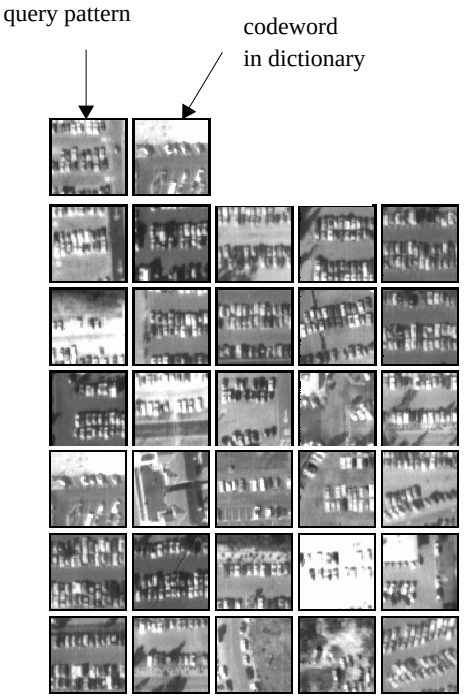
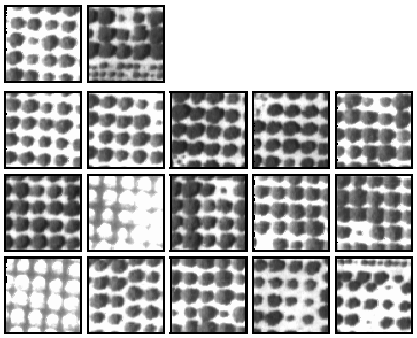
(a)

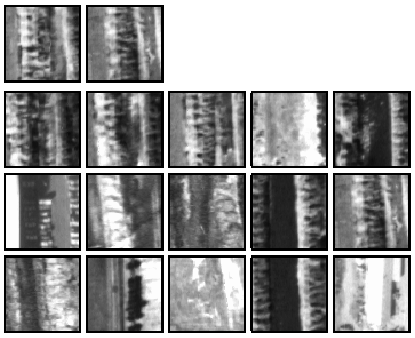
selected region of interest which is shown as query pattern in (b)



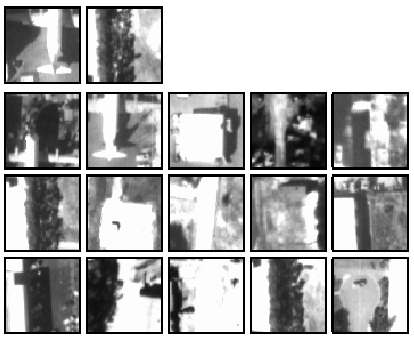
(b)



(c)



(d)



(e)

FIGURE 7. (a) shows an image from the database, from which a region of interest (denoted by the arrow) is selected to search. (b) is the expanded version of the corresponding region (64x64), and its associated codeword, and the best 24 matching patterns are displayed. The retrieved sub-images are from different airphotos in the database. (c)-(e) show three other retrieval examples.

FIGURE 5. A manually labeled air photo.

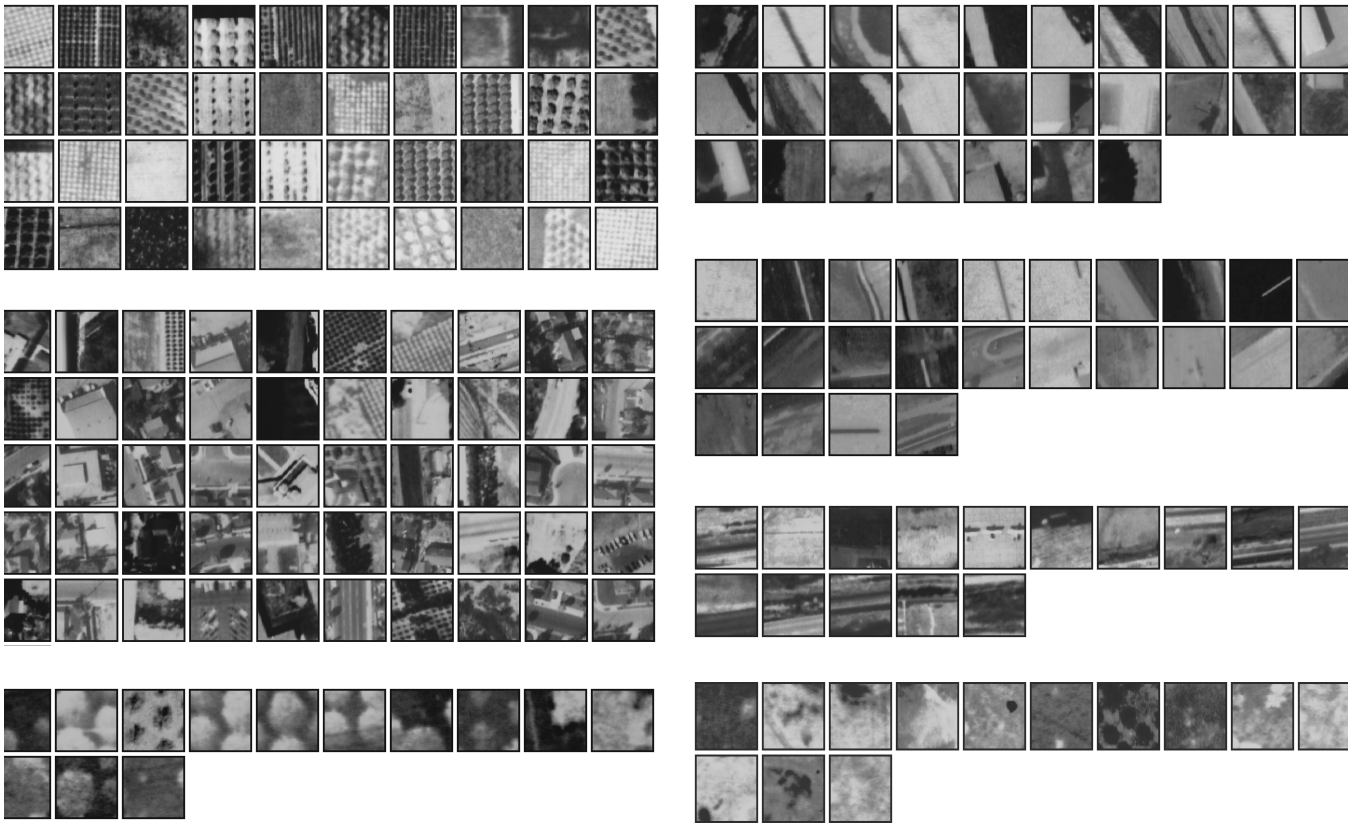


FIGURE 6. Examples of the texture codewords obtained from the training data. The patterns inside each block belong to the same similarity class.

Summarizing, we have proposed a new approach to image indexing which makes use of pattern recognition and vector quantization techniques to design a texture dictionary which is then used for search and retrieval. Conceptually, this is similar to text based search using key words, and instead of keywords we have a texture dictionary to represent potential patterns of interest. Initial results on a large number of airphotos are very encouraging. We plan to provide a detailed experimental evaluation of this algorithm on some standard image databases (such as the Brodatz texture album). A prototype browsing tool using these ideas is under development.

Acknowledgments: This research was in part funded by the UCSB Alexandria Digital Library project (NSF Grant Number IRI94-11330) and by the Lawence Livermore National Laboratory. We also thank Christoph Fischer for providing us with the airphotos, and Amr El Abbadi and Divy Agrawal for many useful discussions.

5 REFERENCES

[1] B. S. Manjunath, W. Y. Ma, "Texture features for browsing and retrieval of image data," Technical report CIPR 95-06, Univ. of California at Santa Barbara, July 1995.

[2] A. D. Alexandria, W. Y. Ma, A. El Abbadi, B. S. Manjunath, "Adaptive filtering and indexing for image databases," Proc. of SPIE on Storage and Retrieval of Image and Video Databases - III, pp. 12-23, San Jose, CA, Feb. 1995.

[3] T. Kohonen, "Self-organized formation of topologically correct feature maps," Biological Cybernetics 43, pp. 59-69, 1982.

[4] T. Kohonen, "The self-organizing map," Proc. IEEE, vol. 78, no 9, pp. 1464-1480, 1990.

[5] S. Haykin, Neural Networks, Macmillan Publishing Comp, 1994.

[6] A. Gersho, R. M. Gray, Vector Quantization and Signal Compression, Kluwer Academic Publishers, 1992.

[7] I. Katsavounidis, C.-C. Jay Kuo, and Z. Zhang, "A new initialization techniques for VQ codebook design," Proc. of the 28th Asilomar conf. on Signal, System & Computers, Pacific Grove, CA, pp. 706-710, 1994.

3. Use the centroid condition to update the codebook $C_{m+1} = \{\text{centriod}(R_i);\ i = 1, 2, \ldots, N\}$, which is the optimal reproduction codebook for the cells just found.

4. Compute the average distortion for C_{m+1}, which is defined as $\sum_{i=1}^{N} \sum_{x \in R_i} \|x - y_i\|$. If it has changed by a small amount since the last iteration, then stop. Otherwise set $m + 1 \rightarrow m$ and go to step 2.

The initialization technique used here is based on [7]. Notice that the above algorithm is used to construct a code book for each of the M classes.

The number of codewords under each sub-tree could be different, and it can depend on the number of training data which fall into the particular sub-tree. Typically, the larger a class size is, the larger the code book size. For visualization purposes, image patterns which are closest to the codewords are used to represent the corresponding code. This set of codewords (and the associated patterns) now form the texture dictionary.

After the texture dictionary is designed, all the images in the database are processed through this dictionary at the ingest time. Each sub-image will be identified and linked to only one codeword in the texture dictionary. When users select one particular image pattern as a query, the system will first extract the texture feature associated with this pattern and searches through the indexing trees to find out the best matching codeword. All the image patterns linked to this codeword will then be retrieved as the candidate matches. At the final stage, a simple Euclidean distance measure on the feature vectors is used to rank-order the similarity with the query pattern, and the best matches are displayed to the user.

4 EXPERIMENTAL RESULTS

We present here some preliminary experimental results which demonstrate the various concepts. The image data used are mostly airphotos of the Santa Barbara area, and each image is about 5400x5400 pixels. A subset (about 25%) of the airphotos is used during the training stage. The training set labels are created manually based on visual similarity. These images are partitioned into smaller 64x64 subimages for texture feature extraction. A total of 30 Gabor filters (six orientations and five scales) are used in this feature extraction stage, resulting in a feature vector of dimension 60. Figure 5 shows an example where two large homogenous region are manually labeled.

In constructing the first level of the indexing tree, we use a Kohonen map with 900 neurons organized on a two dimensional lattice. The number of neurons used was empirically determined after considering the number of different manually classified regions in the training set. The image patterns with texture features closest to the centroids of the codebook are used to visually represent the codewords. Figure 6 shows some of these iconic codewords. As we can see, the patterns within the same class are visually more similar to each other.

Once the dictionary of texture patterns was constructed, all the images in the database are processed and two-way links between each of the subimages and the corresponding codewords are built. Figure 7 shows some retrieval results. An image browsing tool is currently being implemented to facilitate searching the Alexandria Digital Library project at UCSB.

3.1.1 Texture feature extraction

The texture features used in this paper are based on multiresolution Gabor wavelet decomposition. The image pattern is filtered through a bank of orientation and frequency selective Gabor filters. The mean and standard deviation associated with each filter output are used to construct the feature vectors. The details about this feature extraction algorithm and a comparison with different texture features for database applications can be found in [1]. In the following implementation, we use five scales and six orientations for Gabor filters, resulting a feature vector of length 60.

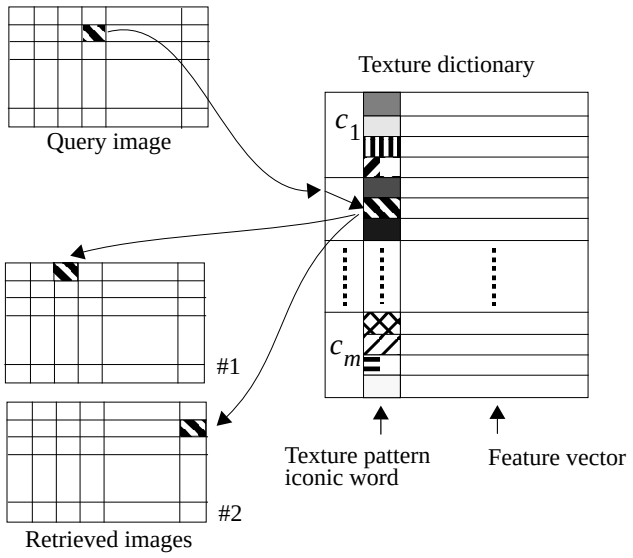


FIGURE 4. A texture dictionary with iconic representations for content-based image indexing.

3.1.2 Codebook design

Following the feature extraction, we have the two layered Kohonen map which assigns a class label (indicated by the active neuron in the output layer). In order to partition the feature space within a given class and to design a code book of patterns for the texture dictionary, we suggest the use of vector quantization. Let M be the number of class labels and we are interested in constructing a tree structured codebook for each of these classes. For this purpose, we use the generalized Lloyd algorithm (GLA) [6]. The algorithm can be summarized as follows:

1. Begin with an initial codebook C_1. set $m = 1$.

2. Given a codebook $C_m = \{y_i;\ i = 1, 2, \ldots, N\}$ obtained from the mth iteration, find the optimal partition of the feature space, that is, use the nearest neighbor condition to form the nearest neighbor cells:

$$R_i = \{x : d(x, y_i) \le d(x, y_j);\ \text{all}\ j \ne i\}.$$

3 PATTERN DICTIONARY AND INDEXING

An important issue in the management of large image databases is the catalog component which indexes into the database. As mentioned earlier, traditional database structures do not generalize well for large dimensional feature space. In particular, these techniques can not be used for dimensions exceeding 20, and typical image feature dimensions are much larger [2]. We propose here the use of a pattern dictionary model to browse through the image data.

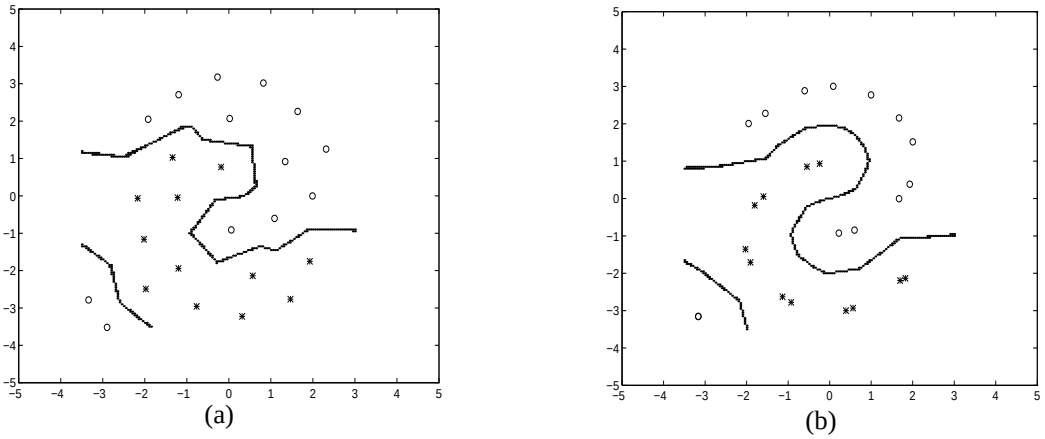


FIGURE 3. (a) is the networks after using the Kohonen map and the supervised learning by majority voting based on the features of Figure 1. (b) shows the neuron positions and decision boundaries after LVQ. “o” and “⋆” represent the neurons with class label 1 and 2, respectively.

Conceptually, the pattern dictionary model can be visualized as an image counterpart of the traditional thesaurus for text search. This pattern dictionary consists of a large collection of feature vectors which are associated with the centroids of the clusters found by the two stage learning algorithm described in the previous section. Each codeword in the dictionary will have a corresponding iconic representation to help users visualize the code words. This scheme reduces the search complexity by providing a tree-structured indexing while preserving the similarity between patterns. The details of this algorithm are described below.

3.1 Texture dictionary design

The dictionary is created using real images taken from the UCSB ADL project database and consists of airphotos of Santa Barbara and nearby regions. The data set consists of large agricultural areas thus providing good textured image data. A subset of the image data is manually labeled and used to train the classification network described in the previous stage. For each of the clusters identified by the neurons in the output layer, a codeword is computed. In designing this code-book, we use a tree structured vector quantization technique. Figure 4 illustrates the overall view of this strategy.

At the time of image data ingest, the images are partitioned into a number of small sub-images for extracting texture features, then these features are used to search the best codeword in the texture dictionary. A two-way link between the matched codeword and the corresponding sub-images is then created and stored as the image meta-data for future search and retrieval.

ing is used to fine tune the network parameters. After the second stage of learning, each output neuron will have an associated class label.

2.2.2 Stage II: Supervised learning using learning vector quantization (LVQ)

The first phase of supervised learning can be done by presenting a number of training feature vectors with known classification to the network obtained in the previous section, and assigning the neurons to different classes by majority voting. Let the class label associated with the ith neuron be represented by C_i. However, the Kohonen map is mainly intended to approximate input feature vectors, or their probability density function. As such, it may not be appropriate to use it directly in defining the decision boundaries. A fine tuning of the network is necessary to improve the classification accuracy. Figure 3(a) shows the results after using the Kohonen map and the supervised learning by majority voting based on the features of Figure 1. Note that in this example we consider the class 1 and class 3 as the same class. As we can see, the neurons approximate the distribution of the feature vectors very well, but they may not be the best for use as decision boundaries.

In the second phase of the supervised learning, the learning vector quantization (LVQ) algorithm [4][5] is used to train the network again in order to move the centroids towards the Bayes decision boundary. In our experiments the type three algorithm (LVQ3) is used to update the weights. The LVQ3 algorithm can be described as follows:

Let $\boldsymbol{m}_i$ and $\boldsymbol{m}_j$ be the two closest weight vectors to a given input feature vector $\boldsymbol{x}$. Let $C(\boldsymbol{x})$ be the known class label associated with $\boldsymbol{x}$. Let $d_i = \|\boldsymbol{x} - \boldsymbol{m}_i\|$ be the distance between the input vector and the ith weight vector. Define w to be the width of the window and let $\gamma = (1-w)/(1+w)$. The input vector $\boldsymbol{x}$ is considered to be within the window and the following updates are computed if the relative distances satisfy the following condition:

$$\min\ (d_i/d_j, d_j/d_i) > \gamma. \tag{3}$$

- Case 1: $C_j = C(x)$ and $C_j \neq C_i$, then

$$\begin{aligned} \boldsymbol{m}_i(n+1) &= \boldsymbol{m}_i(n) - \alpha(n)\,[\boldsymbol{x}(n) - \boldsymbol{m}_i(n)] \\ \boldsymbol{m}_j(n+1) &= \boldsymbol{m}_j(n) + \alpha(n)\,[\boldsymbol{x}(n) - \boldsymbol{m}_j(n)] \end{aligned} \tag{4}$$

- Case 2: $C_j = C(x) = C_i$, then

$$\boldsymbol{m}_k(n+1) = \boldsymbol{m}_k(n) + \varepsilon\alpha(n)\,[\boldsymbol{x}(n) - \boldsymbol{m}_k(n)], k \in \{i, j\}, \tag{5}$$

Since this is a fine-tuning process. the learning rate should begin with a fairly small value (about 0.02 in the experiments) and gradually decrease to zero. In our experiments $w = 0.2$ and $\varepsilon = 0.3$.

Recall that Figure 3(a) represents the topological map after the first stage of learning. Figure 3(b) shows the state of the map after the second stage of LVQ. As we can see, the decision boundaries represented by the neurons are in a better position than in Figure 3(a), and more suitable for computing the intra-cluster distances.

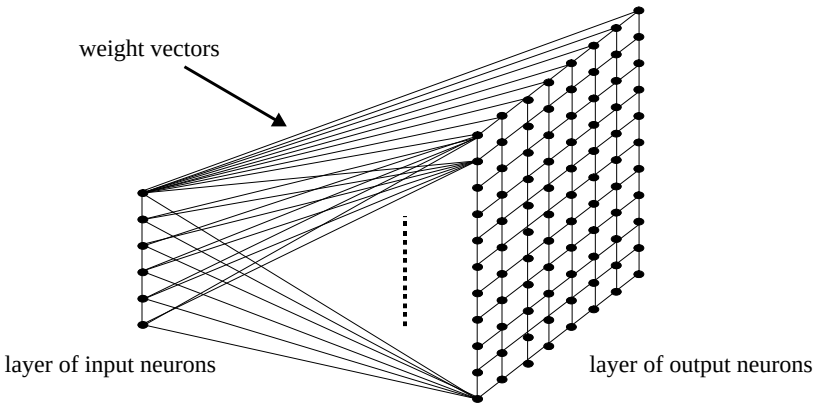


FIGURE 2. Architecture of the Kohonen map

The weights in the Kohonen map are adjusted based on competitive learning; the output neurons of the network compete among themselves in a winner-takes-all representation. Each neuron in the output layer represents a unique cluster of image patterns. For any given input feature vector, only one active neuron can exist in the output layer. The end result is that the feature space is now partitioned into a number of distinct clusters.

The training of the network is performed by randomly presenting a feature vector $\boldsymbol{x}$ to the input layer of the network and adjusting the connection weight vectors $\boldsymbol{m}_i$ according to the updating rules given below. At the beginning stage, all the weight vectors $\boldsymbol{m}_i(0)$ are initialized to random values. The only restriction here is that they are different for $i = 1, 2, \ldots, N$, where N is the number of neurons in the output layer. The minimum Euclidean distance criterion is then used to decide which neuron is activated. Let us denote the fired neuron as c, and

$$c = \arg\min_i \left\| x(n) - \boldsymbol{m}_i \right\|, \; i = 1, 2, \ldots, N. \tag{1}$$

The updating of the weights associated with the neurons is only performed within a neighborhood set $N_c(n)$ of the neuron c. All the neurons outside N_c are left intact. The neighborhood is around the neuron c and its size is reduced with increasing n. The updating rule can be formulated as:

$$\boldsymbol{m}_i(n+1) = \begin{cases} \boldsymbol{m}_i(n) + \alpha(n)\,[\boldsymbol{x}(n) - \boldsymbol{m}_i(n)] & \text{if} \quad i \in N_c(n) \\ \boldsymbol{m}_i(n) & \text{if} \quad i \notin N_c(n) \end{cases} \tag{2}$$

where $\alpha(n)$ represents a time-dependent learning rate with a value between 0 and 1. Both the size of the neighborhood $N_c(n)$ and the learning rate $\alpha(n)$ decrease with the training time n. In our experiments, the learning rate starts at 1 and linearly decrease to 0, and the neighborhood size starts by including all the neurons and gradually shrinks to contain only the activated neuron itself.

In summary, this first stage of learning discrete similarity measure partitions the original feature space into a number of distinct clusters, and the patterns belonging to each cluster are topologically similar (and hence visually similar as well). Since a search will be conducted within each cluster to order the patterns based on simple distance measures (such as the Euclidean distance), a second stage of supervised learn-

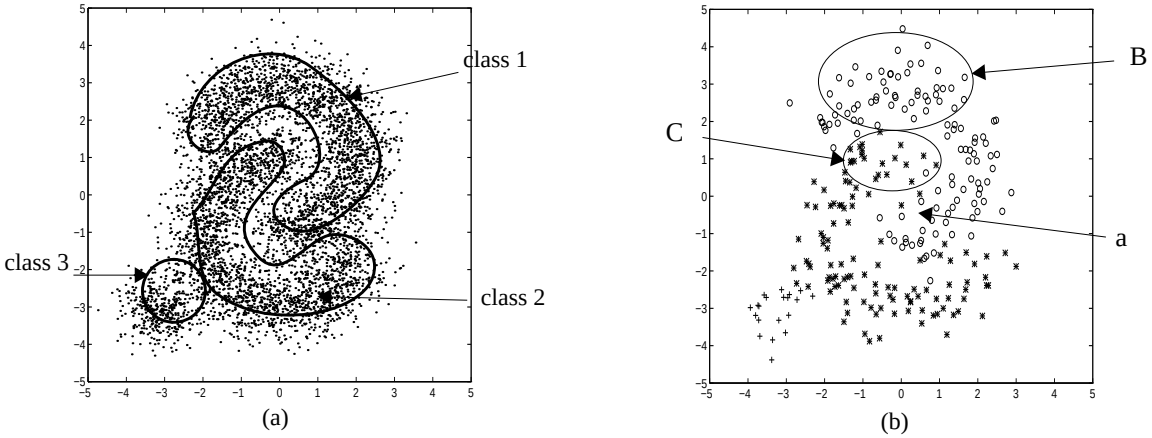


FIGURE 1. (a) shows an example of 2-D image features. They are represented by three different class labels. (b) shows a number of samples associated with each class. 'o' is from the class 1, '⋆' is from the class 2, and '+' from the class 3.

This learning algorithm incorporates both unsupervised and supervised learning using neural networks. A large number of labeled image data and the associated feature vectors are used to train the networks. The goal of the learning process is to partition the original feature space into many subregions, each representing a cluster of visually similar patterns. When a query image is presented, the network assigns a class label based on the feature vector. The final ordered set of retrieval results are then computed using the Euclidean distance measure within the same class.

2.2.1 Stage I: Unsupervised learning using Kohonen map

In the first stage of the learning process, a self-organizing feature-mapping algorithm (also called Kohonen map) is used. This unsupervised learning algorithm was first introduced by Kohonen [3] and has been used in many different applications. The Kohonen map transforms the input features of arbitrary dimension into a two-dimensional discrete map, where the topological relationship between different features are preserved as much as possible. The Kohonen map is a two-layered network which is shown in Figure 2. The first layer of neurons receive the input feature information whereas the second layer of neurons are organized on a 2-D map for presenting the decision results. The two layers are fully connected and the weights associated with these connections are adjusted during the training stage.

The motivation for using the Kohonen map for the first stage of learning similarity measure is for the following reasons. First, this network can adaptively separate clusters in the feature space. It can be shown that the mean of the cluster i is essentially the weight vector $\boldsymbol{m}_i$ of the corresponding neuron. Secondly, the output neurons are topologically ordered in the sense that neighboring neurons in the lattice correspond to *similar* clusters in the original high-dimensional feature space. Thus, in addition to dimensionality reduction, it also preserves the topology of the feature vectors.

than few thousand objects). However, in applications such as face recognition or satellite images where there can be several million entries, there is a need for developing new algorithms for fast browsing.

Many researchers have proposed the use of Karhunen-Loeve (KL) transform as a tool to reduce the feature space. Although the KL transform has some nice optimality properties, one should be careful in its use. Dimensionality reduction does not necessarily help for fast or accurate search, as we illustrate in the next section.

This paper is organized as follows: In Section 2, we propose a neural network algorithm to learn similarity measures. In Section 3, we suggest a texture dictionary based on the results of the neural network learning, which is used to encode all the database objects. Finally, some experimental results are shown in Section 4.

2 LEARNING SIMILARITY MEASURES

The goal of feature extraction is to transform the raw image data into a much lower dimensional space, thus providing a compact representation of the image. It is desirable that the feature vectors computed preserve the perceptual similarity. That is, if two image patterns are visually similar, then the corresponding feature vectors are also close to each other. We will not consider the feature extraction issue in this paper. We refer to [1] for the work related to texture image features where we have demonstrated that the Gabor features provide the best performance. The focus here will be on the analysis following Gabor feature extraction, and to design a strategy to make the best use of the available feature information.

2.1 Feature properties and similarity measures

How can we capture the notion of similarity in the feature space? Figure 1(a) shows an example of a 2-D feature set. Three different pattern classes are well represented by the three separate clusters. The three classes can be easily separated by simple (although non-linear) decision boundaries. However, using simple Euclidean distance measures (or modifications of it) and searching for nearest neighbors might retrieve patterns with no perceptual relevance to the original query pattern.

This problem is illustrated in Figure 1 (b), where a number of samples from three different classes are shown. Let the point “a” (in class 1) correspond to a feature vector obtained from the query image. Then, a simple nearest neighbor search will not be able to retrieve images in the group B (with the same class as the point “a”) without retrieving the images in the group C (belonging to class 2). This is because the images in C have larger similarity values than the images in B using the Euclidean distance measure. This is a classical problem in pattern recognition, but not much effort has been made to address these issues in the context of image database browsing.

2.2 Learning discrete similarity measure using neural networks

In order to simplify the problem, let us consider the case of a finite set of image classes and any given image pattern is assigned to one of these labels. All patterns belonging to the class are considered similar and those belonging to different classes are considered not similar. Thus we have a discrete similarity measure to compare two patterns. In the following we describe a neural network algorithm to learn a similarity measure in the feature space.

Image Indexing Using a Texture Dictionary

W. Y. Ma and B. S. Manjunath

Department of Electrical and Computer Engineering
University of California at Santa Barbara, Santa Barbara, CA 93106

ABSTRACT

We propose a new method for indexing large image databases. The method incorporates neural network learning algorithms and pattern recognition techniques to construct an image pattern dictionary. Image retrieval is then formulated as a process of dictionary search to compute the best matching codeword, which in turn indexes into the database items. Experimental results are presented.

Keywords: content-based image retrieval, texture, indexing, neural networks, pattern recognition, vector quantization.

1 INTRODUCTION

Searching for similar image patterns in a large database can be conceptually visualized as a two step process. In the first step, a set of image processing operations are performed on the query pattern to compute a feature representation. The next step is to search through the database to retrieve patterns which are similar in the feature space. The features could be color, shape, texture, or any other image attributes of interest. Assuming that one can design appropriate feature extraction operators, searching for similar patterns in the feature space poses the following problems:

- Defining distance measures in the feature space to compare two patterns.
- Designing data structures to facilitate fast search and retrieval.

A similarity measure in the feature space should capture the similarity between the original image patterns. However, the latter itself is, in many cases, a subjective measure. This is particularly true when the image features correspond to low level image attributes such as texture, color, or shape. Simple distance measures, such as the normalized Euclidean distance, on these features may or may not preserve the perceptual similarity. What is not obvious is that the choice of a distance measure has a significant impact on the performance of the overall system. We demonstrated this in [1] where we made an extensive experimental evaluation of different texture features such as the Markov random field model parameters, wavelet coefficients, and Gabor features. Our experiments indicated a wide variance in performance while using Euclidean or weighted Euclidean distance on the different feature sets. For example, using the Mahalanobis distance for the Markov model feature gave the best performance on that feature set. On the other hand, the unweighted distance performed better for the Gabor features.

The second problem arises because of the (typically) large dimensional feature space. The standard database indexing methods perform poorly for dimensions exceeding ten [2]. Typical image feature dimensions are of the order of 20-100. The search time may not be a major issue for small databases (less