-coding step and a dictionary update step for a set number of iterations.

- Step 1. Sparse Coding:

  In the first step, the dictionary $\mathbf{D}_i$ is kept fixed and the optimal $\boldsymbol{\gamma}_i^k$ are found for each $\boldsymbol{x}_i^k$ in $\mathbf{B}_i$, according to Eq. 4.4.

  $$\min_{\boldsymbol{\gamma}_i^k} \left\|\boldsymbol{x}_i^k - \mathbf{D}_i \boldsymbol{\gamma}_i^k\right\|_2^2 \text{ s.t. } \forall k \left\|\boldsymbol{\gamma}_i^k\right\|_0 \leq T_0 \tag{4.4}$$

  Eq. 4.4 is solved using any pursuit algorithm such as matching pursuit (MP) and orthogonal matching pursuit (OMP).

- Step 2. Dictionary Update:

  The second step of the algorithm updates each column of $h = 1,2, \dots, K$ of $\mathbf{D}_i$ according to the following steps:

  1. First, a set of examples from $\boldsymbol{x}_i^k$ that use a particular atom $\mathbf{D}_i$ are defined. More precisely, $\omega_h = \{n | \ 1 \leq n \leq N, \ \boldsymbol{\gamma}_i^h(n) \neq 0\}$.
  2. Next, the error representation matrix $\mathbf{E}_m$ is calculated according to Eq. 4.5.

  $$\mathbf{E}_{h,i} = \mathbf{B}_i - \sum_{j \neq h} d_i^j \, \overline{\boldsymbol{\gamma}}_i^j \tag{4.5}$$

  In Eq. 4.5 $\overline{\boldsymbol{\gamma}}_i^j$ represents the *j*th row of $\boldsymbol{\Gamma}_i$.

  3. $\mathbf{E}_h$ is then restricted by only choosing the columns that correspond to ω_m to give $\mathbf{E}_{h,i}^R$.
  4. Finally, SVD is performed on $\mathbf{E}_{h,i}^R = \mathbf{U}\Delta\mathbf{V}^{\mathrm{T}}$. The column $\tilde{d}_i^h$ is set to the first column of $\mathbf{U}$, and $\overline{\boldsymbol{\gamma}}_R^m$ is the first column of $\mathbf{V}$ multiplied by $\Delta(1,1)$.

4.1.3 Image Classification

An image feature vector $\boldsymbol{y}$ with an unknown label can be classified once the C dictionaries, $\mathbf{D}_i$, are determined from K-SVD. This is done by projecting $\boldsymbol{y}$ onto the span of the atoms of $\mathbf{D}_i$, through the orthogonal projector $\mathbf{P}_i$, defined according to Eq. 4.6.

$$\boldsymbol{P}_i = \mathbf{D}_i(\mathbf{D}_i^T \mathbf{D}_i)^{-1} \mathbf{D}_i^T \tag{4.6}$$

Using $\mathbf{P}_i$ we can approximate $\boldsymbol{y}$ according to Eq. 4.7, and the residual vector $\boldsymbol{r}_i(\boldsymbol{y})$ can be calculated according to Eq. 4.8.

$$\hat{\boldsymbol{y}}_i = \boldsymbol{P}_i \boldsymbol{y} \tag{4.7}$$

$$\boldsymbol{r}_i(\boldsymbol{y}) = \boldsymbol{y} - \hat{\boldsymbol{y}}_i = (\boldsymbol{I} - \boldsymbol{P}_i)\boldsymbol{y} \tag{4.8}$$

Then, the estimated class $d \in \{1, \cdots, C\}$ is chosen by selecting the class that produces the lowest reconstruction error [13]. The selection of the estimated class d is shown seen in Eq. 4.9.

$$d = \arg\ \min_i \|\boldsymbol{r}_i(\boldsymbol{y})\|_2 \tag{4.9}$$

4.2 Kernel Dictionary Learning

Like SVM, the dictionary-based recognition scheme shown in Chapter 4.1 can be extended using the kernel trick to allow for non-linear dictionaries. The second dictionary-based method follows a strategy similar to the object recognition method introduced by Nguyen et al. in "Kernel dictionary learning" [10]. It is important to note that the non-linear method follows the same initial problem formulation shown in Chapter 4, Section 1.1, except it focuses on only creating a dictionary for one class, **D**.

The dictionary-based recognition is extended to allow for non-linear dictionaries according to Eq. 4.10. This is done by adapting Eq. 4.4 with the mapping Φ: $\mathbb{R}^N \rightarrow$ F $\subset \mathbb{R}^{\tilde{N}}$, F is the dot product space.

$$\left(\acute{\mathbf{D}}, \acute{\boldsymbol{\Gamma}}\right) = \arg\ \min_{\mathbf{D},\boldsymbol{\Gamma}} \left\|\Phi(\mathbf{B}) - \mathbf{D}\,\boldsymbol{\Gamma}\right\|_F^2\ s.t.\ \|\boldsymbol{\gamma}_k\|_0 \leq T_0\ \forall k \tag{4.10}$$

Next, we reformulate our dictionary **D** to be composed of some predefined base dictionary **B,** and an atom representation **A** shown in Eq. 4.11.

$$\mathbf{D} = \mathbf{BA} \tag{4.11}$$

From Eq. 4.11 we allow **B,** the predefined base dictionary to be $\Phi(\mathbf{B})$. It was shown in [10] that there exists an optimal solution to Eq. 4.10 using Eq. 4.12.

$$\mathbf{D} = \Phi(\mathbf{B})\mathbf{A} \tag{4.12}$$

From Eq. 4.12 we can rewrite Eq. 4.10 to produce Eq. 4.13.

$$\left(\acute{\mathbf{D}}, \acute{\mathbf{\Gamma}}\right) = \arg \min_{\mathbf{D}, \mathbf{\Gamma}} \|\Phi(\mathbf{B}) - \Phi(\mathbf{B})\mathbf{A}\,\mathbf{\Gamma}\|_F^2 \; s.t. \|\boldsymbol{\gamma}_k\|_0 \leq T_0 \; \forall k \tag{4.13}$$

Finally, using the identity $\|\mathbf{Y}\|_F^2 = \mathrm{Tr}(\mathbf{Y}^H\mathbf{Y})$ we can rewrite Eq. 4.13 as Eq. 4.14.

$$\left(\acute{\mathbf{D}}, \acute{\mathbf{\Gamma}}\right) = \arg \min_{\mathbf{D}, \mathbf{\Gamma}} \mathrm{Tr}\left((\mathbf{I} - \mathbf{A}\,\mathbf{X})^T \boldsymbol{K}(\mathbf{\Gamma}, \mathbf{\Gamma})(\mathbf{I} - \mathbf{A}\,\mathbf{X})\right) \; s.t. \; \|\boldsymbol{\gamma}_k\|_0 \leq T_0 \; \forall \mathrm{k} \tag{4.14}$$

Where $\boldsymbol{K}(\mathbf{\Gamma}, \mathbf{\Gamma})$ is the kernel matrix $k(\mathrm{i}, \mathrm{j}) = \Phi(\boldsymbol{x_i})^{\mathrm{T}}\Phi(x_{\mathrm{j}})$. Since $\boldsymbol{K}$ only requires dot products, Mercer kernel functions can be used like the RBF kernel used in Chapter 3.

4.1.2 Kernel KSVD (KKSVD)

An algorithm that can solve Eq. 4.13 is Kernel KSVD (KKSVD). Developed by Ngyuen et al., it follows the same sparse-coding dictionary and dictionary update process as KSVD but incorporates the kernel matrix **K** to create non-linear dictionaries [10]. The algorithm begins by initializing the matrix **A** of Eq. 4.12 by randomly selecting one element from each column to be 1 and normalizing each column of **D** to unit norm. Next, the algorithm alternates between a sparse-coding step and a dictionary update step for a set number of iterations or until some other stopping criteria is met.

- Step 1. Sparse Coding

  In the first step, the matrix **A** of the dictionary is kept fixed and the sparse coefficient matrix **Γ** is found through the algorithm kernel orthogonal

matching pursuit (KOMP). This is accomplished by reformulating Eq. 4.10 to become N smaller problems as shown in Eq. 4.15.

$$\min_{\boldsymbol{\gamma}_k} \left\| \Phi(\mathbf{B}) - \Phi(\mathbf{B})\mathbf{A}\,\boldsymbol{\gamma}_k \right\|_F^2 \;\; s.t. \;\; \|\boldsymbol{\gamma}_k\|_0 \leq T_0 \;\forall k \qquad (4.15)$$

Given the matrix $\mathbf{A}$, $\Phi(\mathbf{B})$, and a signal $\boldsymbol{x} \in \mathbb{R}^N$, KOMP seeks to find a sparse combination of dictionary atoms that represent $\boldsymbol{x}$ in the feature space.

$$\Phi(\mathbf{x}) = \Phi(\mathbf{B})\hat{\boldsymbol{x}}_s + \boldsymbol{r}_s \qquad (4.16)$$

Where $\hat{\boldsymbol{x}}_s$ is the current representation of the signal $\Phi(\mathbf{x})$ and $\boldsymbol{r}_s$ is the current residual.

1. The first step of KOMP projects the residual vector $\boldsymbol{r}_s$ onto the dictionary atoms belonging to the set of atoms not belonging to I_s, the current set of atoms that have been selected shown in Eq. 4.17.

$$\begin{aligned} \boldsymbol{r}_s^T(\Phi(\mathbf{B})\boldsymbol{a}_i) &= \left(\Phi(\mathbf{x}) - \Phi(\mathbf{B})\hat{\boldsymbol{x}}_s\right)^T(\Phi(\mathbf{B})\mathbf{a}_\mathrm{i}) \\ &= (\boldsymbol{K}(\boldsymbol{z},\mathbf{B}) - \hat{\boldsymbol{x}}_s^t\boldsymbol{K}(\mathbf{B},\mathbf{B}))\mathbf{a}_\mathrm{i} \end{aligned} \qquad (4.17)$$

   Where $\boldsymbol{a}_i$ is the i-th column of $\mathbf{A}$, and $\boldsymbol{K}(\boldsymbol{z},\mathbf{B})$ is defined according to Eq. 4.18.

$$\boldsymbol{K}(\boldsymbol{z},\mathbf{B}) = [k(\boldsymbol{z},\boldsymbol{x_1}), k(\boldsymbol{z},\boldsymbol{x_2}),\cdots, k(\boldsymbol{z},\boldsymbol{x_\mathrm{n}})]$$

2. Then, the algorithm selects a new dictionary atom not belonging to the set I_s that gives the largest projection coefficient which guarantees the largest reduction in the approximation error.
3. Following this step, a new $\boldsymbol{\gamma}_s$ is created by projecting the signal $\Phi(\mathbf{x})$ onto the subspace spanned by the selected dictionary atoms $\Phi(\mathbf{B})\mathbf{A}_{I_s}$ using the Moore-Penrose pseudoinverse. $\mathbf{A}_{I_s}$ is the set of dictionary

atoms with set indices from I_s.

$$\boldsymbol{\gamma}_s = \left(\left(\Phi(\mathbf{B})\mathbf{A}_{I_s}\right)^T\left(\Phi(\mathbf{B})\mathbf{A}_{I_s}\right)\right)^{-1}\left(\Phi(\mathbf{B})\mathbf{A}_{I_s}\right)^T\Phi(\mathbf{x}) \quad (4.18)$$

4. Finally, a new representation $\widehat{\boldsymbol{x}}_s$ is found from Eq. 4.19.

$$\widehat{\boldsymbol{x}}_s = \mathbf{A}_{I_s}\boldsymbol{\gamma}_s \quad (4.19)$$

Steps one through four are then repeated T_0 times.

- Step 2. Dictionary Update

The second step of the algorithm is the updating of the dictionary **D**, particularly the matrix **A**. The approach begins by fixing **Γ** in Eq. 4.13 and rearranging the equation to produce to Eq. 4.20.

$$\left\|\Phi(\mathbf{B})\mathbf{E}_k - \Phi(\mathbf{B})\mathbf{M}_k\right\|_F^2 \quad (4.20)$$

Where $\mathbf{E}_k$ and $\mathbf{M}_k$ are defined according to Eq. 4.21 and Eq. 4.22 respectively.

$$\mathbf{E}_k = \left(\mathbf{I} - \sum_{j\neq k}\boldsymbol{a}_j\boldsymbol{\gamma}_T^j\right) \quad (4.21)$$

$$\mathbf{M}_k = \left(\boldsymbol{a}_k\boldsymbol{\gamma}_T^k\right) \quad (4.22)$$

Where $\boldsymbol{a}_k$is a column of **A**. Next, the group of indices ω_k that correspond to examples that use atoms of $(\Phi(\mathbf{B})\mathbf{A})_k$ are found according to Eq. 4.23.

$$\omega_k = \left\{i | 1 \leq i \leq n, \boldsymbol{\gamma}_T^k \neq 0\right\} \quad (4.23)$$

From $\omega_k$it is possible to make a matrix Ω_k of size $n\ x\ |\omega_k|$, where ones are on $(\omega_k(i), i)$ entries and zeros everywhere else. From this result we can define column-reduced matrixes $\mathbf{E}_k^R = \mathbf{E}_k\Omega_k$ and $\mathbf{M}_k^R = \mathbf{M}_k\Omega_k$ to produce Eq. 4.24.

$$\left\|\Phi(\mathbf{B})\mathbf{E}_k^R - \Phi(\mathbf{B})\boldsymbol{a}_k\boldsymbol{\gamma}_R^k\right\|_F^2 \quad (4.24)$$

Following this step, SVD can be performed to produce the equality 4.25.

$$(\mathbf{E}_k^R)^T K(\mathbf{B}, \mathbf{B})(\mathbf{E}_k^R) \;=\; \mathbf{V}\,\Delta\,\mathbf{V}^T \tag{4.25}$$

Where $\mathbf{\Delta} = \mathbf{\Sigma}\mathbf{\Sigma}$ and $\Phi(\mathbf{B})\mathbf{E}_k^R \;=\; \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$. Finally, it is possible to update $\boldsymbol{a}_k$ according to Eq. 4.26.

$$\widehat{\boldsymbol{a}}_k \;=\; \sigma_1^{-1}\mathbf{E}_k^R\mathbf{v}_1 \tag{4.26}$$

Where $\mathbf{v}_1$ is the first column of $\mathbf{V}$ and $\sigma_1 = \sqrt{\mathbf{\Delta}(1,1)}$.

4.1.3 Image Classification

An image feature vector $\boldsymbol{y}$ with an unknown label can be classified once the C dictionaries, $\boldsymbol{D_i} = \Phi(\boldsymbol{B_i})\boldsymbol{A}_i$, are determined from KKSVD [10]. For each test sample, the reconstruction error a dictionary would produce can be attained through Eq. 4.27.

$$r_i = \left\|\Phi(\mathbf{y}) - \Phi(\boldsymbol{B_i})\boldsymbol{A}_i\boldsymbol{\gamma}\right\|_2^2 \tag{4.27}$$

In Eq. 4.27, $\boldsymbol{\gamma}$ is calculated using KOMP, as described in Section 4.1.2. Once all reconstruction errors have been calculated, the class d producing the lowest reconstruction error is chosen.

$$d = \arg\,\min_i \boldsymbol{r}_i(\boldsymbol{y}) \tag{4.28}$$

4.3 System Structure of Dictionary-Based Method

Once the dictionaries, $\boldsymbol{D_i}$, for each class of the C locations are created, a new image from an unknown location is processed as follows:

1. Features Extraction

   The first step of the geolocation process is the extraction of image features. The image feature gist and histograms of RGB data, SIFT descriptors, textons, and lines are all energy normalized using the sample mean and variance

parameters taken from the training features. Following normalization, the image features are concatenated together to form a 2762-dimensional vector $\boldsymbol{y}$.

2. Reconstruction Error

   Following feature extraction, the reconstruction error r_i is calculated using each $\boldsymbol{D_i}$, according to Eq. 4.8 for the linear dictionary-based method, and Eq. 4.28 for the non-linear dictionary-based method.

3. Location Estimation

   Finally, once all reconstruction errors are calculated, the location corresponding to the dictionary with the lowest reconstruction error is selected.

This process is summarized by Figure 4.1.

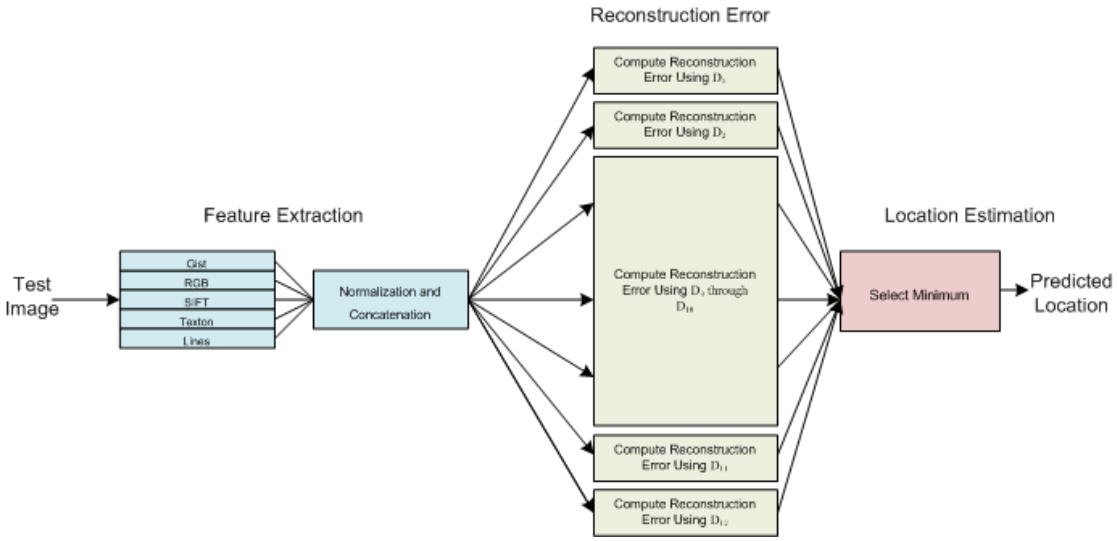

Figure 4.1 Structural diagram showing how a test images location was predicted using the dictionary-based system.

Chapter 5: Image Dataset

5.1 Flickr

To acquire images for this thesis, geotagged images from the website Flickr were downloaded. This was accomplished by utilizing and modifying MATLAB and Python code developed by Hays [1]. The code takes as an input a list of positive tags, such as "YellowstoneNationalPark," and a list of tags it would like to exclude, such as "People", "Party", and "Wedding". Then, a Python script utilized Flickr's API and searched their server for geotagged images that contained any of the positive tags and excluded images with any of the negative tags. Once a list of potential images was found, a MATLAB script attempted to download the images from the list. Certain images could not be downloaded because the authors of the images had placed restrictions on who could download their images. Finally, MATLAB stored all of the extra image data, including the geolocation, in the images files EXIF data.

5.2 Image Screening

After the images were acquired, a script was used to manually screen the acquired images. The user was given the option to accept or reject an image. This was a time intensive process that proved necessary because a large portion of the downloaded images were either incorrectly tagged or not usable. For example, the images from San Francisco contained a few thousand blurry images taken at night of a large outdoor pillow fight. While the event looked exciting, there was little content in the image that could be used to identify where the images were taken.

There were many different reasons images were considered to be not usable. Typically images were rejected because the image was a portrait of a person and the persons face and body took up more than approximately 75% of the image. Other common reasons an image was rejected were because the image was taken indoors, the image was blurry or over-processed with a computer program like Photoshop, or the image was of an event like a wedding or car show. It is important to note that while the Python script used to find the geotagged images rejected images with tags like "Wedding," this required the image to have this tag to begin with. Unfortunately, it seemed that many people would tag their images with the name of the city and leave out tags that were helpful in filtering. Examples of rejected and accepted images are shown in Figure 5.1. The top row contains examples of images that were rejected (graffiti, food, a duck), and the bottom row contains examples of images that were accepted (buildings, canyon, bridge).

Figure 5.1 Examples of rejected images and accepted images. Rejected images are on the top row, accepted images on the bottom.

Finally, once the images had been screened, the accepted images were separated from the rejected images. This produced a total of 22,834 acceptable images out of a total of 88,809 images. Consequently, only 25% of the images downloaded from Flickr were used for this thesis.

5.3 Image Categories

For this thesis, four environmental types were chosen: desert, coast, forest, and city. There are other obvious environmental types that could be chosen, like mountain tops, grasslands, farmlands, wetlands, and underwater. However, due to the amount of time it takes to screen images and images from these environments being less available, only the aforementioned four types were chosen. Additionally, it is important to note that the four chosen types could be further subdivided into more precise types, like desert canyons, evergreen forest or deciduous forest to add additional categories.

5.3.1 Desert

Three locations from different desert locations were chosen: Grand Canyon National Park, Death Valley National Park, and Bryce Canyon National Park. After screening, 2635 photos were found for Grand Canyon National Park, 1565 photos were found for Death Valley National Park, and 2020 photos were found for Bryce Canyon National Park. Figure 5.2 demonstrates some of the images of these locations.

Figure 5.2 Example images of the three desert locations. From left to right: Grand Canyon, Death Valley, and Bryce Canyon National Parks.

5.3.2 Coast

Three locations from different coastal locations were chosen: Acadia National Park, Maine; Cannon Beach, Oregon; and the Outer Banks, North Carolina. After screening, 1348 photos were found for Acadia National Park, 1365 photos were found for Cannon Beach, and 2014 photos were found for the Outer Banks. Figure 5.3 demonstrates some of the images of these locations.

Figure 5.3 Example images of the three coastal locations. From left to right: Acadia National Park, Maine; Canon Beach, Oregon; the Outer Banks, North Carolina.

5.3.3 Forest

Three locations from different forest locations were chosen: Yellowstone National Park; Olympic National Park; and the Shenandoah National Park. After screening, 1737 photos were found for Yellowstone National Park, 1997 photos were found for Olympic National Park, and 2782 photos were found for the Shenandoah National Park. Figure 5.4 demonstrates some of the images of these locations.

Figure 5.4: Example images of the three forest locations. From left to right: Yellowstone, Olympic, and Shenandoah National Parks.

5.3.4 City

Three locations from different cities were chosen: New York, New York; San Francisco, California; and the New Orleans, Louisiana. After screening, 2225 photos were found for New York, 1365 photos were found for San Francisco, and 1681 photos were found for the New Orleans. Figure 5.5 demonstrates some of the images of these locations.

Figure 5.5 Example images of the three cities. From left to right, New York, San Francisco, and New Orleans.

5.4 Mean-Shift

While twelve geographically distinct locations were chosen for this thesis, the mean-shift clustering algorithm was performed on the GPS data of the images [11]. This step was done to simulate situations where the most photographed locations might not be as known. Additionally, mean-shift was necessary to remove any images that were tagged as being from a location but their GPS location indicated they were

from somewhere else. For example, an image could be tagged "deathvalley" with a GPS tag from Las Vegas.

Mean-shift was used to estimate the modes of the GPS data. Like "Mapping the World's Photos," mean-shift was chosen because it does not require any prior knowledge of the number of clusters and instead uses a bandwidth parameter to control the resolution of the clusters. For example, the bandwidth parameter can be decreased so that the locations of highly photographed landmarks can be determined, or the bandwidth can be increased so that the locations of highly photographed cities and national parks can be determined.

Mean-shift is an iterative algorithm that uses the gradient estimate to update itself, stopping when it reaches a mode, a zero gradient. The algorithm begins by selecting an initial GPS location, x, and then calculates $m_h(x)$ according to Eq. 5.1.

$$m_h(x) = \frac{\sum_{i=1}^{n} x_i g(\left\|\frac{x-x_i}{h}\right\|)}{\sum_{i=1}^{n} g(\left\|\frac{x-x_i}{h}\right\|)} - x \qquad (5.1)$$

Where x_i are the data values, g is the weight which corresponds to a kernel function, and h is the bandwidth. Next, the procedure updates x using Eq. 5.2.

$$x^{(t+1)} = x^{(t)} + m_h\left(x^{(t)}\right) \qquad (5.2)$$

For each initial location, the algorithm is run for a fixed number of iterations or runs until the changes in x are negligible. For this thesis, a maximum of 200 iterations were performed, and the bandwidth parameter *h* was set to 0.6. It is important to note that within the uniform kernel, distance was computed between latitude longitude points instead of performing a distance calculation through the Haversine or Vincenty's formula. This decision was made because the resolution of the clusters was relatively small compared to the size of the earth and none of the locations were

close to the North or South Poles. Additionally, the algorithm was sped up by bucketing the images and only updating the gradient with images from neighbor bins.

Many initial location seeds were performed to find the twelve largest modes of the experiment. As expected, these modes directly corresponded to the twelve locations listed earlier in this chapter. Once the twelve largest modes were discovered, all of the downloaded images were labeled according to the mode they were closest to. However, if an image was over 200 km away from a mode, it was removed from the experiment. An image showing the GPS locations of the collected images can be seen in Figure 5.6.

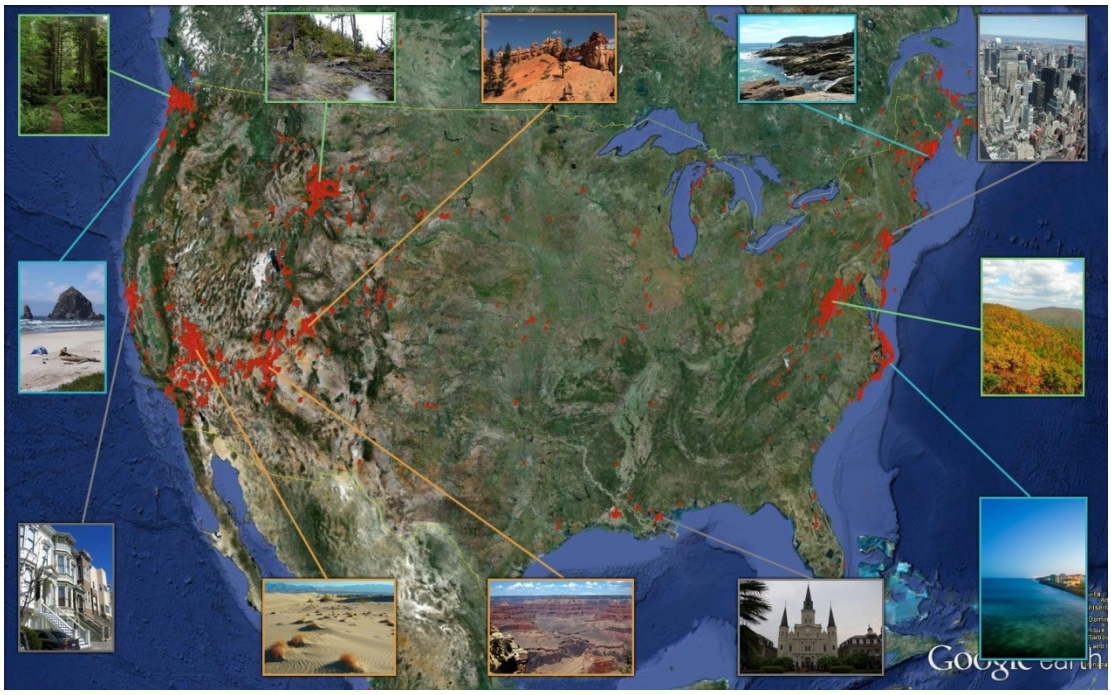

Figure 5.6 Image showing the GPS locations of the images downloaded from Flicker as red dots. Example images of each cluster are also shown and their cluster center is indicated.

Chapter 6: Experimental Results

Sections 6.1, 6.2 and 6.3 contain the experiments that were performed to test the two geolocation algorithms. It is important to note that in all experiments the same set of testing images were used to allow the results to be comparable across experiments. The testing set was created by selecting 100 images from each location and keeping these images isolated from the remaining images so that they would not influence histogram creation and K-means whenever it was performed.

6.1 Support Vector Machine Method

6.1.1 Performance Across Database Sizes

The first test was conducted to determine how the database size affected the overall accuracy and the hierarchical accuracy. To perform this task efficiently, 100 images per class were selected as test images, and 1,200 images per class were selected as training images. 1,200 images per class were selected because 1,301 was the maximum number of images each class could have without creating an imbalance of images per class.

Once the 1,200 images per class were selected, the experiment was set up using different training sizes. This step was accomplished by first extracting image features and creating histograms for the SIFT, RGB, texton, and line image features. Next, a random selection of training images were selected to form training sets of 50, 100, 150, 200, 300, 400, 500, 600, 800, 1,000, and 1,200 images per class. Then, hierarchical and location classifiers were generated according to Section 3.2, using 10-fold cross-validation and 2,500 iterations per SVM. Finally, the test images were used to compute the overall and hierarchical accuracies.

The results for these tests are shown in Figure 6.1. The overall accuracy is shown by a red, solid line and the hierarchical accuracy by a dotted, blue line. The graph demonstrates that as additional training images are incorporated into the classifier creation, accuracy improves. It is also interesting to note that the hierarchical classifiers require fewer images per class to perform well, with greater than 70% accuracy at 50 images per class. Additionally, increasing the number of training images from 50 to 1,200 images per class only increases hierarchical accuracy 13%, while the overall accuracy increases 22%.

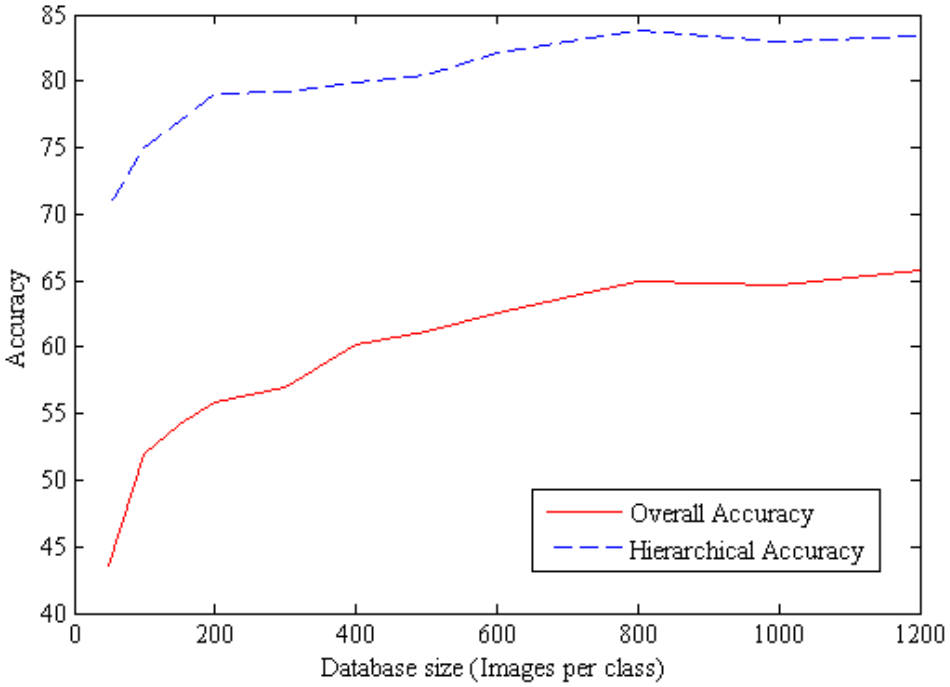

Figure 6.1 Performance across database sizes using the hierarchical SVM method. The overall accuracy (red line) and the hierarchical accuracy (dotted blue line) were computed for different database sizes. The database sizes represent how many images per location were selected for training the SVM classifiers.

Additionally, observe that at around 600 training images per class the amount of additional accuracy added by incorporating additional images begins to reduce.

This observation was utilized in the following experiment due to the long computation times required to create the SVMs.

6.1.2 Performance Across Features

The second test was conducted to determine how the different image features affected the overall and hierarchical accuracies. Classifiers were created using each image feature on its own to reveal how well they could perform geolocation. Furthermore, classifiers were generated by withholding an image feature from the classifiers to expose how the lack of an image feature impacted the accuracy.

For efficiency, 100 images per class were selected as test images, and 600 images per class were selected as training images. 600 images per class were selected as training images because the experiment described in 6.1.1 indicated that adding additional images did not significantly impact the accuracy of the classifiers.

Once the 600 images per class were selected, the experiment tested each of the features individually. First, the image features for the SIFT, RGB, texton, and line image were extracted. Next, each feature was used in isolation to create the hierarchical and location classifiers. Once the set of individual feature classifiers was created, a set of five additional classifiers were formed by withholding an image feature. Instead of using the classifier creation algorithm described in Section 3.2, only energy normalization was performed on the data. Additionally, no aggregation was performed on the gist data, and histogram aggregation was performed on the remaining image features. This decision was made to reduce the overall time required to conduct the experiment and to avoid a feature being left out. Furthermore, preliminary tests established that the aforementioned normalizations and aggregations

often performed best. Once the features were aggregated and normalized they were concatenated together. The SVMs were created using 10-fold cross-validation and 2,500 iterations. Lastly, the test images were used to compute the overall and hierarchical accuracies.

The overall and hierarchical results are shown in Figure 6.2 and Figure 6.3, respectively. A critical finding is that when all image features are forced to be used in the classifier creation, the classifiers achieved an overall accuracy of 63% and a hierarchical accuracy of 81.1667%.

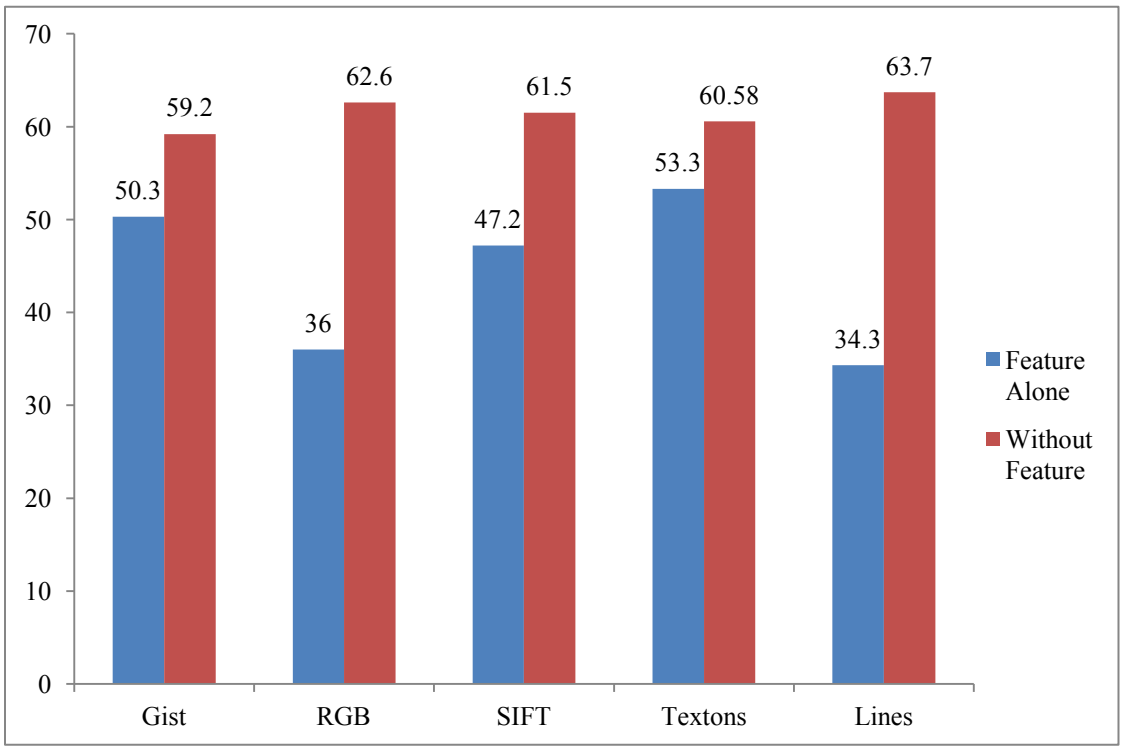

Figure 6.2 Accuracy across features using the hierarchical SVM method. The blue bars display the accuracy induced by only using a single image feature on its own. The red bars display the accuracy induced by not allowing the use of a single image feature.

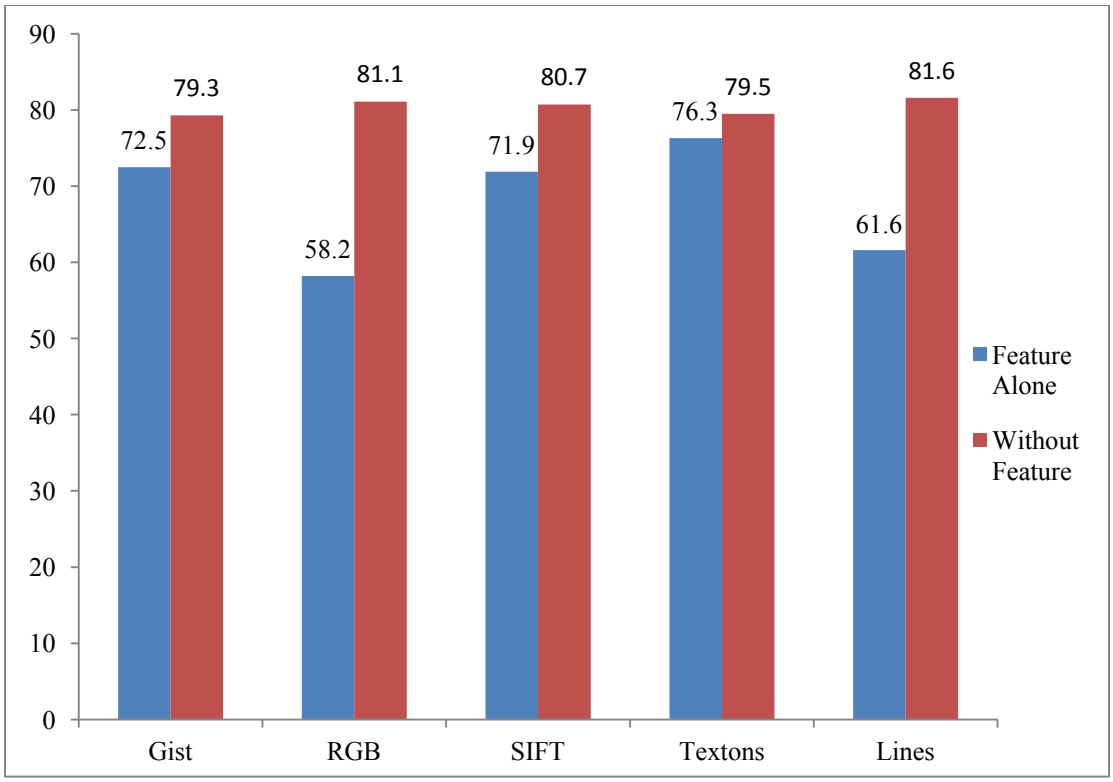

Figure 6.3 Hierarchical accuracy across features for the hierarchical method. The blue bars display the accuracy induced by only using a single image feature on its own. The red bars display the accuracy induced by not allowing the use of a single image feature.

From Figures 6.2 and 6.3 it is possible to rank the features by how well they performed geolocation on their own. The results indicate that when this algorithm is used, textons can perform both geolocation and environmental classification the best when only one feature is selected in isolation. Compared to other isolated features, line features alone performed geolocation the worst and RGB histograms performed environmental classification the worst. The classifiers produced without an image feature reveal two facts. First, leaving out gist negatively impacted geolocation and environmental classification the most. Second, leaving out line features improved geolocation. The accuracy was 63.7% without line features and only 63% when all image features were used.

6.1.3 Performance Without Environmental Classifiers

The third test was conducted to determine if the environmental classifiers actually improved the geolocation accuracy. Individual classifiers for each location were formed according to Section 3.2. Each classifier was created using the maximum amount of training images per class by selecting an equal amount of images from the location represented by the classifier and an equal amount of images from the other locations, resulting in an even amount of {+1,-1} labeled images. Next, the 100 testing images per class, selected before all experiments were conducted, were used to test the classifiers. The results for this experiment are shown in Figure 6.4.

Predicted Location / **True Location**	Grand Canyon	Death Valley	Bryce Canyon	Yellowstone	Olympic	Shenandoah	Acadia	Outer Banks	Cannon Beach	San Francisco	New York	New Orleans
Grand Canyon	**65**	7	9	3	3	6	3	1	2	0	0	1
Death Valley	4	**68**	6	7	0	1	3	5	6	0	0	0
Bryce Canyon	6	7	**72**	4	3	1	4	1	2	0	0	0
Yellowstone	8	0	0	**67**	6	10	2	3	4	0	0	0
Olympic	1	1	1	9	**65**	12	4	1	2	0	1	3
Shenandoah	5	4	1	6	16	**62**	3	2	1	0	0	0
Acadia	6	7	5	3	4	5	**60**	7	3	0	0	0
Outer Banks	3	6	4	6	0	4	4	**52**	7	10	1	3
Cannon Beach	2	2	3	5	0	4	9	8	**66**	0	1	0
San Francisco	0	2	0	1	1	1	2	3	0	**57**	17	16
New York	1	0	0	0	1	0	0	3	1	24	**58**	12
New Orleans	0	0	1	1	1	1	0	4	2	22	5	**63**

Figure 6.4 Confusion matrix produced by withholding hierarchical classifiers. Colored boxes indicate environmental groups.

The algorithm produced an overall accuracy of 62.9%. The results have been grouped according to their environment. An essential fact is that the colored squares represent misclassifications which occurred within the locations of the same environment. Upon analysis of the confusion matrix in Figure 6.4, many misclassifications within the city and forest environments are evident.

Using the non-hierarchical approach it is possible to determine the accuracy of the algorithm described in "Mapping the World's Photos" [2]. This was done by simply restricting the algorithm to only use SIFT features and not use environmental classifiers. Using the same test images as before their algorithm produced an overall accuracy of 47.17%.

6.1.4 Alternative Classifier Performance

To test the alternative classifier creation method described in Section 3.3, classifiers were created for each location and environment. Each classifier was generated using the maximum amount of training images per class by selecting an equal amount of images from the location or environment represented by the classifier and an equal amount of images from the other locations, creating an even amount of {+1,-1} labeled images. The training data for each classifier was then separated into two groups: two thirds was used to train the classifiers for each feature, and the remaining third was reserved to train the final classifier. The later third was fed through the feature classifiers to generate the vector $\boldsymbol{v_i}$ for each image. Next, the 100 testing images, per class selected before all experiments were conducted, were employed to test the classifiers.

After testing all of the testing images, the algorithm produced an overall accuracy of 65.5% and a hierarchical accuracy of 82.25%.

6.1.4 Overall Performance

The final test, using the hierarchical geolocation algorithm with SVMs, was performed to determine the accuracy when the information gathered in section 6.1.1 and 6.1.2 was incorporated. Hierarchical and individual classifiers were created according to section 3.2 to find an optimal combination of features. Since the results from 6.1.1 suggested that increasing the amount of training images increased accuracy, the maximum amount of images for each location were used. Each classifier was generated using the maximum amount of training images per class by selecting an equal amount of images from the location or environment represented by the classifier and an equal amount of images from the other locations, creating an even amount of {+1,-1} labeled images. Next, the 100 testing images per class, selected before all experiments were conducted, were employed to test the classifiers.

After testing all of the testing images, the algorithm produced an overall accuracy of 68.83% and a hierarchical accuracy of 85.33%. The results are given in the confusion matrix of Figure 6.5.

Predicted Location / True Location	Grand Canyon	Death Valley	Bryce Canyon	Yellowstone	Olympic	Shenandoah	Acadia	Outer Banks	Cannon Beach	San Francisco	New York	New Orleans
Grand Canyon	**71**	6	7	2	3	6	2	2	1	0	0	0
Death Valley	10	**66**	6	3	0	1	1	9	4	0	0	0
Bryce Canyon	7	4	**71**	4	3	1	5	0	4	1	0	0
Yellowstone	5	1	4	**71**	11	3	4	0	1	0	0	0
Olympic	0	1	1	7	**75**	10	2	1	1	0	0	2
Shenandoah	7	1	2	7	13	**60**	4	5	0	0	0	1
Acadia	1	7	5	3	3	7	**63**	9	2	0	0	0
Outer Banks	1	6	3	5	1	2	2	**66**	4	6	2	2
Cannon Beach	0	0	2	2	0	2	6	9	**78**	0	1	0
San Francisco	0	1	2	0	1	1	0	3	0	**57**	15	20
New York	0	0	0	0	0	1	1	2	1	17	**67**	11
New Orleans	0	0	1	1	1	1	0	2	1	8	4	**81**

Figure 6.5 Confusion matrix produced using hierarchical classifiers. Colored boxes indicate environmental groups.

Comparing Figures 6.4 to 6.5, an additional advantage aside from the increase in overall accuracy can be seen. This advantage is that when the correct environment is identified, the within-environment accuracy is higher. The hierarchical geolocation method produced an accuracy of 80.67%, assuming the estimated environment is correct, while the non-hierarchical geolocation method produced an accuracy of 76.5%. This advantage is particularly apparent in the classification of images from New Orleans. Using the non-hierarchical classification method, 22 images from New Orleans were incorrectly labeled as images from San Francisco, while only 8 images were misclassified as being from San Francisco using the hierarchical classification.

The hierarchical accuracy produced from the hierarchical classifiers is shown in Table 6.1. A significant aspect is the high accuracy the city environmental

classifier produced. This classifier essentially indicates if an image contains man-made objects like houses, buildings, and roads.

Table 6.1 Hierarchical accuracies from hierarchical geolocation system.

Environment	Accuracy
Desert	82.67
Forest	85.67
Coast	79.67
City	93.33

6.2 Linear Dictionary-Based Method

The first test using the dictionary-based method was executed to determine how well linear dictionaries could perform image geolocation. Class-specific dictionaries were formed according to Chapter 4.1 by reserving the specified 100 images for testing and using all of the remaining images for testing. An ideal dictionary size and sparsity constant were determined using 3-fold cross-validation and 80 iterations per dictionary. Instead of dividing the data into three even subsets, a unique set of 200 training images per location were selected each time to determine the cross-validation accuracy. This decision was made due to the long computation times required to train the dictionaries. The dictionary sizes tested were 100, 200,300,400, and 500, and the sparsity constants tested were 5, 15, 35, and 45. After testing all parameter combinations, the combination producing the highest cross-validation accuracy was a dictionary size of 200 atoms and a sparsity constant equal to five.

Using the testing images, the accuracy of the system was determined to be 59.75%. The confusion matrix associated with this experiment is shown in Figure 6.6.

True Location \ Predicted Location	Grand Canyon	Death Valley	Bryce Canyon	Yellowstone	Olympic	Shenandoah	Acadia	Outer Banks	Cannon Beach	San Francisco	New York	New Orleans
Grand Canyon	**45**	3	21	9	5	4	9	1	1	0	1	1
Death Valley	5	**45**	20	8	2	3	6	5	4	1	1	0
Bryce Canyon	2	0	**75**	4	5	1	8	4	1	0	0	0
Yellowstone	3	0	7	**62**	12	5	1	6	2	1	0	1
Olympic	0	1	2	8	**81**	2	4	0	1	0	0	1
Shenandoah	7	0	2	10	35	**32**	6	3	1	1	1	2
Acadia	5	2	8	2	7	6	**59**	8	2	0	0	1
Outer Banks	2	3	2	11	1	6	1	**55**	8	2	4	5
Cannon Beach	2	0	6	7	1	1	9	9	**65**	0	0	0
San Francisco	1	0	6	2	3	0	2	1	0	**49**	18	18
New York	0	0	0	1	0	0	0	3	0	12	**69**	15
New Orleans	0	0	0	2	3	0	2	1	0	7	5	**80**

Figure 6.6 Confusion matrix produced using linear class specific dictionaries. Colored boxes indicate environmental groups.

From the confusion matrixes and an overall accuracy of 59.75%, it is clear that the linear dictionaries can perform image geolocation. However, upon examining some of the individual locations, some weaknesses are revealed. For example, the testing images from Grand Canyon and Death Valley National Park were correctly classified 45% of the time. These accuracies are very different from the accuracies produced by the hierarchical SVM method which achieved 71% and 66%, respectively. From the confusion matrix, it appears that this is mostly due to images from these locations being misclassified as being from Bryce Canyon National Park with 21 and 20 images being misclassified, respectively.

6.3 Non-linear Dictionary-Based Method

6.3.1 Performance Across Database Sizes

The first test using non-linear dictionaries was conducted to determine how the database size affected the overall accuracy of the system. To perform this task efficiently, 100 images per class were selected as test images, and 1,200 images per class were selected as training images. 1,200 images per class were selected because 1,301 was the maximum number of images each class could have without creating an imbalance of images per class.

Once the 1,200 images per class were selected, the experiment was set up using different training sizes. This step was accomplished by first extracting image features and creating histograms for the SIFT, RGB, texton, and line image features. Next, a random selection of training images were selected to form training sets of 50, 100, 150, 200, 300, 400, 500, 600, 800, 1,000, and 1,200 images per class. Then, location specific dictionaries were generated according to Section 4.2, using 3-fold cross-validation and 80 iterations, using the same cross validation training procedure described in 6.2. Finally, the test images were used to compute the overall and hierarchical accuracies.

The results for these tests are shown in Figure 6.7. The overall accuracy of the non-linear dictionary-based recognition algorithm is shown by a red, solid line. For comparison, the hierarchical SVM-based recognition algorithm's results are shown by a blue, dotted line. From the graph it is clear that as additional training images are incorporated into the classifier creation, accuracy improves. Comparison of the two

graphs reveals that the non-linear dictionary-based recognition algorithm performs better for all tested dictionary sizes.

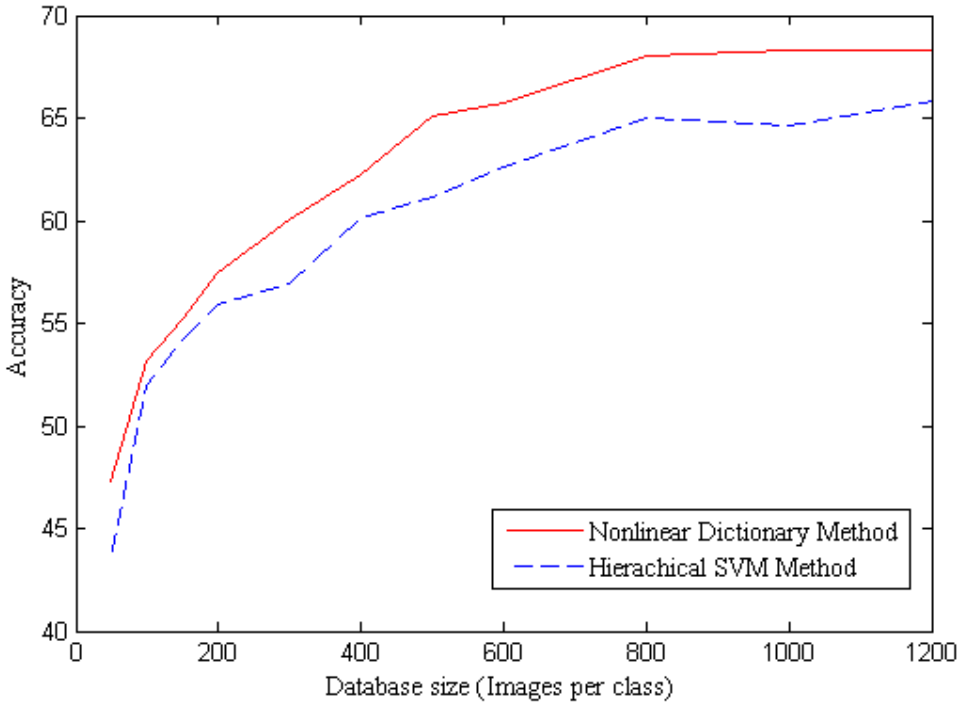

Figure 6.7 Performance across database sizes using the non-linear dictionary method. The accuracy of the non-linear dictionaries (red line) and the hierarchical SVM method's accuracy (dotted blue line) for comparison. The database sizes represent how many images per location were selected for training the class dictionaries and SVM classifiers.

Additionally, it is important to note that at around 300 to 400 training images per class the amount of additional accuracy added by incorporating additional images begins to reduce. This observation was utilized in the following experiment due to the long computation times required to create the dictionaries.

6.3.2 Performance Across Features

The second test using non-linear dictionaries was conducted to determine how the different image features affected the accuracy. Similar to Section 6.2, dictionaries were created using each image feature on its own to reveal how well they could

perform geolocation. Additionally, classifiers were generated by withholding an image feature to expose how the lack of an image feature impacted the accuracy.

For efficiency, 100 images per class were selected as test images, 300 images per class were selected as training images, and 900 images per class were withheld for parameter searches. 300 images per class were selected as training images because the experiment described in 6.3.1 indicated that 300 images per class produced decent results and while adding additional images would increase accuracy, the additional amount of time required to train the dictionaries would cause the experiment to take too long. 900 images per class were withheld to allow for optimal parameter searches instead of performing cross validation. This was done to reduce the number of dictionaries created to the number of parameter combinations searched.

The image features were normalized using energy normalization. Additionally, no aggregation was performed on the gist data, and histogram aggregation was performed on the remaining image features. This decision was made to reduce the overall time required to conduct the experiment and to avoid a feature being left out. Once the features were aggregated and normalized, they were concatenated together, and location specific dictionaries were tested with different parameter combinations using the 900 training images that were withheld.

The results for these tests are shown in Figure 6.8. A critical finding is that when all image features are forced to be used in the classifier creation, the classifiers achieved an accuracy of 60.083%.

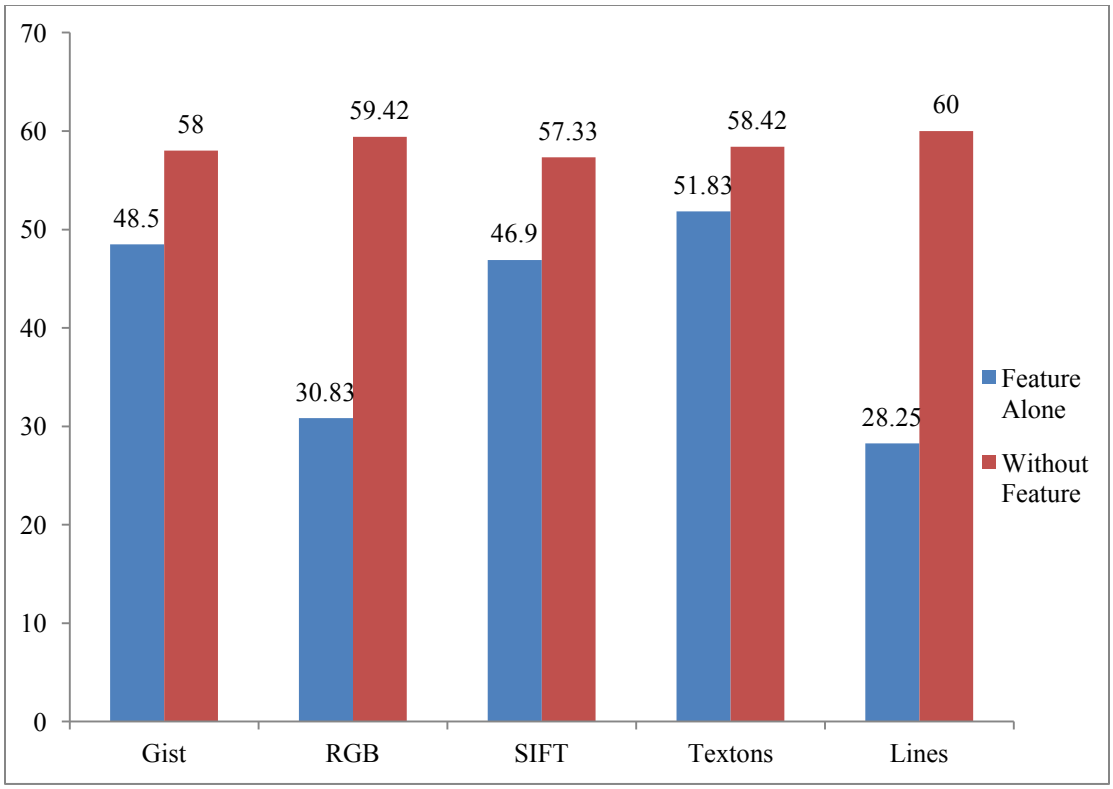

Figure 6.8 Overall accuracy across features using non-linear dictionary-based recognition. The blue bars display the accuracy induced by only using a single image feature on its own. The red bars display the accuracy induced by not allowing the use of a single image feature.

From Figures 6.8 it is possible to rank the features by how well they performed geolocation on their own. The results indicate that when this algorithm is used, textons can perform geolocation the best. Line features alone performed geolocation the worst. The classifiers produced without an image feature reveal two facts. First, leaving out gist negatively impacted geolocation and environmental classification the most. Second, leaving out line features did not detrimentally affect geolocation like it affected the hierarchical geolocation system. Instead, line features added a small .083% increase, equivalent to one additional correctly recognized image.

A direct comparison between Figures 6.8 and 6.2 is not possible due to the selection of training samples. The experiment in Section 6.1.2 used 600 training

samples to train the data, and the experiment in Section 6.3.2 used only 300 with no guarantee of overlap.

6.3.3 Overall Performance

The final test, using the non-linear dictionaries, was to determine the accuracy when the information gathered in section 6.3.1 and 6.3.2 was incorporated. Class specific dictionaries were created according to Section 4.2. Since the results from 6.3.2 suggested that increasing the amount of training images increased accuracy, the maximum amount of images for each location were used. Each dictionary was generated using the maximum amount of training images per class. Due to the increased number of training samples, the parameters found in Section 6.3.1 for the 1,200 training samples per class case were used. This was done to reduce the amount of time required to train all of the dictionaries (around two and a half days on a high-end PC). Next, the 100 testing images per class, selected before all experiments were conducted, were employed to test the dictionaries.

After testing all of the testing images, the algorithm produced an overall accuracy of 70.50% and a hierarchical accuracy of 85.67%. The results are given in the confusion matrix of Figure 6.9.

True Location \ Predicted Location	Grand Canyon	Death Valley	Bryce Canyon	Yellowstone	Olympic	Shenandoah	Acadia	Outer Banks	Cannon Beach	San Francisco	New York	New Orleans
Grand Canyon	**82**	0	6	1	1	6	2	2	0	0	0	0
Death Valley	14	**58**	8	1	2	1	2	11	2	1	0	0
Bryce Canyon	9	2	**71**	1	3	3	5	5	1	0	0	0
Yellowstone	4	0	6	**64**	7	10	3	5	1	0	0	0
Olympic	0	0	1	5	**77**	12	0	2	3	0	0	0
Shenandoah	8	0	2	3	10	**73**	1	2	0	1	0	0
Acadia	10	3	2	2	2	7	**64**	7	3	0	0	0
Outer Banks	2	4	1	6	1	3	0	**73**	4	2	4	0
Cannon Beach	2	0	3	0	0	5	6	10	**72**	1	1	0
San Francisco	1	0	0	1	2	5	0	2	0	**56**	24	9
New York	0	0	0	0	1	0	0	1	0	9	**82**	7
New Orleans	0	0	1	0	1	4	0	3	0	7	10	**74**

Figure 6.9 Confusion matrix produced using non-linear class specific dictionaries. Colored boxes indicate environmental groups.

Comparing Figures 6.5 to 6.9, similarities are apparent. The first similarity is that both methods can accurately classify the city environment from the natural environments, with the hierarchical SVM method achieving 92.67% and the non-linear dictionary-based method achieving 91.67%. Additionally, both methods encounter difficulty classifying images from San Francisco. Another similarity is the difficulty both methods encounter while classifying images from Death Valley National Park. The difficulty classifying Death Valley National Park is more pronounced with the dictionary-based method, achieving an accuracy of 58%, compared to 66%.

Chapter 7: Conclusions

7.1 Comparison Between Methods

The results for all of the different algorithms implemented in this thesis are shown in Table 7.1. For non-hierarchical structured algorithms, the environmental accuracy was determined by adding together all of the results of images whose estimated location belonged to the correct environment. From Table 7.1, it is evident that the non-linear dictionaries perform geolocation slightly better than the hierarchical SVMs. The improved accuracy is also evident in Figure 6.7 where the non-linear dictionaries performed more accurately for each amount of training images. Additionally, the table indicates that hierarchical SVMs and non-linear dictionaries perform environmental accuracy with a similar degree of accuracy.

Table 7.1 Comparison of results between all recognition systems. For non-hierarchical systems, hierarchical accuracy was computed by combining the results of locations within their environment.

Recognition System	Accuracy	Environmental Accuracy
"Mapping the World's Photos"	47.17	72.42
Non-hierarchical SVMs	62.90	82.25
Hierarchical SVMs	68.83	85.35
Hierarchical SVMs (Alternate Classifier Creation)	65.50	82.25
Linear Dictionaries	59.75	79.33
Non-linear Dictionaries	**70.50**	**85.67**

It should be noted that a hierarchical structure was attempted using both linear and non-linear dictionaries. A full scale test using all twelve locations was not implemented because smaller tests using fewer locations indicated that the hierarchical structure did not provide an improvement in accuracy. This may be due

to the high environmental accuracy the non-linear dictionaries achieve without the hierarchical structure. However, there may be other methods to implement a hierarchy to allow the dictionaries to benefit from histograms consisting only of elements from the same environment.

The results from Table 7.1 also indicate that the non-linear dictionary algorithm provides a significant increase in accuracy when compared to using linear dictionaries. However, the linear dictionaries have one advantage over the non-linear dictionaries in that the dictionaries can be trained faster.

Despite the similar classification performance, there are large differences in the amount of time required to create the dictionaries and SVMs and the space required to store them for future use. The amount of time required to train 12 non-linear class-specific dictionaries using the maximum amount of training images without a parameter search or cross-validation takes the same amount of time required to train 12 SVMs with a parameter search of 30 parameter combinations and 12-fold cross-validation. In other words, training the dictionaries takes about 360 times longer. This difference may be due to the fact that the code that creates non-linear dictionaries is written in MATLAB and the code to create the SVMs is written in C++. When comparing the space required to store the different algorithms, storing 12 class-specific dictionaries requires about 320 MB, while storing 16 SVMs requires 2GB. Using these differences it is possible to conceive some scenarios. For example, if one wanted to quickly test a new image feature and see how well it could aid in geolocation, SVMs would be recommended. However, if one wanted to have an

accurate method to classify images with a small file size and did not care about computation times, non-linear dictionaries would be better suited.

7.2 Improvements to Geolocation

The results of Chapter 6 enforce a few principles that can aid geolocation algorithms. The first principle demonstrated by this thesis is that a hierarchical structure can bring improvements in accuracy. This is specifically seen by the 6% difference in accuracy between the nonhierarchical and hierarchical classification schemes. The second principle demonstrated by this thesis is that none of the tested individual image features can perform geolocation as well as a combination of features. Additionally, the results indicate that simply forcing the classifiers to use all of the features can be detrimental and that the algorithm should be allowed to choose the features that work best. The final principle demonstrated by this thesis is that both SVMs and class-specific dictionaries can perform comparably. Overall, non-linear class-specific dictionaries perform geolocation better than SVMs; however, through planning like hierarchical classification, an algorithm utilizing SVMs can be competitive.

Chapter 8: Future Work

To improve the geolocation algorithms described in this thesis, many different extensions could be incorporated. These include: adding additional locations and environments, additional image features, performing multi-scale recognition, and performing automatic image screening.

8.1 Additional Locations

Due to time and data limitations this experiment does not perform true geolocation, giving GPS coordinates from any outdoor image. Since only 12 locations were tested, the testing images were limited to these locations. There are, therefore, numerous national parks, beaches, and cities that were not included in the experiment. Additionally, the locations chosen for this experiment were all in the United States, excluding the rest of the world.

Adding additional locations will be crucial to determining how well the system can scale with the number of locations. While images were screened and added to the experiment, it was noticed that additional locations slowly decreased the accuracy of the system. However, when the accuracy was compared to chance, the systems in this thesis were much better. Despite this improvement, it would seem unsatisfactory to add many additional locations and have the algorithm only perform well when compared to chance.

8.2 Additional Image Features

From the results of Chapter 6, it is clear there is not an individual feature that performs geolocation well when compared to using all of the features. From this we

can surmise that adding or replacing the current selection of features should improve the accuracy. Other image features that could be included are histograms of oriented gradients (HOG) [15], local binary patterns (LBP) [16], and self-similarity descriptors (SSIM) [17]. Additionally, when using the hierarchical model, features specific to environment could be introduced, such as a histogram of the flora and fauna observed in the image found by plant identification in natural environments [18]. A city specific image feature could also be used to improve recognition between cities, like identifying the content on signs or examining license plates on cars.

8.2 Multi-Scale Recognition

Another method to improve the algorithm would be to allow for multi-scale resolution. This would allow for landmark level classification using the same hierarchical mode in Chapter 3. The algorithm would first determine the city/national park like it currently does. Once the general location was determined, the landmark would be estimated by creating classifiers for all landmarks within the general location. This change could be implemented using mean-shift on the GPS coordinates around each location to identify areas of high photograph activity.

8.3 Automatic Image Screening

The final change that could be implemented is automatic image screening. As specified in Chapter 5, of the 89 thousand images downloaded, only 23 thousand of the images were acceptable images. With an average screening time of two seconds per image, the screening of all of the images took approximately 50 hours to complete. Since no images were deleted, this has left a large labeled dataset with the

images labeled either accepted or rejected. Using this dataset, classifiers could be trained to automatically approve images. This could dramatically reduce the amount of time involved in adding new locations and allow for the entire system be automated.

Bibliography

[1] J. Hayes and A. A. Effros, "IM2GPS: estimating geographic information from a single image," in *IEEE Confrence on Computer Vision and Pattern Recogntion (CVPR)*, 2008.

[2] D. Crandall, L. Backstrom, D. Huttenlocher and J. Klienberg, "Mapping the world's photos," in *Eighteenth International World Wide Web Confrence (WWW)*, 2009.

[3] B. Julesz, "Textons, the elements of texture perception, and ther interactions," *Nature,* vol. 290, pp. 91-97, 1981.

[4] T. Leung and J. Malik, "Representing and recognizing the visual apperance of materials using three-dimensional textons," *International Journal of Computer Vision,* vol. 43, no. 1, pp. 29-44, 2001.

[5] D. Martin, C. Fowlkes, D. Tal and J. Malik, "A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics," in *Eighth International Confrence on Computer Vision*, 2001.

[6] J. Kosecka and W. Zhang, "Video compass," in *Seventh European Confrence on Computer Vision*, 2002.

[7] B. Li, K. Peng, X. Ying and H. Zha, "Vanishing point detection using cascaded 1D hough transform from single images," *Pattern Recogntion Letters,* vol. 33, no. 1, pp. 1-102, 2012.

[8] A. Olivia and A. Torralba, "Modeling the shape of the scene: a holistic representation of the spatial envelope," *International Journal of Computer Vision,* vol. 42, no. 3, pp. 145-175, 2001.

[9] D. G. Lowe, "Object recogntion from local scale-invariant features," in *International Confrence on Computer Vision*, 1999.

[10] H. V. Nguyen, V. M. Patel, N. M. Nasrabadi and R. Chellappa, "Kernel dictionary learning," in *IEEE International Confrence on Acoustics, Speech and Signal Processing*, 2012.

[11] D. Arthur and S. Vassilvitskii, "K-means++: the advantages of careful seeding," in *Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, 2007.

[12] A. Pronobis, O. Martinez Mozos and B. Caputo, "SVM-based discriminative accumulation scheme for place recognition," in *IEEE International Confrence on Robotics and Automation*, Pasadena, 2008.

[13] V. M. Patel, T. Wu, S. Biswas, J. Phillips and R. Chellappa, "Dictionary-based face recogntion under variable lighting and pose," *IEEE Transactions on Information Forensics and Security,* vol. 7, pp. 954-965, 2012.

[14] M. Aharon, M. Elad and A. Bruckstein, "K-SVD: an algorithm for designing overcomplete dictionaries for sparse representation," *IEEE Transactions on Signal Processing,* vol. 54, no. 11, pp. 4311-4322, 2006.

[15] K. Fukunaga and L. D. Hosteller, "The esitmation of the gradient of a density function, with applications in pattern recogntion," *IEEE Transactions on Information Theory,* vol. 21, no. 1, pp. 32-40, 1975.

[16] N. Dalal and B. Triggs, "Histograms of Oriented Gradients for Human Detection," in *IEEE Confrence on Computer Vision and Pattern Recogntion*, San Diego, 2005.

[17] T. Ojala, M. Pietikainen and D. Harwood, "Performance evaluation of texture measures with classification based on kullback discrimination of distributions," in *International Confrence on Pattern Recogntion*, 1994.

[18] E. Shechtman and M. Irani, "Matching local self-similarities across images and videos," in *IEEE Confrence on Computer Vision and Pattern Recognition*, 2007.

[19] P. N. Belhumeur, D. Chen, F. Steven, D. Jacobs, W. J. Kress, H. Ling, I. Lopez, R. Ramamoorthi, S. Sheorey, S. White and L. Zhang, "Searching the world's herbaria: a system for visual identification of plant species," in *European Confrence on Computer Vision*, 2008.

[20] N. Kumar, A. Berg, P. Belhumeur and S. K. Nayar, "Describable visual attributes for face verfication and image search," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 33, no. 10, pp. 1962-1977, 2011.