EC520 FINAL PROJECT REPORT: EXAMPLE-BASED IMAGE RETRIEVAL

Joshua Rapp, Sarah Welch, Srilalitha Kumaresan

Boston University
Department of Electrical and Computer Engineering
8 Saint Mary's Street
Boston, MA 02215
`www.bu.edu/ece`

June 18, 2015

Technical Report No. ECE-2015-02

Contents

List of Figures

List of Tables

1 Introduction

The goal of this project was to develop an example-based image retrieval system, which sorts through a large database of images by using image characteristics rather than previously added metadata. Given a query image, the system returns images from the database that are similar to the query. Potential applications of example-based image retrieval today include quick personal photo organization, surveillance, and medical diagnosis.

The first step in example-based image retrieval is to represent the database and the query images in a useful format for comparison. When searching for images similar to a query, not all image information is necessary or relevant. Important aspects of a query image can be enhanced or extracted to perform a search using only essential information. Once an image representation method is chosen, the next task is to determine which images in the database are similar to the example image, with similarity measured by a distance metric. A classifier uses this metric to select database images based on their close distance to the query image. Assuming that the images in the database are preclassified based on metadata, the results of example-based image retrieval can be validated against ground truth.

2 Literature Review

Historically, a variety of image characteristics have been used as the basis for example-based image retrieval. Early computer vision techniques used the raw pixel values as region descriptors, however, changes in illumination and non-rigid motion were problematic with this technique. Extensive normalization can resolve these problems at the cost of computational complexity [1] [2]. Another representation originally used by Swain and Ballard in [3] is the histogram method, which estimates the probability densities of raw pixel values. This technique resolves the problems associated with using raw pixel values but is computationally complex even with fast methods, since exhaustive search must be used to find the best match of the example image [4]. Furthermore, joint distributions need to be computed in order to represent the relationships between multiple features, which leads to exponential complexity [2].

Another class of algorithms use scale-space representations of an image to extract strong local features. In a scale-space representation, one image is represented by a parametrized set of smoothed images. The size of the smoothing kernel parametrizes each image in the scale-space representation. The smoothing kernels will suppress fine-scale features and enhance the image structures that are present at different scales [5]. The Scale Invariant Feature Transform (SIFT) and its derivatives such as Speeded Up Robust Features (SURF) are robust to illumination changes, clutter, partial occlusion, rotation, affine projections, and scale [6] [7]. However, since these algorithms are so well suited to detecting strong features, they are best at identifying particular objects and recognizing the same scene from different view points or under partial occlusion [7]. SIFT is less suited to recognizing similar images of different

scenes, such as images of different beaches or different buildings.

3 Problem Statement

The example-based image retrieval system presented in this report was designed to match images with similar scenes, rather than to detect specific objects within images. Our system has three core components: covariance matrix representation, feature selection, and classification.

3.1 Covariance Matrix Representation

The covariance of image features is a descriptor that can represent the overall scene in an image more generally and with much lower dimensionality than histograms. Specifically, a feature vector is computed at each pixel, where each component of the vector represents a particular feature such as pixel position or color. Then, the sample covariance of all of the feature vectors is computed, which forms a compact second-order characterization of the features. The dimensionality of the covariance matrix is determined by the dimensionality of the feature vector rather than by the size of the image region. For example, if there are n features, then the covariance matrix is $n \times n$, regardless of the size of the considered region. The covariance is robust to illumination changes since it is mean centered; it is also scale and rotation invariant, and does not require normalization [2] [8] [9]. As our goal was to describe general image scenery, we chose to compute the covariance of features over the entire image rather than over image subregions.

3.2 Feature Selection

We investigated three distinct feature sets which were each described in previous research and adapted them to our covariance of features framework.

3.2.1 Standard Features

The first feature set was termed the "Standard" set because of its use in the related object-detection work in [2]. Following that work, a feature vector calculated at each pixel (x, y) is given by

$$F(x,y) = \left[x, y, R(x,y), G(x,y), B(x,y), \left|\frac{\partial I(x,y)}{\partial x}\right|, \left|\frac{\partial I(x,y)}{\partial y}\right|, \left|\frac{\partial^2 I(x,y)}{\partial x^2}\right|, \left|\frac{\partial^2 I(x,y)}{\partial y^2}\right|\right]. \quad (1)$$

The elements of $F(x, y)$ represent the pixel location, followed by the red, green, and blue intensities and the first and second partial derivatives in each direction at that pixel location.

3.2.2 Histogram of Oriented Gradients

Our second feature set is based on the Histogram of Oriented Gradients (HOG) feature vector proposed in [10]. The main idea of the HOG feature set is to describe local shapes encoded in the luminance pattern. The HOG feature vector is computed by dividing the image into cells and calculating the local gradient at each pixel within the cell. The gradients are then quantized to several pre-determined directions. The magnitudes of the quantized gradients within each cell are accumulated into a "weighted" histogram, rather than simply accumulating counts of each orientation. The original work in [10] combined the HOGs from every image cell into one long vector and used it directly in their classification algorithm. Our adaptation fits the HOG feature vectors into the covariance matrix representation as described in section 4.2.2 below.

3.2.3 Gabor wavelet based filter bank

The third feature set takes a Gabor wavelet approach motivated by the properties of the human visual system. According to [11], Gabor functions can model the response to spatial frequency patterns of simple cells in the visual cortex, the portion of the brain which processes visual information. Each simple cell responds to a particular frequency band, with a Gabor-like response. This model motivated the work of Manjunath *et al.* in [12], where a Gabor wavelet filter bank was used to extract texture features from an image. This filter bank represents the responses of many simple cells in the visual cortex over a range of frequencies.

The Gabor function is given by a two-dimensional Gaussian function multiplied by a complex sinusoid:

$$g(x,y) = \left(\frac{1}{2\pi\sigma_x\sigma_y}\right)\exp\left[-\frac{1}{2}\left(\frac{x^2}{\sigma_x^2}+\frac{y^2}{\sigma_y^2}\right)+2\pi jWx\right]. \tag{2}$$

The function $g(x,y)$ is used as a mother wavelet to create scaled and rotated Gabor wavelets, which form a complete, non-orthogonal basis. The complex sinusoid in $g(x,y)$ causes a shift in the frequency response based on W, which leads to the positioning of the filters. This parameter along with the choices of σ_x and σ_y from [12] leaves little spectral gap between filters. The parameter a is chosen to adjust the scales between filters such that they span the entire frequency range.

To calculate the feature values, Manjunath *et al.* convolve each image with each of the Gabor impulse responses. From each filter output, $W_{mn}(x,y)$, two features μ_{mn} and σ_{mn} are calculated by finding the energy of each filter's output:

$$\mu_{mn} = \frac{\sum_y \sum_x |W_{mn}(x,y)|}{N} \tag{3}$$

$$\sigma_{mn} = \frac{1}{N}\sqrt{\sum_y \sum_x \left(|W_{mn}(x,y)| - \mu_{mn}\right)^2}, \tag{4}$$

Table 1: Distance Metric Definitions

Name	Form
Euclidean	$\|K_1 - K_2\|$
Log-Euclidean	$\|\log(K_1) - \log(K_2)\|$
Riemannian	$\|\log(K_1^{-1/2} K_2 K_1^{-1/2})\|$
Cholesky	$\|\text{chol}(K_1) - \text{chol}(K_2)\|$
Root Euclidean	$\|K_1^{1/2} - K_2^{1/2}\|$
Power Euclidean	$\frac{1}{\alpha}\|K_1^{\alpha} - K_2^{\alpha}\|$

where N is the total number of x and y values being summed over. For 24 Gabor filters, the 48 dimensional feature vector for each image is then given by

$$F = [\mu_{00}, \sigma_{00}, \ldots, \mu_{24}, \sigma_{24}]. \tag{5}$$

Section 4.2.3 describes how this method was extended to fit into the covariance of features image descriptor by dividing an image into subregions and computing feature vectors by independently filtering those regions.

3.3 Classifiers

There are a number of standard machine learning methods that can be used for query classification, such as nearest neighbor (NN) used in [9] for action recognition or k-nearest neighbors (kNN) used in [2] for texture classification. Given an example image, the algorithm will find the single image (in the case of NN) or the k images (for kNN) whose covariance matrices are closest to the covariance matrix of the query. Both the NN and kNN classifiers operate on a distance metric, which describes the similarity of two images on the covariance matrix manifold.

3.4 Distance Metrics

Since a covariance matrix is positive semi-definite, the set of all covariance matrices does not form a vector space, so the Euclidean distance is not a useful measure of similarity between covariance matrices [9]. However, a number of techniques exist to map the space of covariance matrices to a vector space, upon which the Euclidean metric can be used. Tuzel *et al.* introduce an affine invariant Riemannian metric in [2], in addition to which a log-Euclidean metric is used in [9]. Other available metrics include those described in [13] or the Power-Euclidean metric introduced in [14]. Table 1 shows the six distance metrics presented in [13] that were implemented for this project.

4 Implementation

4.1 Datasets

We created a database containing 800 images in order to test our algorithms. The database has eight pre-defined image classes with 100 images per class. We obtained the images from Creative Commons licensed images on the Internet, from a research database originally created and used in [15], and from our personal collections. The images are of varying aspect ratios, but we resized each to be 300 pixels in its largest dimension. The eight image classes are Buses, Flowers, Cities, Quadrupeds, Mountains, Beaches, Crowds, and Underwater. Each image was manually verified to ensure that it was assigned to the correct class. Since our goal was to identify database images similar to a query, we were willing to accept a small amount of crossover in the defined image classes. For example, images with horses in the foreground were assigned to the quadruped class even if they had water or mountains in the distance. There are also crowd images that contain building structures in the background. We believe these classes realistically represent the types of images that would likely be encountered in an example-based image retrieval application.

4.2 Feature Sets

4.2.1 Standard Features

Ideally the frequency responses of first- and second-derivative filters should have magnitudes given by $|H_1(e^{j\omega})| = \omega$ and $|H_2(e^{j\omega})| = \omega^2$, respectively. One problem in designing practical filters for these applications is that high frequency noise can be enhanced. One-dimensional filters with impulse responses given by $h_1 = -\delta[n]+\delta[n+2]$ and $h_2 = -\delta[n] + 2\delta[n+1] - \delta[n+2]$ were used in [2]. Figures 1 and 2 show that these filters have near-ideal frequency responses at low frequencies but attenuated responses at high frequencies, avoiding the high-frequency noise problem.

We adapted the filters from [2] by repeating the impulse responses as rows in a 2D filter, such as $h_1 = \frac{1}{\sqrt{6}} \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$. For this horizontal filter example, the 2D response has the same horizontal performance as the 1D case, but the repetition of rows of the matrix allows for smoothing in the vertical dimension, further reducing high-frequency noise.

The nine dimensional feature vector from Equation 1 is calculated at each pixel of the query image. From these vectors, a 9×9 covariance matrix is calculated. To ensure no single feature dominates the covariance matrix, each feature is normalized to the range $[0, 1]$ according to [16]. This allows us to determine which features contributions are more important for image classification.

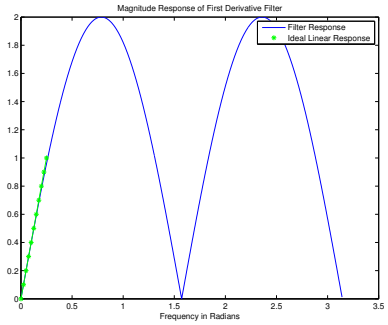


Figure 1: First Derivative Filter Frequency Response

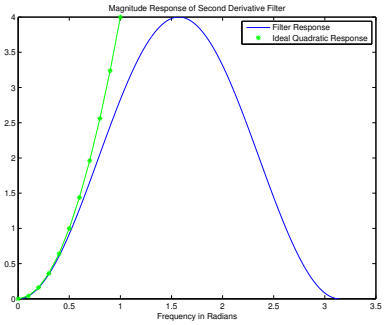


Figure 2: Second Derivative Filter Frequency Response

4.2.2 Histogram of Oriented Gradients Features

Rather than compute a feature vector at each pixel, we divided each image into a 10×10 grid of cells to compute feature vectors for one hundred cells per image. The cells contained roughly 600 pixels a piece, though their exact size and aspect ratio depended on the size of the original image. We transformed the images into YC_BC_R space and considered only the luminance component. Within each cell we computed the horizontal and vertical gradients with the same first-derivative filters as in Section 4.2.1. The gradient orientations were quantized to eight directions uniformly distributed on the unit circle. Finally, rather than use the HOG vectors directly, we computed the sample covariance from all of the cells' feature vectors. While this does obfuscate the location origin of each HOG vector from within the original image, our hypothesis was that the correlation between orientations would still form a unique image fingerprint. For example, in natural images like beaches, there may be little correlation among orientations since any "edges" in water or sand are essentially random, but in cities there may be strong correlations between vertical and horizontal components.

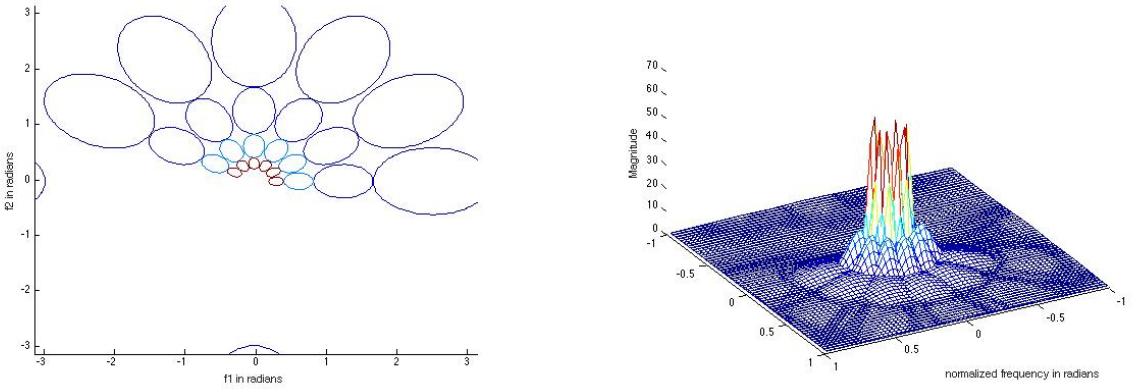
4.2.3 Gabor Features

The MATLAB implementation of the Gabor filters was based on the work of Manjunath *et al.* in [12] and Haghighat *et al.* in [17]. Based off the mother wavelet in Equation 2, a generating function is used to create scaled and rotated Gabor wavelets. For S scales and K rotations, the generating function is given by

$$g_{mn}(x, y) = a^{-m} g(x', y'), \tag{6}$$

where $x' = a^{-m}(x\cos\theta + y\sin\theta)$; $y' = a^{-m}(-x\sin\theta + y\cos\theta)$; $\theta = \frac{n\pi}{K}$; $m = 0, \ldots, S-1$; and $n = 0, \ldots, K-1$.

Based on the example in [12], the parameters were chosen to be $K = 6$ and $S = 4$, which generates 24 filters, and $W = 0.4$, which set the center frequency of the smallest scale impulse response.



(a) Gabor Filter contours at 3dB

(b) Gabor filter magnitude reponses

Figure 3: Frequency Response Plots of 24 Gabor Filters

Filters centered at higher frequencies have a larger bandwidth while filters at lower frequencies have smaller bandwidth, which enables the detection of edges and high frequency changes. Figure 3a shows the contours of the 3dB point of the magnitude of the 24 filters, while the magnitude response of these filters is shown in Figure 3b. The σ_x and σ_y values are chosen such that the filters are touching at the 3dB point and therefore covering the entire frequency range with little spectral gap.

Instead of convolving an entire image with the filters, the image was first divided into 100 cells, as in the HOG implementation. Each filtered cell produces a 48 element feature vector from which the 48×48 covariance matrix is calculated. In the retrieval system described in [12], the query image consisted of a small textured region of an image rather than a full sized image. In place of this, the filtering of the image cell by cell allows filtering of small textured regions rather than using a smaller query image.

4.3 Distance Metrics

All six distance metrics listed in Table 1 were implemented in order to compare their performance. The Euclidean metric was expected to perform poorly compared to the log-Euclidean and Riemannian metrics used in related applications and the other three metrics from unrelated work also involving covariance matrices. Note that the norm used is the Euclidean or Frobenius norm, defined as $\|X\| = \sqrt{\text{trace}(X^T X)}$. For the purpose of comparison in the implementation, the square root was left out to save on computation time.

Given that the distance metrics are simply the Frobenius norm applied to transformations of the feature covariance matrices, the metrics were easy to implement in MATLAB. One important note in terms of computational complexity is the definition of the matrix logarithm, which is $\log(K) = U \log(\Lambda) U^T$, where $K = U \Lambda U^T$ is the spectral decomposition of the matrix K. For large feature covariance matrices, computing the spectral decomposition was a time-consuming process. The computation

duration was especially noticeable when the process is repeated many times for the validation procedure. As a result, it was found that precomputing the covariance matrix logarithm and storing it as K_{log} along with the covariance matrix K during the preprocessing stage allowed the log-Euclidean metric to simply operate the Euclidean metric on the precomputed log-covariance matrices.

4.4 Classifiers

4.4.1 Nearest Neighbor

For the application of an example-based image retrieval system, it is most likely that a user would want to return several images similar to the query and then choose some subset of the returned images. It is less likely that the user would want to return the single most-similar image to the query. Still, it is useful to implement the single-most-similar algorithm for baseline validation of the image retrieval system. When using the feature covariance matrix image descriptor, this translates simply to finding the nearest neighbor to the query on the covariance matrix manifold. To find the nearest neighbor for a given distance metric, the system exhaustively computed the distance between the query and each image in the database, keeping track of the database entry with the smallest distance from the query.

4.4.2 k Nearest Neighbors

Implementation of the k nearest neighbors (kNN) algorithm was similar to that for the NN algorithm, with the main difference being that the algorithm kept track of k database entries rather than a single one. This involves maintaining an array of distances of the k nearest neighbors and then determining where in the array a new entry belongs if it is closer to the query than the kth entry.

5 Experimental Results

5.1 Leave-One-Out Cross Validation

The image-retrieval system's performance was evaluated using the Leave-One-Out Cross-Validation (LOOCV) technique. This process entails removing one database entry at a time and using it as the query to the algorithm. If the class of the query matched the class of its nearest neighbor, the test was considered a success. Otherwise, the test was considered a failure, and the class of the nearest neighbor to the query was noted. The test was then repeated for each of the 800 images in the test database, and the results of the LOOCV tests were tabulated in confusion matrices, such as the example in Figure 4. The matrix diagonal shows successes, while off-diagonal elements show the actual class of the nearest neighbor in the case of a mismatch.

Query Class \ Nearest Neighbor Class	Beach	Bus	City	Crowd	Flowers	Mountains	Quadruped	Underwater
Beach	71%	3%	4%	1%	0%	12%	6%	3%
Bus	0%	92%	4%	3%	0%	0%	1%	0%
City	4%	11%	53%	18%	0%	7%	6%	1%
Crowd	0%	1%	11%	79%	3%	2%	3%	1%
Flowers	0%	0%	0%	1%	98%	0%	1%	0%
Mountains	9%	3%	6%	5%	0%	65%	12%	0%
Quadruped	2%	2%	8%	12%	2%	18%	53%	3%
Underwater	2%	0%	2%	2%	1%	0%	3%	90%

Figure 4: Example Result from LOOCV with Standard Features and log-Euclidean Distance Metric

The process of LOOCV was repeated for each combination of the 3 feature sets and all 6 distance metrics for a total of 18 confusion matrices. The next step was to further digest the data to determine the feature set/distance metric pair for the best performance of each class and the best performance across all classes. Table 2 shows each class, the highest successful classification rate for that class, and the metric and feature set leading to the highest success rate.

Table 2: Best Distance Metric and Feature Set for Each Class

Class	Rate	Metric	Features
Buses	100%	Riemann	Gabor
Flowers	98%	Log/Riemann	Standard/Gabor
Underwater	94%	Riemann	Standard
Crowds	79%	Log	Standard
Beaches	75%	Cholesky	Gabor
Mountains	65%	Log/Riemann	Standard
Quadrupeds	60%	Riemann	Standard
Cities	56%	Root	HOG

One of the most noticeable aspects of the table is the large variation in the best performance rates. While over 90% of the nearest neighbors of the bus, flower, and underwater classes were images of the same class, almost half of the city images had nearest neighbors of a different class. The most likely cause of this difference is the origin and nature of these images. For instance, the bus and flower classes were derived from the research database in [15], and thus had relatively little variation in images within the class. On the other hand, the worst-performing class was the city class, composed of a diverse set of images that may have included too much variation in illumination (both night and day scenes) and scale (both close-ups and distant views).

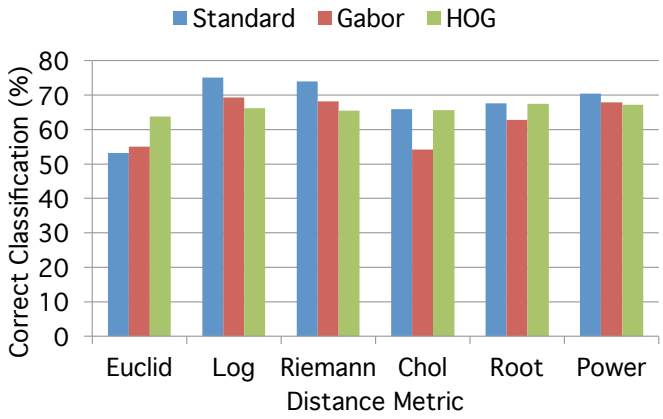
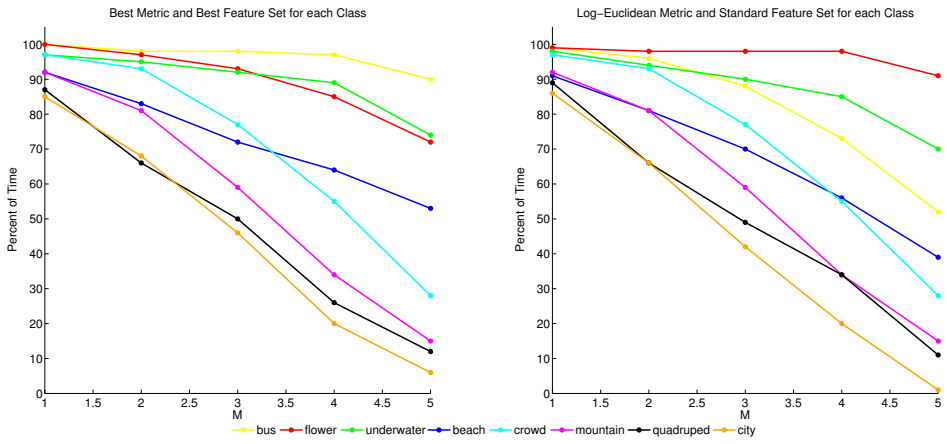


Figure 5: Average Performance Over All Classes

Another important observation from the table is the frequency of the feature sets' appearance. Of the feature sets, the most frequently occurring is the Standard set. Looking at the classes that performed best with the Standard set, one likely explanation of its good performance is the use of color. For instance, both the flower and underwater classes tend to have strong color components, such as the green background and red center common to the flowers or the overall blue-green color that dominates the underwater images. The Gabor feature set performs best when texture dominates rather than color, which makes sense, given the original purpose of the Gabor features. Finally, histogram of oriented gradients perform best only for the cities, which might be because the HOG features encapsulate the strong edges of the manmade scenes but discard the nonuniform color and texture of that class.

Finally, it is important to observe the most frequently occurring distance metric in the table. Clearly, the Riemannian metric dominates, appearing as the best metric for the majority of classes. The log-Euclidean metric also appears as the best for several classes, often tied in performance with the Riemannian metric. This ability to represent the geometry of the covariance matrix manifold is likely why the logarithm-based methods have been used in related applications. It is important to note, however, that the comparison metrics of the Cholesky decomposition and the root-Euclidean metric also appear in the table. While these metrics might not be as useful overall, they still have the potential to be the most effective metric depending on the class definition.

It is interesting to look at differences in performance by class for the image retrieval algorithm, but for a database with different classes or no labeled classes at all, the information in Table 2 is irrelevant. What is more important is to look at the average performance of a feature set/distance metric combination over all classes in the test database. Figure 5 shows the average correct classification rate over all classes using the NN classifier. As expected, the top metrics in Table 2 are also the best over all classes, although in this case, the log-Euclidean metric on Standard features is the



(a) (b)

Figure 6: Comparing application performance using (a) best distance metric/feature set pair for each class and (b) the best average metric/features (log-Euclidean metric, Standard features) over all classes

winner at 75.12% correct classification. Remarkably, the log/Standard pair performs almost as well as the mean performance using the best metric/features for each class, which averages to 78.4%.

5.2 Application Results

For the purpose of the image retrieval application, an important metric of success for the algorithm is to determine how many of the returned images were actually similar to the query, i.e., belong to the same class. For each class, the best metric/feature pair was selected to return the five nearest neighbors of each image in that class. The results were tabulated to find how often all five of the five returned images were of the same class, how often four out of five were correct, etc. Figure 6a shows that for the four best-performing classes, all five nearest neighbors are of the same class in a majority of instances, providing highly useful results in the application setting. Even for the worse-performing classes, a large majority of instances return at least one neighbor of the correct class.

As previously mentioned, the results using the best distance metric and feature set are dependent on the classes used and how well-defined the classes are. In order to get a more general sense of the application performance for the image retrieval system, the same data collection was performed using the best overall distance metric (log-Euclidean) and feature set (Standard). Figure 6b shows the results for all classes using this same metric/feature set pair. For many classes, the probability of having most or all of the nearest five neighbors from the same class is diminished when using log/Standard versus the best metric/features for each class. However, as in Figure 6a, a large majority of instances return at least one neighbor of the correct class in all

cases.

6 Conclusions

The example-based image retrieval system was designed using feature covariance matrices to represent entire image scenes while exploiting the covariance matrices' low dimensionality. System performance was exhaustively validated over 800 images to find the best feature sets and distance metrics to describe the similarity of images. The results indicate that the Riemannian metric and Standard features would be the best choice for the algorithm for several classes. However, as noted in Section 4.3, the spectral decomposition used to calculate the matrix-logarithm was a relatively slow process. Unlike the log-Euclidean metric, which can operate quickly on the precomputed log-covariance matrices, the Riemannian metric involves the logarithm of contributions from both matrices being compared, which must be computed in real time. Since the log-Euclidean metric performance yielded the best average performance over all classes, when speed is of importance, the log-Euclidean metric is likely the most practical option. Indeed, the final results show that, while performance may be optimized by using the best parameters for a class, selecting the log-Euclidean metric and Standard features will likely still result in useful retrieved images.

6.1 Future Work

There are many possible directions for future work in example-based image retrieval. One possible area for major progress would be determining how to combine features into better-performing feature sets. Specifically, one of the reasons that the Standard feature set worked well was the use of the RGB color components, while the Gabor and HOG features operated only on the luminance of the images. If color could be incorporated into the Gabor or HOG features, they might perform better than operating on luminance alone. Another possible direction of research would be to identify significant features in a query image. For example, for the underwater image, an application would identify the significant blue values, especially with respect to the green and red values. Then, the application would know to choose a feature set that includes color, because this identifies underwater images better than the edge directions. On the other hand, manmade scenes may be less identifiable by their colors than by their many horizontal and vertical edges. One possible method of determining the important features could be implementing a version of Principal Component Analysis that determines the most important features, such as in [18] or [19].

The algorithm described in this report is aimed at comparing entire images, which means the classes were primarily chosen to describe scenes. However, when an object within the scene is of interest, such as one of the quadrupeds, the algorithm is not as adept at finding other four-legged animals. One way to extend the algorithm would be

to continue the work in [2], which looks at subregions of the image and calculates the covariance over the features in the subregion. This would allow the algorithm to eliminate the background of a horse, for example, so that the horse is compared to other regions that may contain horses instead of comparing the mountainous background to other mountains.

There are other features, classifiers, and distance metrics that could be explored. For example, the Power-Euclidean distance metric was implemented with the power value α set to a constant of $\frac{1}{3}$. Dryden *et al.* discuss how the metric approaches the log-Euclidean metric as α approaches zero and discuss how to choose an optimal α for diffusion tensor imaging [14]. It could be worthwhile to explore whether $\alpha = 0$ is the optimal value for the Power-Euclidean metric, or if another value surpasses the log-Euclidean performance.

Finally, the problem of estimating population covariance matrices from small sample sizes came to our attention after the initial submission of this report. In [20], Raudys and Jain describe feature selection for pattern recognition systems and state that, while using more features initially helps discriminate among pattern classes, adding features beyond the most representative ones can actually increase classification error. It is also essential for the sample size to be large enough for the sample covariance matrix to be reliable. The necessary sample size depends on the complexity of a pattern recognition decision rule (such as the number of neighbors used for kNN) and the number of features used. For our project, we extended the Gabor filter work done by Manjunath et al. by generating 48 features from 24 Gabor filters over 100 cells within each image. Because the 48 by 48 covariance matrix has over 1000 unique entries, the 100 cells likely did not provide enough samples to be reliable for classification. An important future extension of the project would use only the most representative filters to produce a smaller covariance matrix for which the 100 image blocks provide a sufficient number of samples.

References

[1] R. Brunelli and T. Poggio, "Face recognition: features versus templates," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 15, pp. 1042–1052, Oct 1993.

[2] O. Tuzel, F. Porikli, and P. Meer, "Region covariance: A fast descriptor for detection and classification," in *Computer Vision–ECCV 2006*, pp. 589–600, Springer, 2006.

[3] M. Swain and D. Ballard, "Color indexing," *International Journal of Computer Vision*, vol. 7, pp. 11–32, November 1991.

[4] F. Porikli, "Integral histogram: a fast way to extract histograms in cartesian spaces," in *Proc. IEEE Computer Society Conf. on Computer Vision and Pattern Recognition*, vol. 1, pp. 829–836, Jun. 2005.

[5] T. Lindeberg, "Scale Invariant Feature Transform," *Scholarpedia*, vol. 7, no. 5, p. 10491, 2012. revision 149777.

[6] H. Bay, A. Ess, T. Tuytelaars, and L. V. Gool, "Speeded-up robust features (surf)," *Computer Vision and Image Understanding*, vol. 110, no. 3, pp. 346 – 359, 2008. Similarity Matching in Computer Vision and Multimedia.

[7] D. Lowe, "Object recognition from local scale-invariant features," in *Proc. of the Seventh IEEE Int. Conf. on Computer Vision*, vol. 2, pp. 1150–1157 vol.2, Sept 1999.

[8] F. Porikli, O. Tuzel, and P. Meer, "Covariance tracking using model update based on lie algebra," in *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, vol. 1, pp. 728–735, IEEE, 2006.

[9] K. Guo, P. Ishwar, and J. Konrad, "Action recognition from video using feature covariance matrices," *Image Processing, IEEE Transactions on*, vol. 22, pp. 2479–2494, June 2013.

[10] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 1, pp. 886–893 vol. 1, June 2005.

[11] M. S., "Mathematical description of the responses of simple cortical cells," *Journal of the Optical Society of America*, vol. 70, no. 11, pp. 1297–1300, 1980.

[12] M. W. Manjunath B.S., "Texture features for browsing and retrieval of image data," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 18, pp. 837–842, August 1996.

[13] I. L. Dryden, A. Koloydenko, and D. Zhou, "Non-euclidean statistics for covariance matrices, with applications to diffusion tensor imaging," *The Annals of Applied Statistics*, vol. 3, no. 3, pp. pp. 1102–1123, 2009.

[14] I. L. Dryden, X. Pennec, and J.-M. Peyrat, "Power euclidean metrics for covariance matrices with application to diffusion tensor imaging," 09 2010.

[15] J. Wang, J. Li, and G. Wiederhold, "Simplicity: semantics-sensitive integrated matching for picture libraries," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 23, pp. 947–963, Sep 2001.

[16] S. Aksoy and R. M. Haralick, "Feature normalization and likelihood-based similarity measures for image retrieval," *Pattern Recognition Letters*, vol. 22, no. 5, pp. 563–582, 2001.

[17] M. A.-M. M. Haghighat, S. Zonouz, "Identification using encrypted biometrics," *Computer Analysis of Images and Patterns, Springer Berlin Heidelberg*, pp. 440–448, 2013.

[18] Y. Lu, I. Cohen, X. S. Zhou, and Q. Tian, "Feature selection using principal feature analysis," in *Proceedings of the 15th international conference on Multimedia*, pp. 301–304, ACM, 2007.

[19] Y. Cui and J. G. Dy, "Orthogonal principal feature selection," 2008.

[20] S. J. Raudys and A. K. Jain, "Small sample-size effects in statistical pattern-recognition - recommendations for practitioners," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 3, pp. 252–264, 1991.

A Example Images