An Automatic Hierarchical Image Classification Scheme

Jing Huang * S Ravi Kumar † Ramin Zabih ‡
Department of Computer Science
Cornell University
Ithaca, NY 14853.

Abstract

Organizing images into semantic categories can be extremely useful for searching and browsing through large collections of images. Not much work has been done on automatic image classification, however. In this paper, we propose a method for hierarchical classification of images via supervised learning. This scheme relies on using a good low-level feature and subsequently performing feature-space reconfiguration using singular value decomposition to reduce noise and dimensionality. We use the training data to obtain a hierarchical classification tree that can be used to categorize new images. Our experimental results suggest that this scheme not only performs better than standard nearest-neighbor techniques, but also has both storage and computational advantages.

1 Introduction

The proliferation of the world-wide web has given easy access to an explosively growing volume of visual data. Unfortunately, this data on the web is both scattered and unorganized, making search and retrieval of information difficult. Large digital libraries, which are built by collecting resources from different locations [20, 18], can make searching relatively easier. Users, however, are not only interested in searching for specific images or video shots. They would also like to browse and navigate through the image corpus. Such requirements have created great demands for effective and flexible systems to manage digital images/videos (e.g., [5, 8, 9, 1, 13, 7, 16]).

Almost all of the above systems generate low-level image features such as color, texture, shape, motion, etc., for image indexing and retrieval. This is partly because low-level features (e.g., color histograms, texture patterns) can be computed automatically and efficiently. The semantics of images, with which users prefer most of their interaction, however, are seldom captured by low-level features. On the other hand, there is no effective method yet to automatically generate good semantic features of an image. One common compromise is to obtain some semantic information through manual annotations. Since visual data contains rich information, and the manual annotation process is quite subjective and ambiguous, it is very difficult to capture the content of an image using words, not to mention the tedious work involved in such a process.

Image Classification. One approach to this problem is to organize the digital library in a meaningful manner using *image classification*. Image classification is the task of classifying images into (semantic) categories based on the available training data. This categorization of images into classes can be helpful both in semantic organizations of digital libraries and in obtaining automatic annotations of images.

The classification of natural imagery is quite hard in general, for, real images from the same semantic class may have large variations (see Figure 1)[1] and images from different semantic classes might share a common background (such as images from "clouds" and "aviation", images from "waves" and "dolphins and whales" in Figure 1). These issues limit the applicability of object-based and knowledge-based approaches.

A common approach to image classification involves addressing the following three issues: (i) image features — how to represent the image, (ii) organization of feature data — how to organize the data, and (iii) classifier — how to classify an image. Acquiring "nice" image features and carefully modeling the feature data are vital steps in this approach.

As noted before, image classification can lead to a semantic organization of a digital database. Though this type of organization can be done in several ways, a hierarchical approach has multi-fold advantages: (i) easy browsing and navigation through the database, (ii) efficient retrieval, and (iii) ergonomically friendly presentation of the database. For instance, WebSEEK, a web image search engine [4], uses hierarchical semantic structure for collecting and searching images from the web. The image categories and hierarchies are preset by human design. An image is classified into one of the classes by first extracting key words from its html tag and then mapping the key words to classes. This approach requires human assistance and depends on the information in the html tag, which may be insufficient or even inaccurate and misleading.

Our Approach. In this paper, we propose a new scheme for automatic hierarchical image classification. We assume a training set of images with known class labels is available. We use an easy-to-compute low-level feature — banded color correlograms — which seems to be effective and efficient for content-based image retrieval [12]. Using banded color correlograms for the training images, we model the feature data using *singular value decomposition* [10]

*Supported by DARPA under contract DAAL01-97-K-0104, monitored by ONR. E-mail: huang@cs.cornell.edu

†Supported by the ONR Young Investigator Award N00014-93-1-0590, the NSF grant DMI-91157199, and the career grant CCR-9624552. E-mail: ravi@cs.cornell.edu

‡Supported by DARPA under contract DAAL01-97-K-0104, monitored by ONR. E-mail: rdz@cs.cornell.edu

[1] The color versions are available at http://www.cs.cornell.edu/home/huang.

and construct a *classification tree*. Once the classification tree is obtained, any new image can be classified easily. Our recursive method for constructing the classification tree is summarized below.

At each level of the classification tree, we aim to choose the best modeling of the training data. We first eliminate the "noise" (or irrelevant variations) from the feature vectors using singular value decomposition (or two-mode factor analysis). This step not only reduces the dimensionality of the feature vectors but also rearranges the feature space to reflect the major correlation patterns in the data and ignores the smaller, less important variations[2]. Using this noise-tolerant SVD representation, we next classify each image in the training data using the nearest-neighbor algorithm with the first neighbor (which is the image itself) dropped (this is similar to leave-one-out cross-validation scheme). Based on the performance of this classification, we then partition the set of classes into two subclasses such that the intra-subclass association is maximized while simultaneously the inter-subclass disassociation is minimized. This is accomplished using *normalized cuts* [19]. Finally, the subclasses and those training images that were correctly classified with respect to the subclasses are worked upon recursively to obtain a hierarchical classification tree, with the hope of improving the classification performance.

Notice that a different SVD representation is used at each level of the tree. This flexibility in our method gives us the freedom to choose the size of the SVD representation as demanded by each level, which in turn is dictated by the characteristics of classes involved.

We test our method on 11 fairly representative classes of Corel images. These eleven image classes are: aviation photography, British motor car collection, Canadian Rockies, cats and kittens, clouds, dolphins and whales, flowers, night scenes, spectacular waterfalls, sunsets around the world, and waves. These images contain a wide range of content (scenery, animals, objects, etc.) and colors.

We test our scheme using banded color correlograms and color histograms as features and compare our method to the nearest-neighbors directly applied to both color features. Our results suggest that this hierarchical scheme is able to perform consistently better than the already effective nearest-neighbor algorithm (see [3]). The classification tree we obtain also conforms with the semantic content of the 11 classes. Our results also suggest that correlograms have more *latent semantic* structures (than histograms) that can be extracted by SVD procedure.

Organization. The rest of the paper is organized as follows: Section 2 briefly describes the previous work in automatic image classification. Section 3 contains a brief description of the banded color correlogram we use in our experiments; Section 4 outlines how to use singular value decomposition to model feature vectors; and Section 5 describes our hierarchical classification method. Section 6 contains our experimental results and Section 7 concludes our discussions.

2 Related Work

Since classification itself is a long-studied research area, different classifiers can be tried on image classification (e.g., k-nearest neighbor, decision trees, Bayesian nets, maximum likelihood analysis, linear discriminant analysis, neural networks, etc.). Not much work has been done on how to represent images (i.e., features) and how to organize features. In the following, we review some previous work in image classification.

Yu and Wolf present a one-dimensional *Hidden Markov Model* (HMM) for indoor/outdoor scene classification. An image is first divided into horizontal (or vertical) segments and each segment is further divided into blocks. Vector quantization techniques is applied to model the color histograms of blocks. These color features are used to train HMM's for a preset standard set of clusters, such as a cluster of sky, tree, river, a cluster of sky, tree, grass, etc. Maximum likelihood classifier is then used to classify an image as indoor or outdoor. The overall results of classification depend on the standard set of clusters which describe the indoor scene and outdoor scene. Generally, it is difficult to enumerate a set to cover a general case such as indoor/outdoor.

The *configural recognition* scheme proposed by Lipson *et. al.* [14] is also a knowledge-based scene classification method. A model template, which encodes the common global scene configuration structure using qualitative measurements, is hand-crafted for each category. An image is then classified to the category whose model template that best matches the image by deformable template matching (which requires heavy computation, despite that the images are subsampled to low resolutions) — the nearest neighbor classification. The average percentage of correct classification on 4 classes of scenery (snowy mountains, snowy mountains with lakes, fields, and waterfalls) is about 64. To avoid the drawbacks of manual templates, a learning scheme that automatically construct scene templates from a few examples is proposed by [17]. The learning scheme was tested on two scene classes and suggested promising results.

Carson *et. al.* [3] propose a new representation for images. Each image is thought to consist of several *blobs*; each blob is coherent in color and texture space[3]. All the blobs in the training data of 14 image categories are clustered into about 180 "canonical" blobs using Gaussian models with diagonal covariance. Each image is then assigned a score vector which measures the nearest distance of each canonical blob to the image. These score vectors are used to train a decision-tree classifier. The results of this method are compared to color histograms with the decision-tree classifier.

Interestingly, the color histograms seem to perform better than blobs. Several explanations were given for this performance degradation: (i) the clustering procedure did a bad job on choosing canonical blobs and the canonical blobs were not interesting enough to distinguish categories; (ii) the trained decision-tree focused too much on irrelevant blobs which did not help in classification; and (iii) the 14 categories have large amount of overlapping which cause the difficulties, such as fields appear in the categories "fields", "horses", and "elephants".

3 Banded Color Correlograms

In this section, we briefly review the banded color correlograms we use in our experiments.

If we treat the color histogram as a probability distribution of colors in an image, we ask the following question: pick any pixel p_1 of in the image $\mathcal{I}$, at distance k away from p_1 pick another pixel p_2, what is the probability that p_2 has the same color as p_1? The answer gives us the conditional probability distribution that depicts the spatial correlation between same color pixels. The color correlogram describes how this spatial correlation of colors changes with distances. We give the formal definitions below.

Let I be an $n_1 \times n_2$ image. The colors in I are quantized into m colors $c_1, \ldots, c_m$. For a pixel $p = (x, y) \in I$, let $I(p)$ denote its color. Let $I_c \triangleq \{p \mid I(p) = c\}$. Thus, the notation $p \in I_c$ is synonymous with $p \in I, I(p) = c$. For convenience, we use the L_∞-norm to measure the distance between pixels, i.e., for pixels $p_1 = (x_1, y_1), p_2 = (x_2, y_2)$, we define $|p_1 - p_2| \triangleq \max\{|x_1 - x_2|, |y_1 - y_2|\}$. We denote the set $\{1, 2, \ldots, n\}$ by $[n]$. The size of I is denoted $|I| = n_1 n_2$.

[2] SVD has been successfully used in latent semantic indexing for document retrieval [6].

[3] This is one kind of color and texture based image segmentation method.

Histogram. The *color histogram* (henceforth histogram) h of I is defined for $i \in [m]$ by

$$h_{c_i}(I) \triangleq \Pr_{p \in I}[p \in I_{c_i}].$$

$h_{c_i}(I)$ thus gives for any pixel in I, the probability that the color of the pixel is c_i. Given the count $H_{c_i}(I) \triangleq |\{p \mid p \in I_{c_i}\}|$, it follows that $h_{c_i}(I) = H_{c_i}(I)/(n_1 n_2)$.

Autocorrelogram. Let a *distance set* D be fixed *a priori* (e.g., $D \subseteq [\min\{n_1, n_2\}]$). Let $d = |D|$. Then, the *autocorrelogram* of I is defined for $i \in [m], k \in D$ as

$$\begin{aligned}\alpha_{c_i}^{(k)}(I) &\triangleq \Pr_{p_1 \in I_{c_i}, p_2 \in I}[p_2 \in I_{c_i} \mid |p_1 - p_2| = k] \\ &= \frac{|\{p_1, p_2 \in I_{c_i} \mid |p_1 - p_2| = k\}|}{H_{c_i}(I) \cdot 8k}.\end{aligned}$$

Given any pixel p of color c_i in the image, $\alpha_{c_i}^{(k)}$ gives the probability that a pixel at a distance k from the given pixel has the same color of p. (The factor $8k$ is due to the properties of L_∞-norm used to compute distance between pixels.) Note that the size of the autocorrelogram is md. Since local correlations between colors are more significant than global correlations in an image, a small value of d is sufficient to capture the spatial correlation.

We now define the *banded autocorrelogram* to be

$$\beta_{c_i}(I) \triangleq \frac{1}{k} \sum_{k'=1}^{k} \alpha_{c_i}^{(k')}(I).$$

This measure computes the local *density* of color c_i's correlation with itself, thus suggesting one kind of a local structure of colors. Note that the size of banded autocorrelogram is m, i.e., the same as that of histogram. We use $\beta(I)$ denote the banded autocorrelogram of I, treated as vectors in an m-dimensional space.

We will use the L_1 (or the city-block) distance measure for comparing histograms and banded autocorrelograms because it is simple and robust. For simplicity, we will address banded autocorrelograms merely as correlograms for the remainder of the paper.

4 Singular Value Decomposition

In this section, we briefly review the singular value decomposition that we use for organizing image feature vectors.

Without loss of generality, let $m \geq n$. For an $m \times n$ matrix A, the *singular value decomposition* of A is given by $A = U\Sigma V^T$ (see [10]), where

(i) U is an $m \times n$ matrix, Σ, V are $n \times n$ matrices;

(ii) U, V are column orthonormal i.e., $U^T U = V^T V = I_n$; and

(iii) $\Sigma = \mathrm{diag}(\sigma_1, \ldots, \sigma_r, 0, \ldots, 0)$ where $r = \mathrm{rank}(A)$ and the *singular values* are $\sigma_1 \geq \sigma_2 \ldots \geq \sigma_r > 0$.

The first r columns of U and V together with the non-zero singular values actually are the eigenvectors and the r non-zero eigenvalues of AA^T and $A^T A$ respectively. Several efficient algorithms exist to compute the SVD of a matrix, especially if the matrix is known to be sparse.

The SVD of a matrix can be used to obtain lower rank approximations of the matrix. If we take the first k columns of U, V (denoted $U_{[k]}, V_{[k]}$) and the leading $k \times k$ submatrix of Σ (denoted $\Sigma_{[k]}$), and define

$$A_k \triangleq U_{[k]} \Sigma_{[k]} V_{[k]}^T = \sum_{i=1}^{k} U_i \Sigma_{i,i} V_i^T,$$

then, A_k is the best rank k approximation to A, i.e.,

$$\min_{\mathrm{rank}(B)=k} |A - B|_2 = |A - A_k|_2 = \sigma_{k+1}.$$

This property of the SVD helps to obtain a good trade-off between the quality of approximation and the size of the approximation (i.e., k). (To compute A_k, we use the Matlab built-in function `svd`.)

The advantages of SVD are nicely exploited in *latent semantic indexing* (LSI) for document retrieval [6]. The SVD, in some sense, derives the underlying structure that is hidden in A. The approximation A_k can be thought of dampening the "noise" and unreliability that is present in the original matrix A. When SVD is applied to feature vectors, it not only eliminate the "noise" in the feature vectors, but also reduce the dimension of the feature when $k < m$.

We outline our approach of using SVD with correlograms. Let $\mathcal{I} = \{I_1, \ldots, I_n\}$ denote the set of training images and let m be the number of color quantizations. We define the matrix $A_{i,j}(\mathcal{I}) \triangleq \beta_{c_i}(I_j)$. We compute the SVD of $A(\mathcal{I})$ to be $A(\mathcal{I}) = UDV^T$. Let $A_k = U_{[k]} \Sigma_{[k]} V_{[k]}^T$ be an approximation to A. We can choose $U_{[k]}$ as the basis for the new k-dimensional feature space. Then $V_{[k]}$ is the new representation for the correlograms in this reduced feature space. When we have a new image that needs to be classified, we first compute its correlogram q, then project q onto the reduced feature space by computing

$$q' = q \cdot U_{[k]} \cdot \Sigma_{[k]}^{-1}$$

Now, the question is: how to choose k for the approximation? We use the following heuristic to pick the k. Note that we want to find the best approximation A_k such that the SVD representation of correlograms give the best classification results using nearest-neighbor rule. Instead we compute the classification for each k between the number of classes (i.e., c) to an upper limit k^* and choose the best k in this range. Now, we show how to choose k^*. Notice that the singular values of A correspond to the eigenvalues of AA^T, which is the correlation matrix of local color density for the training images. We set k^* be the k^*-th biggest eigenvalue that is within 2% of the maximum eigenvalue, i.e., we ignore those correlations whose values are less than 2% of the maximum correlation[4].

Note that in the above SVD method, histograms can be used instead of correlograms. We will see (Section 6) that the performance with correlograms is much better than with histograms.

5 The Hierarchical Classification Scheme

Image classification is the problem of classifying images into known semantic classes. Let $\mathcal{C} = \{C_1, \ldots, C_c\}$ be the image classes known *a priori*. We assume that we have a set $\mathcal{S}$ of training images whose class membership is known and $\mathcal{T}$ of images that need to be classified. We want build a classification tree from training images. At each level of the classification tree, we aim to choose the best modeling of the training data. We first use SVD to eliminate "noise" from the training data as described in Section 4. We then classify each image in the training data using the nearest-neighbor algorithm with the first neighbor dropped (similar to the leave-one-out cross-validation scheme). Based on the performance of this classification, we then split the classes into two subclasses such that the intra-subclass association is maximized while simultaneously the inter-subclass disassociation is minimized. This is accomplished using *normalized cuts* [19]. Finally, the subclasses and those training images that were correctly classified with respect to the subclasses are worked upon recursively to obtain the hierarchy

[4] There is no good heuristic for choosing k. The rule of thumb is finding the k that gives the best performance [2].

in the classification tree, with the hope of improving the classification performance.

5.1 Confusion Matrix

We construct the matrix $A(\mathcal{S})$ as indicated in Section 4 and compute its SVD: $A(\mathcal{S}) = U\Sigma V^T$. Then we choose the best approximation A_k that gives the best classification of $\mathcal{S}$ on itself. The details are the following.

For an image $I \in \mathcal{S}$, and $C(I)$ be the class of I, let $\beta_k(I)$ denote the k-dimensional reduced SVD representation of I. We consider each $I \in \mathcal{S}$ as a query and obtain the class $C'(I)$ where

$$C'(I) \triangleq C(\arg \min_{J \in \mathcal{S} \setminus \{I\}} \{|\beta_k(I) - \beta_k(J)|\}).$$

In other words, $C'(I)$ is the class assigned by the nearest neighbor classification when all images other than I itself is considered. Intuitively, this procedure helps to find the best association patterns between classes from SVD.

Now, the $c \times c$ *confusion matrix* M is then defined by

$$M_{i,j} = \text{size of } \{I \mid C(I) = C_i, C'(I) = C_j\}$$

The diagonal entries of M are the number of images that are correctly classified, while the off-diagonal entries are the misclassifications. The average percentage of correct classification is just the sum of the diagonal entries ($\mathrm{trace}(M)$) divided by the size of $\mathcal{S}$.

5.2 Normalized Cuts

We now show how to partition the confusion matrix M on basis of maximizing the inter-class association and minimizing the intra-class disassociation simultaneously. First, we review some basic definitions from graph theory.

Given a weighted graph $G = \langle V, E \rangle$ with $w(u, v)$ being the weight of an edge (u, v), the *mincut* is defined to be a partition of $V = V_1 \cup V_2$ such that

$$\mathrm{cut}_w(V_1, V_2) \triangleq \sum_{(u,v) \in V_1 \times V_2} w(u, v)$$

is minimized. Mincuts can be computed in polynomial time using network flow techniques.

The confusion matrix M defines a natural directed graph. The mincut in this graph corresponds to a partition of the classes into M_1 and M_2 such that the number of misclassification among these classes is minimized. A partition of M according to the mincut, however sometime favors cutting small sets [22], i.e., one of V_1, V_2 is very small. This problem is considered in [19] where *normalized cuts* are introduced.

Formally, the normalized cut is given by the best partition of $V = V_1 \cup V_2$ that minimizes

$$\mathrm{ncut}_w(V_1, V_2) \triangleq \mathrm{cut}_w(V_1, V_2) \left(\frac{1}{\mathrm{cut}_w(V_1, V)} + \frac{1}{\mathrm{cut}_w(V_2, V)} \right).$$

The partition based on normalized cut is shown to have the property that is minimizes the disassociation between the groups and maximizes the association within the group.

Define the diagonal matrix $M'_{i,i} = \sum_j M_{i,j}$. Normalized cuts can be computed reasonably well and efficiently by computing the second smallest eigenvalue of the system defined by $(M - M')x = \lambda M' x$ and using some additional heuristics. The details can be looked up in [19].

We use normalized cuts to partition M into M_1 and M_2, in effect obtaining a partition of the classes $\mathcal{C}$ into $\mathcal{C}_1$ and $\mathcal{C}_2$.

5.3 Classification Tree

Using the normalized cuts, we can build the classification tree recursively. Given the original set of classes $\mathcal{C}$, we compute the partition $\mathcal{C} = \mathcal{C}_1 \cup \mathcal{C}_2$ based on normalized cuts. We define $\mathcal{S}_1 = \{I \in \mathcal{S} \mid C'(I) \in \mathcal{C}_1 \text{and } C(I) \in \mathcal{C}_1\}$ and $\mathcal{S}_2 = \{I \in \mathcal{S} \mid C'(I) \in \mathcal{C}_2 \text{and } C(I) \in \mathcal{C}_2\}$. In this way, the images that are misclassified across $\mathcal{S}_1$ and $\mathcal{S}_2$ are not considered from now on. We then recursively work on classifying $\mathcal{S}_1$ (resp. $\mathcal{S}_2$) with $\mathcal{C}_1$ (resp. $\mathcal{C}_2$) as the set of classes.

The algorithm is described in its entirety in Figure 2.

5.4 Trimming

Sometimes, the performance of the classification on the training data does not always improve level by level using reduced SVD representations. This is because some variations that are important to a set of classes may be removed by the SVD reduction. In this case, it does not pay-off to recursively split such classes. Notice that this scenario can be detected automatically by comparing the performance of the tree before and after trimming on the training set. More precisely, we perform a trimming procedure on the tree we obtain from the algorithm in the following manner: if the classification correctness of a node in the tree is higher than that of its two children, then we trim both the children; otherwise we keep the child with the higher correctness than the node itself and trim the other child. For instance, Figure 3 shows a classification tree (corresponding to our sample set) with the trimmed portions marked.

6 Experiments and Results

6.1 Experiments

We choose 11 image classes from Corel collections: aviation photography, British motor car collection, Canadian Rockies, cats and kittens, clouds, dolphins and whales, flowers, night scenes, spectacular waterfalls, sunsets around the world, and waves (for some samples, see Figure 1). These images contain a wide range of content (scenery, animals, objects, etc.), colors, and lighting conditions. We delete some images in each class which are not consistent with the rest of the class (as in [3]) and leave 90 images in each class. Since we use the nearest-neighbor rule with the classification tree, we want to make sure that the color distributions of training images and test images are more or less the same. Therefore, we randomly shuffle the images in each class and take 70 images as the training set and the rest 20 images as the test set. By doing so three times (to ensure fairness), we obtain 3 sets of training data and test data.

To compute color histograms and color correlograms, we quantize the RGB color space into $8 \times 8 \times 8 = 512$ colors (3 bits for each color channel)[5]. This level of quantization is good enough for the SVD to extract the underlying structure, while not being too big (unlike 6912 colors used in [3]) so as to affect efficiency.

6.2 Results

We test both color correlograms and color histograms on the hierarchical classification approach and compare the hierarchical approach with the nearest-neighbor classification. The three classification trees from three training sets are more or less the same and are consistent with the color content of the 11 classes. We only present the tree from the first data set (Figure 3). For the sake of simplicity, we abbreviate names for the 11 classes: aviation (A1), British motors (B1), Canadian Rockies (C1), cats and kittens (C2), clouds (C3), dolphins and whales (D1), flowers (F1), night scenes (N1), spectacular waterfalls (S1), sunsets around the world (S2),

[5] We also tried the HSV color space. The results don't change much.

and waves (W1). From the classification tree, we see that A1 and C3 share the same parent because of the same sky background; similarly, W1, S1, and D1 are grouped together because of the same water background.

The confusion matrices for different methods are shown in Table 1, Table 2, and Table 3. The classification behavior of the classification tree is quite different from the nearest-neighbor. The classification tree is better than the nearest-neighbor in that: (i) the overall number of misclassification between classes is smaller and (ii) the overall number of correct classification is larger.

The average percentage of correctness of the three test sets is summarized in Table 4. The results show that the hierarchical method is consistently better than the nearest-neighbor classification, and the color correlogram is consistently better than the color histogram.

Using the simple nearest-neighbor classification, the correlogram performs 3% better than the histogram; using the classification tree, the correlogram performs 21% better than the histogram. Using the classification tree, the correlogram improves 3% over the nearest-neighbor; it improves 7% over the nearest-neighbor on the histogram. Note that the average data size of the SVD representations is about 15, 3% of the original size. The average number of non-leaf nodes in the classification trees is five after trimming. Therefore the computation and storage of the data for the classification saves about 85%, which is significant.

Remark. We notice from the results that the color histogram performs consistently worse with the classification tree than with the nearest-neighbor, while the color correlogram performs consistently better with the classification tree than with the nearest-neighbor. This suggests that correlograms have an underlying "latent semantic" structure (local color density). Color histograms do not seem to have such a property.

	Nearest-Neighbor			Classification Tree (Trim)		
	1	2	3	1	2	3
Hist	0.786	0.746	0.786	0.696	0.677	0.668
Corr	0.818	0.800	0.786	**0.850**	**0.805**	**0.823**

Table 4: Correctness classification on three data sets.

7 Conclusions and Future Work

We propose using the singular value decomposition with banded color correlogram to extract a "latent semantic" structure of images for classification into categories. Our tests on 11 image classes show that our method using this scheme and a classification tree not only performs better than the nearest-neighbor classification but also saves much computation and data storage. In addition, the results also suggest that the correlogram is more suitable for the image classification task that the color histogram.

It will be interesting to use *feature weighting* techniques [21] to further assist SVD to get latent semantic structures from training data.

References

[1] J. Bach, C. Fuller, A. Gupta, A. Hampapur, B. Horowitz, R. Humphrey, R. Jain, and C. Shu. Virage image search engine: an open framework for image management. *Storage & Retrieval for Image and Video Databases IV, Proc. of SPIE 2670*, 1996.

[2] H. Borko and M. Bernick. Automatic document classification. *Journal of the ACM*, 9:512–521, 1962.

[3] C. Carson, S. Belongie, H. Greenspan, and J. Malik. Color- and texture-based image segmentation using EM and its application to image querying and classification. Manuscript, submitted to *Pattern Analysis and Machine Intelligence*.

[4] S-F. Chang, J.R. Smith, M. Beigi, and A. Benitez. Visual information retrieval from large distributed online repositories. *Communications of the ACM*, 40:12, pp. 63–67, 1997.

[5] Content-based image retrieval systems. *IEEE Computer*, 28(9), 1995.

[6] S. Deerwester, S. Dumais, G. Furnas, T. Landauer, and R. Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41:391–407, 1990.

[7] Excalibur Tech. Corp. `http://vrw.excalib.com`.

[8] M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Q. Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steele, and P. Yanker. Query by image and video content: The QBIC system. *IEEE Computer*, 28(9):23–32, September 1995.

[9] W. Niblack, *et al.* Updates to the QBIC System. Proc. of SPIE 3312, pp.150–161, 1998.

[10] G. Golub and C. Van Loan. *Matrix Computations.* The Johns Hopkins University Press, 1989.

[11] J. Huang, S. R. Kumar, M. Mitra, W. J. Zhu, and R. Zabih. Image indexing using color correlograms. *Proc. of 16th IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 762–768, 1997.

[12] J. Huang, S. R. Kumar, M. Mitra, and W. J. Zhu. Spatial color indexing and applications. *Proc. of 8th Intl. Conference on Computer Vision*, 1998.

[13] Magnifi, Inc. `http://www.magnifi.com`.

[14] P. Lipson, E. Grimson, and P. Sinha. Configuration based scene classification and image indexing. *Proc. of 16th IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 1007–1013, 1997.

[15] G. Pass and R. Zabih. Histogram refinement for content-based image retrieval. *IEEE Workshop on Applications of Computer Vision*, pp. 96–102, 1996.

[16] A. Pentland, R. Picard, and S. Sclaroff. Photobook: Content-based manipulation of image databases. *Intl. Journal of Computer Vision*, 18(3):233–254, 1996.

[17] A. L. Ratan and W. E. L. Grimson. Training templates for scene classification using a few examples. *Proc. of IEEE Workshop on Content-Based Access of Image and Video Libraries*, pp. 90–97, 1997.

[18] S. Sclaroff, L. Taycher, and M. La Cascia. ImageRover: a content-based image browser for the world wide web. *Proc. of IEEE Workshop on Content-Based Access of Image and Video Libraries*, pp. 2–9, 1997.

[19] J. Shi and J. Malik. Normalized cuts and image segmentation. *Proc. of 16th IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 731–737, 1997.

	A1	C3	C1	D1	W1	B1	C2	S1	F1	N1	S2
A1	**17**	0	0	3	0	0	0	0	0	0	0
C3	2	**15**	0	0	0	0	2	0	0	0	1
C1	0	2	**16**	2	0	0	0	0	0	0	0
D1	0	1	0	**17**	0	0	1	1	0	0	0
W1	0	1	1	4	**13**	0	0	1	0	0	0
B1	0	0	0	0	0	**20**	0	0	0	0	0
C2	0	0	0	0	0	0	**20**	0	0	0	0
S1	0	1	0	2	0	0	0	**17**	0	0	0
F1	1	0	0	0	0	1	1	0	**17**	0	0
N1	0	0	0	0	0	0	1	0	0	**18**	1
S2	0	0	0	0	0	1	0	0	2	0	**17**

Table 1: Class confusion matrix for trimmed classification tree (correlogram).

	A1	C3	C1	D1	W1	B1	C2	S1	F1	N1	S2
A1	**16**	1	0	2	0	0	1	0	0	0	0
C3	1	**14**	0	0	0	0	1	2	0	0	1
C1	0	0	**15**	5	0	0	0	0	0	0	0
D1	0	1	0	**19**	0	0	0	0	0	0	0
W1	0	0	0	2	**17**	0	0	1	0	0	0
B1	0	0	0	0	0	**14**	4	2	0	0	0
C2	0	0	0	0	0	0	**20**	0	0	0	0
S1	0	1	0	2	1	0	0	**16**	0	0	0
F1	1	0	0	0	0	0	2	0	**14**	1	2
N1	0	0	0	0	0	0	1	0	0	**19**	1
S2	0	2	0	0	0	0	0	0	0	2	**16**

Table 2: Class confusion matrix for the nearest-neighbor classification (correlogram).

	A1	C3	C1	D1	W1	B1	C2	S1	F1	N1	S2
A1	**14**	0	2	0	1	0	2	1	0	0	0
C3	0	**13**	1	2	0	0	3	1	0	0	1
C1	0	1	**16**	0	2	0	1	0	0	0	0
D1	0	0	0	**17**	2	0	1	0	0	0	0
W1	0	0	0	3	**16**	0	0	1	0	0	0
B1	0	0	0	0	0	**17**	1	1	0	1	0
C2	0	0	0	0	0	0	**20**	0	0	0	0
S1	0	1	0	0	2	1	0	**16**	0	0	0
F1	0	0	0	0	0	1	1	0	**18**	0	0
N1	0	0	0	0	0	0	1	0	0	**13**	6
S2	0	2	0	0	0	0	0	3	0	2	**13**

Table 3: Class confusion matrix for the nearest-neighbor classification (histogram).

[20] J. Smith and S-F. Chang. Visually searching the Web for content. *IEEE Multimedia 4*, 3, pp. 12–20, 1997.

[21] D. Wettschereck, D. W. Aha, and T. Mohri. A review and evaluation of feature weighting methods for a class of lazy learning algorithms. *Artificial Intelligence Review*, Special issue on lazy learning algorithms, 1997.

[22] Z. Wu and R. Leahy. An optimal graph theoretic approach to data clustering: theory and its application to image segmentation. *Pattern Analysis and Machine Learning*, 11:1101–1113, 1993.

[23] H. Yu and W. Wolf. Scenic classification methods for image and video databases. *SPIE International Conference on Digital Image Storage and Archiving Systems*, 2606:363–371, 1995.

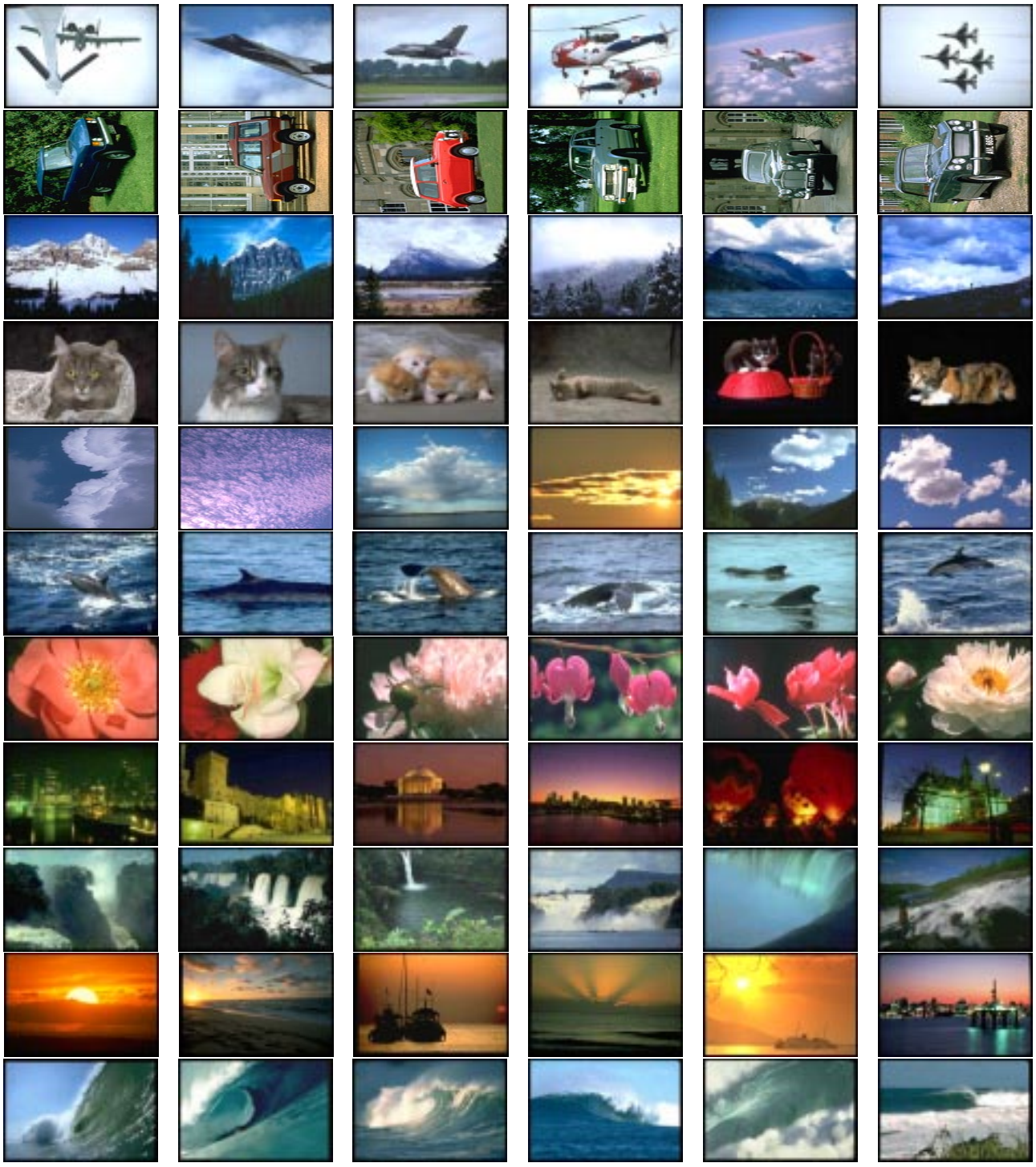

Figure 1: Sample images from various classes.

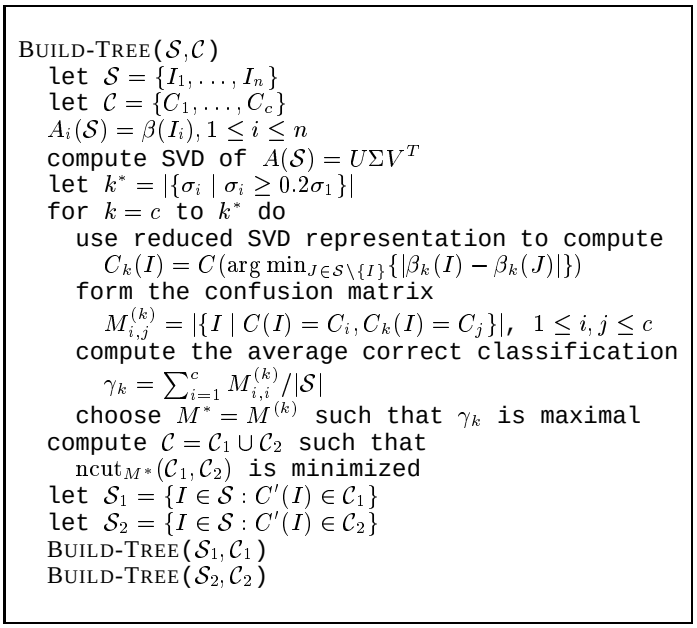

Figure 2: The Training Algorithm

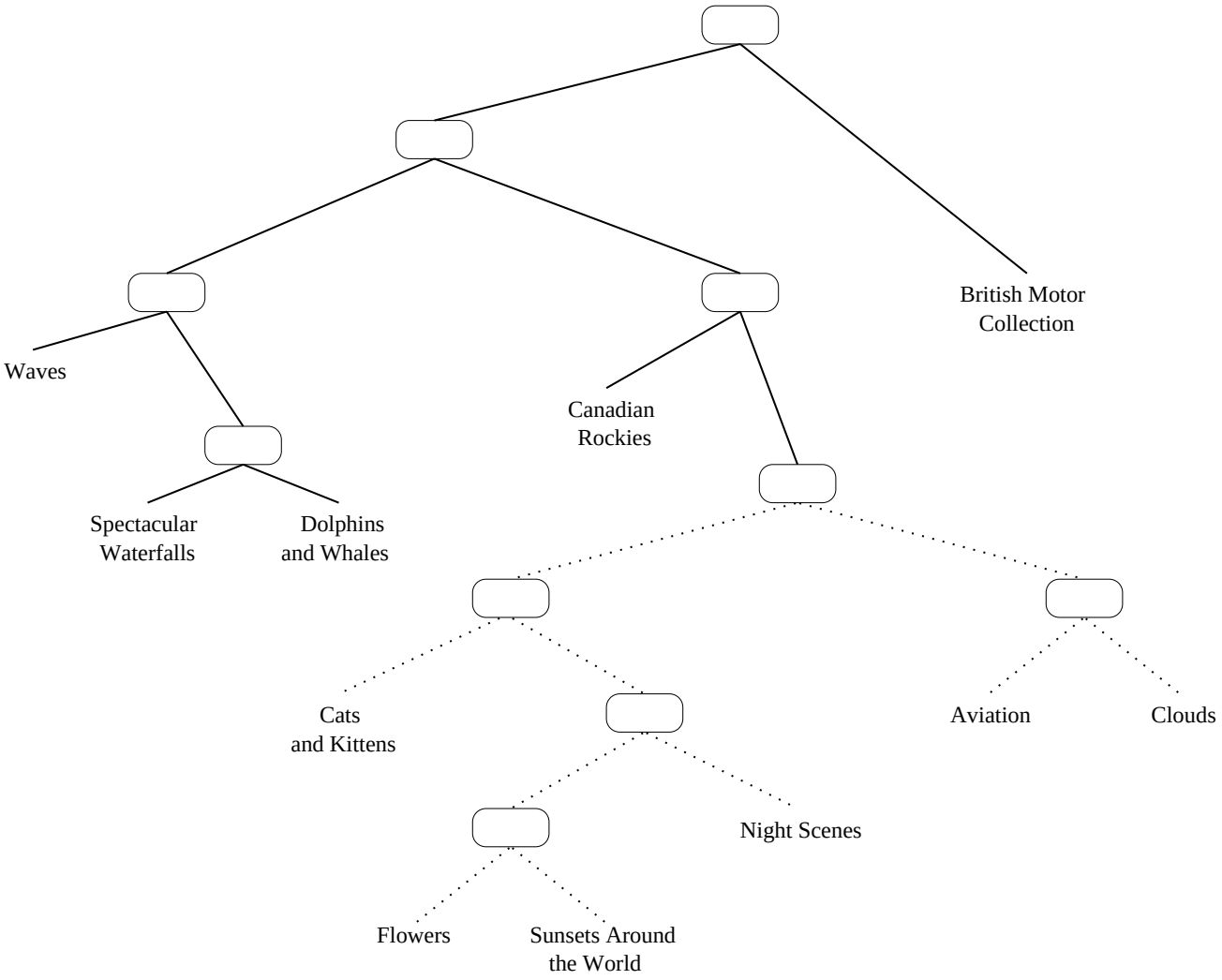

Figure 3: The classification tree obtained from the first training set using correlograms. The dotted lines indicate the trimmed portions.