A Texture Thesaurus for Browsing Large Aerial Photographs

Wei-Ying Ma and B. S. Manjunath
Department of Electrical and Computer Engineering, University of California, Santa Barbara, CA 93106-9560.
E-mail: wei@iplab.ece.ucsb.edu; manj@ece.ucsb.edu

A texture-based image retrieval system for browsing large-scale aerial photographs is presented. The salient components of this system include texture feature extraction, image segmentation and grouping, learning similarity measure, and a texture thesaurus model for fast search and indexing. The texture features are computed by filtering the image with a bank of Gabor filters. This is followed by a texture gradient computation to segment each large airphoto into homogeneous regions. A hybrid neural network algorithm is used to learn the visual similarity by clustering patterns in the feature space. With learning similarity, the retrieval performance improves significantly. Finally, a texture image thesaurus is created by combining the learning similarity algorithm with a hierarchical vector quantization scheme. This thesaurus facilitates the indexing process while maintaining a good retrieval performance. Experimental results demonstrate the robustness of the overall system in searching over a large collection of airphotos and in selecting a diverse collection of geographic features such as housing developments, parking lots, highways, and airports.

1. Introduction

The use of texture as a visual primitive to search and retrieve aerial photographs is investigated. With the growing amount of imagery in the geographic databases, there is a need to develop tools for efficient extraction of information, and for intelligent searches and manipulation of the image data, so that the potential use of the accumulated images can be fully realized. The required techniques include image feature extraction for characterizing the underlying image attributes, such as texture, shape/contour, and color/multispectral information, and efficient search and indexing in the multidimensional feature space. The QBIC (Niblack et al., 1993, 1995) and the Photobook (Pentland, Picard, & Sclaroff, 1994) are examples of image content-based retrieval systems which make use of several of these image attributes.

In this article, we will demonstrate that texture could be used to select a large number of geographically salient features including vegetation patterns, parking lots, and building developments. Using texture primitives as visual features, one can query the database to retrieve similar image patterns. Much of the results presented are with airphotos, although a similar analysis can be applied to LANDSAT and SPOT satellite images. A schematic diagram of the prototype system is shown in Figure 1. This is currently being integrated into the Alexandria Digital Library (ADL) project (Smith, 1996), whose goal is to establish an electronic library of spatially indexed data, providing Internet access to a wide collection of geographic information. A significant part of this collection includes maps, satellite images, and airphotos. For example, the Maps and Imagery Library at the UCSB contains over 2 million historically valuable aerial photographs. A typical airphoto can take over 25 MB of disk space, and providing access to such data raises several important issues, such as multiresolution browsing (Strobel, Mitra, & Manjunath, 1995), and selecting images based on content.

What distinguishes image search for database-related applications from traditional pattern-classification methods is the fact that there is a human in the loop (the user), and there is a need to retrieve more than just the best match. In typical applications, a number of top matches with rank-ordered similarities to the query pattern will be retrieved. Comparison in the feature space should preserve visual similarities between patterns. This is an important but difficult problem in content-based image retrieval (Picard, 1995). Towards this objective, a neural network algorithm to learn pattern similarity in the feature space is proposed (Section 3). This approach uses training data containing pattern class information to partition the feature space into many visually similar clusters. A performance evaluation of this approach using the Brodatz texture database is provided (Section 3.2). Experi-

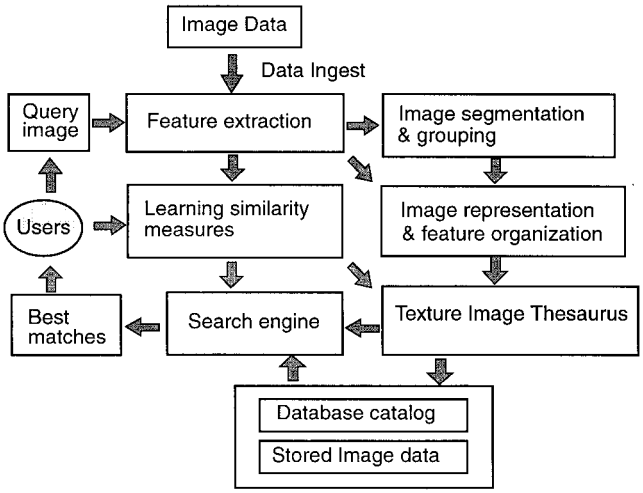


FIG. 1. Content-based image retrieval system for browsing large-scale aerial photographs.

mental results indicate that a significantly better retrieval performance can be achieved. An additional advantage of this approach is that it also provides an efficient indexing tree to narrow down the search space. By combining a learning similarity algorithm with a hierarchical vector quantization scheme, a texture image thesaurus is created to facilitate fast retrieval of patterns during query time (Section 4).

The main contributions of this work are in identifying the various components required for building a working system and in implementing a texture thesaurus for content-based search of a large collection of images. Novel features of this system include a fast image segmentation scheme based on texture edge flow and the use of a hybrid neural network algorithm for developing the image texture thesaurus. The texture feature extraction work is described in detail in Manjunath and Ma (1996), and some of the early work are presented in Ma and Manjunath (1995, 1996).

The organization of this article is as follows: The next section briefly explains the feature extraction and segmentation components. Section 3 discusses the problem of similarity measures and the use of a neural network learning algorithm for improving the retrieval performance. Section 4 introduces the indexing mechanism based on a texture image thesaurus. Some experimental results on browsing airphotos are shown, and Section 5 concludes with discussions.

2. Image Analysis

2.1. Texture Features for Content-Based Search

In our recent research (Manjunath & Ma, 1996), we have proposed a Gabor texture feature extraction scheme and provided a comprehensive evaluation and comparison with other multiresolution texture features using the Brodatz texture (Brodatz, 1966) database. The experimental results indicate that the Gabor features provide good pattern retrieval accuracy. In this article, these features are further used to segment the images and to create a texture thesaurus for browsing and indexing images in the database. A brief review of the texture feature extraction is given below.

First consider a prototype Gabor filter:

$$G(x, y) = \left(\frac{1}{2\pi\sigma_x\sigma_y}\right) \times \exp\left[-\frac{1}{2}\left(\frac{x^2}{\sigma_x^2} + \frac{y^2}{\sigma_y^2}\right)\right] \cdot \exp[2\pi jWx] \quad (1)$$

which is a Gaussian function modulated by a complex sinusoid. A bank of Gabor filters can be generated by dilating and rotating the above function:

$$G_{mn}(x, y) = a^{-m}G(x', y'),\ a > 1,\ m, n = \text{integer}$$
$$x' = a^{-m}(x\cos\theta + y\sin\theta),$$
$$y' = a^{-m}(-x\sin\theta + y\cos\theta), \quad (2)$$

where $\theta = n\pi/K$ and K is the total number of orientations. The scale factor a^{-m} normalizes the filter responses. These Gabor filters can be considered as orientation and scale tunable edge and line (bar) detectors. The statistics of the filtered outputs can be used to characterize the underlying texture information. Given an image $I(x, y)$, let $w_{mn}(x, y)$ be the filtered output for $G_{mn}(x, y)$. The mean and standard deviation of the amplitude $|w_{mn}(x, y)|$ are then used to form a feature vector $\bar{f}$ to represent the input image pattern. Five different scales and six orientations are used in the experiments described in the following sections. This results in a feature vector of length 60 ($5 \times 6 \times 2$). For more details, please see the article by Manjunath and Ma (1996).

2.2. Image Segmentation and Grouping

The airphoto database used in the experiments contains images which are 5K $\times$ 5K pixels. For simplicity, the initial implementation partitioned the images into 64 $\times$ 64 blocks of pixels, thus resulting in about 6,400 blocks per image. For each block, a texture feature vector is computed as explained in the previous section. Figure 2a shows a small portion of an airphoto and its associated tiles. A more compact image representation can be done by grouping the tiles together based on texture similarity.

An image segmentation scheme which is appropriate for such large images and database retrieval applications is now outlined. The proposed scheme utilizes the Gabor

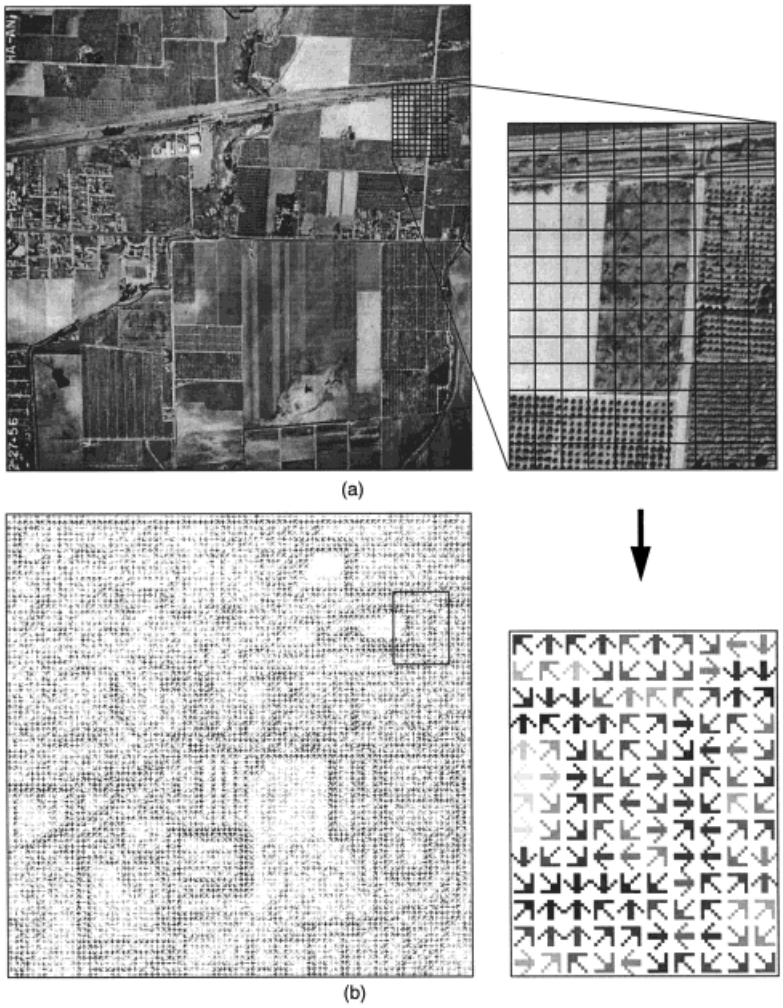


FIG. 2. Image segmentation based on texture flow. **a:** An aerial photograph. **b:** The local texture flow after orientation competition. **c:** The results after the texture flow propagation reaches a stable state. **d:** Boundaries detected by turning on the edge signals between texture flows in opposite directions. **e:** The initial set of image regions by connecting the boundaries. **f:** The final segmentation results after a conservative region merging.

texture features extracted from the image tiles and performs a coarse image segmentation based on local texture gradient. Note that accurate pixel level segmentation is often not necessary for such image retrieval applications, although the segmentation method itself can achieve pixel level accuracy (Ma & Manjunath, 1997). Figure 2 shows the different stages of the segmentation algorithm, which are summarized below:

Local texture gradient computation. Using the feature vectors, a local texture gradient is computed between each image tile and its surrounding eight neighbors. The dominant flow direction is identified in a competitive manner which is similar to a winner-takes-all representation. We call this a *texture edge flow* as the gradient information is propagated to neighboring pixels (or tiles), and this texture edge flow contains information about the (spatial) direction and energy of the local texture boundary. Figure 2b shows the results of flow computation where the local texture gradient is represented by the arrows whose directions point to the texture boundaries, with darker intensities representing stronger texture gradients.

Texture edge flow propagation. Following the orientation competition, the local texture edge flow is propagated to its neighbors if they have the same directional preference. The flow continues till it encounters an opposite flow. This helps to localize the precise positions of the boundaries and concentrate the edge energies towards pixels where the image boundaries might exist. Figure 2c shows the results of this stage.

Boundary detection. After the propagation reaches a stable state, the final texture edge flow energy is used for boundary detection. This is done by turning on the edge signals between two neighboring image tiles if their final texture edge flow points in opposite directions. The

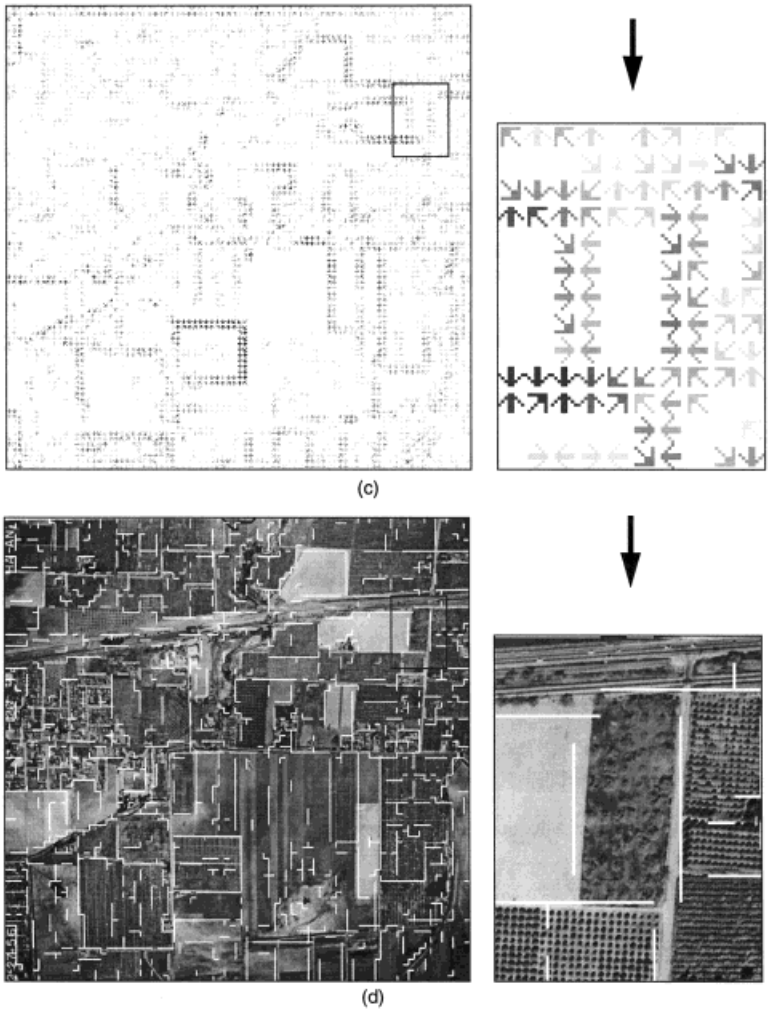


FIG. 2. (Continued)

texture edge energy is then defined to be the summation of texture edge flow energies in the two neighboring image tiles. Figure 2d shows the results of this stage.

Region merging. The previous stage results in many discontinuous image boundaries. They are connected to form an initial set of image regions (Fig. 2e). At the end, a conservative region merging algorithm is used to group similar neighboring regions. The final image segmentation result is shown in Figure 2f.

The advantage of this approach is that it directly uses the extracted texture features to compute the image segmentation. In the current implementation, the number of regions obtained from each large aerial photograph is about 100–200 (compared to 6,400 original 64 × 64 tiles).

2.2.1. Region texture features. After the image is segmented, the mean and standard deviation of Gabor filtered outputs of each image region are used to construct the corresponding region texture features. Let $\mu_{mn}^{(k)}$ and $\sigma_{mn}^{(k)}$ be the mean and standard deviation of the amplitude of Gabor filtered outputs $w_{mn}(x, y)$ of the image tile k (note that m and n refer to the filter $G_{mn}(x, y)$ as discussed in Section 2.1). Now we use $W_{mn}(x, y)$ to represent the filter output of the region which contains many image tiles. Based on the simple expectation rule $E(W) = E(E(w|k))$, the mean of filter output of the region can be obtained as

$$\boldsymbol{\mu}_{mn} = \frac{1}{N} \sum_{k=1}^{N} \mu_{mn}^{(k)}, \tag{3}$$

where N is the total number of tiles belonging to a given region. Similarly, the standard deviation of filter output can be obtained using

$$\boldsymbol{\sigma}^2 = E(W^2) - E(W)^2 \text{ and } E(W^2) = E(E(w^2|k)) = E((\mu^{(k)})^2 + (\sigma^{(k)})^2).$$

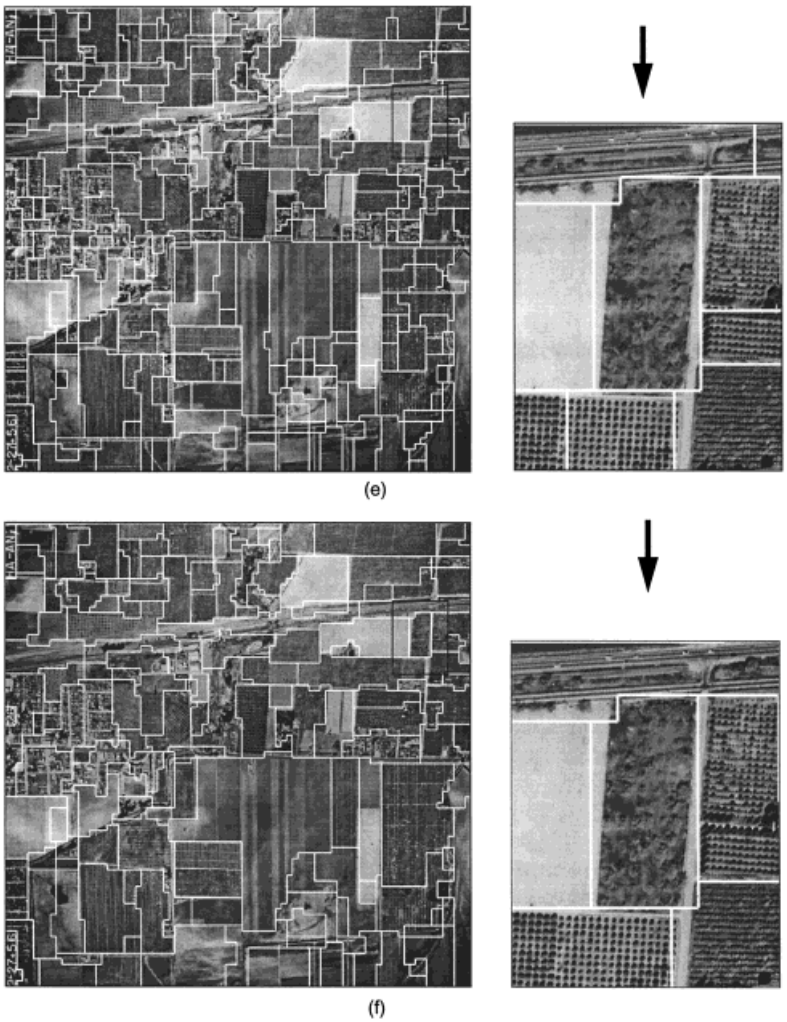


FIG. 2. (Continued)

Thus,

$$\boldsymbol{\sigma}_{mn} = \sqrt{\left(\frac{1}{N}\sum_{k=1}^{N}(\mu_{mn}^{(k)})^2 + (\sigma_{mn}^{(k)})^2\right) - \boldsymbol{\mu}_{mn}^2}. \quad (4)$$

3. Similarity Measures and Learning

The search for similar image patterns can be considered a two-step process. The first step extracts feature vectors to annotate the underlying image information, and the second step is to search through the database to retrieve similar patterns. A similarity measure in the feature space needs to be defined to 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. A nearest neighbor search in the entire feature space may not be appropriate. This problem was also observed and discussed in Minka and Picard (1995). They noticed that different models have different performances on different tasks, and therefore, proposed a ''society of models'' approach to combine a variety of models and try to utilize them in the best way. The learning algorithm, based on the user's input, dynamically determines which model or combination of models is best for subsequent classification.

In contrast, our approach focuses on the improvement of using a single feature model. The strategy is to first cluster the patterns in the feature space and then assign a similarity class label to each subspace. When a query pattern is presented, the system first classifies it into one of the subspaces, and conducts the final nearest neighbor search only within that subspace.

3.1. Learning Discrete Similarity Measure Using Neural Networks

A hybrid neural network algorithm is used to cluster texture patterns in the feature space. This algorithm con-

tains two stages of training. The first stage performs an unsupervised learning using the Kohonen feature map to capture the underlying feature distribution. In the second stage, clusters are labelled using a winner-takes-all representation, and class boundaries are fine-tuned using a learning vector quantization scheme. This results in a partitioning of the original feature space into clusters of visually similar patterns, based on the class label information provided by human observers.

3.1.1. Stage I: A Kohonen feature map for clustering texture features. The self-organizing feature-mapping algorithm (Kohonen, 1982, 1990) is used in the first stage of learning. In this, a two-layer network transforms the input features of arbitrary dimension into a two-dimensional discrete map, where the topological relationship between different features are preserved. The first layer of neurons receive the input feature information, whereas the second layer of neurons is 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 feature 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 $\mathbf{m}_i$ 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.

The weights in the network are adjusted based on competitive learning. The output neurons of the network compete among themselves in a winner-takes-all representation (Haykin, 1994). 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 $\mathbf{x}$ to the input layer of the network and adjusting the connection weight vectors $\mathbf{m}_i$ according to the updating rules given below. At the beginning, all the weight vectors $\mathbf{m}_i(0)$ are initialized to random values. The only restriction here is that they be different for $i = 1, 2, \ldots, N$, where N is the number of neurons in the output layer. In the absence of any additional information about cluster formation, the intuition is to use the minimum Euclidean distance criterion to decide which neuron is activated. Let us denote the fired neuron as c, then

$$c = \arg\min_i \|x(n) - \mathbf{m}_i\|, \; i = 1, 2, \ldots, N. \quad (5)$$

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:

$$\mathbf{m}_i(n+1) = \begin{cases} \mathbf{m}_i(n) + \alpha(n)[\mathbf{x}(n) - \mathbf{m}_i(n)] & \text{if } i \in N_c(n) \\ \mathbf{m}_i(n) & \text{if } i \notin N_c(n) \end{cases} \quad (6)$$

where $\alpha(n)$ represents a time-dependent learning rate with a value between 0 and 1. Note that the weights of the selected neurons are adjusted to move in the direction of the input vector. Both the size of the neighborhood $N_c(n)$ and the learning rate $\alpha(n)$ decrease with the training time n. In the experiments in Sections 3.2 and 4.1, the learning rate starts at 1 and linearly decreases to 0, and the neighborhood size starts by including all the neurons and gradually shrinks to contain only the activated neuron itself.

In summary, the first stage of learning partitions the original feature space into a number of distinct clusters, and the patterns belonging to each cluster are topologically similar. 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 learning is used to fine-tune the network parameters. After the second stage of learning, each output neuron will have an associated class label.

3.1.2. Stage II: Cluster labeling and learning vector quantization. Following clustering, the first phase of supervised learning labels the output neurons of the Kohonen map. This is done by presenting a number of training feature vectors with known classification to the network. The output neurons are assigned to different classes by majority voting. However, the feature map is mainly intended to approximate input feature vectors, or their probability density functions. A fine-tuning of the network is necessary to improve the classification accuracy.

Fine-tuning is performed in the second phase using a learning vector quantization (LVQ) algorithm (Haykin, 1994; Kohonen, 1990). This algorithm has been widely used to further improve the clustering obtained from classical Vector Quantization (VQ) methods. The Kohonen map of the first stage is basically a vector quantizer. The output neuron of the map, after labeling, represents different pattern class centroids. Fine-tuning often helps in readjusting the class boundaries after the clusters have been labeled for better retrieval. In this article, the type 3 algorithm (LVQ3) (Haykin, 1994; Kohonen, 1990) is used

to update the weights. The details of this algorithm are as follows:

Let $\mathbf{m}_i$ and $\mathbf{m}_j$ be the two closest weight vectors to a given input feature vector $\mathbf{x}$. Let $C(\mathbf{x})$ be the known class label associated with $\mathbf{x}$. Let the class label associated with the ith neuron be represented by C_i. Let $d_i = \|\mathbf{x} - \mathbf{m}_i\|$ 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 $\mathbf{x}$ is considered to be within the window, and the following updates are computed if the relative distances satisfy the following condition:

$$\min(d_i/d_j, d_j/d_i) > \gamma. \quad (7)$$

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

$$\mathbf{m}_i(n+1) = \mathbf{m}_i(n) - \alpha(n)[\mathbf{x}(n) - \mathbf{m}_i(n)]$$
$$\mathbf{m}_j(n+1) = \mathbf{m}_j(n) + \alpha(n)[\mathbf{x}(n) - \mathbf{m}_j(n)]. \quad (8)$$

- Case 2: $C_j = C(\mathbf{x}) = C_i$, then

$$\mathbf{m}_k(n+1) = \mathbf{m}_k(n) + \varepsilon\alpha(n)[\mathbf{x}(n) - \mathbf{m}_k(n)],\ k \in \{i, j\}. \quad (9)$$

- Case 3: $C_i \neq C_j \neq C(\mathbf{x})$, then no action.

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 the experiments below, $w = 0.2$ and $\varepsilon = 0.3$.

Once the network is trained, the search and retrieval process is performed in the following way:

- When a query pattern is presented, the network first identifies a subspace of the original feature space which is more likely to contain visually similar patterns.
- The final retrievals are then computed using a simple Euclidean distance measure with the patterns which belong to the corresponding sub-space.

In addition to retrieving visually similar patterns, an additional advantage of this clustering approach is that it provides an efficient indexing tree to narrow down the search space. The cluster centers can be used to construct a visual texture thesaurus for facilitating the search process (Section 4).

3.2. Performance Evaluation

The Brodatz texture database is used to evaluate the performance of the learning algorithms for texture similarity. This database contains 116, 512 × 512 texture images. In order to create a number of small images which belong to the same class, we partition each of the 512 × 512 images into 49 128 × 128 subimages (with overlap, and centered on a 7 × 7 grid over the 512 × 512 image). The first 33 subimages are used as the training data, and the last 16 are for testing. The total number of features for training and testing are thus 3,828 (33 × 116) and 1,856 (16 × 116), respectively. Figure 3 shows a subimage from each of the 116 large texture images. The Gabor features from each of these subimages are extracted and normalized.

The 116 texture classes are further grouped into 32 similarity classes, each of them containing 2–6 texture classes. All the texture subimages belonging to the same similarity class are visually similar. This classification was done manually by researchers in the laboratory, and Table 1 shows these various similarity classes and the corresponding textures. While it is possible that different groups of individuals might come up with slightly different categorizations, our experiments indicate that the actual classification will have little effect on the final performance as long as similar images are within the same class.

For the two layered feature mapping network, we selected 400 output neurons. As a general rule, the number of output neurons is approximately 10–15 times the number of similarity classes (clusters) to be learned. After the training stage, the majority rule is used to assign a class label to each of the neurons.

During the retrieval time, the query image is first transformed into the feature space. The network then classifies it into one of the 32 similarity classes, and within a class a linear search for the closest set of feature vectors is performed using the Euclidean distance measure.

Figure 4 illustrates an evaluation of the retrieval performance with and without learning similarity measures. The evaluation is based on the 32 similarity classes using only the test data set (1,856 subimages). By considering different numbers of the top retrieval (horizontal axis), the average percentage of the images from the same similarity class is used to measure the performance (vertical axis). As can be seen from Figure 4, the performance without learning deteriorates rapidly (in terms of visual similarity) after the first 10–15 top matches, the retrievals based on learning similarity continue to perform very well. Figure 5 shows a retrieval example which illustrates the improvement obtained with the learning algorithm.

4. A Texture Thesaurus for Indexing

The previous sections demonstrated that clustering in feature space improves retrieval results significantly. In addition, it also provides a natural hierarchical data structure for fast indexing and retrieval. Typically, the image features are in a high dimensional space, ranging from tens to a few hundred components. It is well known that traditional indexing structures, such as B-tree or R-tree, do not generalize well to such large dimensions (Alexandrov, Ma, Abbadi, & Manjunath, 1995).

Many researchers have proposed the use of the Karhu-

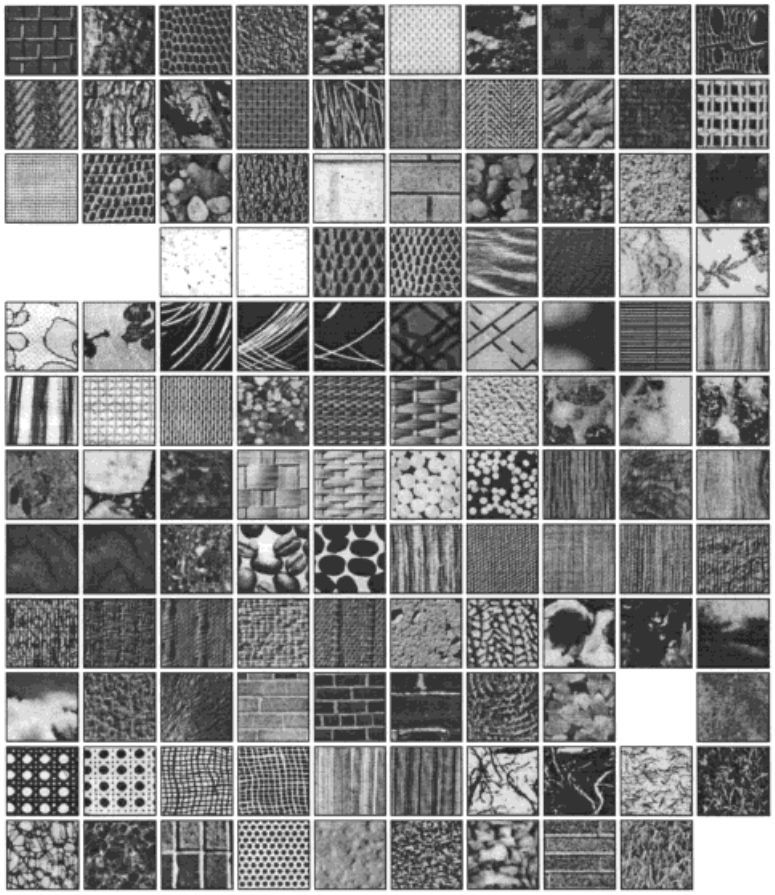


FIG. 3. Example images of 116 different textures. From right to left, and top to bottom, are the D1–D112 textures from the Brodatz album. The last seven textures are O1–O7 from the USC database. Missing D* textures are left blank (white space).

nen-Loeve (KL) transform to reduce the dimensions of the feature space. Although the KL transform has some nice properties, such as identifying the most important subspace, one should be careful in its use as the feature properties that are important for identifying the pattern similarity might be destroyed during blind dimensionality reduction (Krzanowski, 1995).

As an alternative, a texture thesaurus model is pro-

TABLE 1. Texture clusters used in learning similarity measures.*

Cluster	Texture Class	Cluster	Texture Class	Cluster	Texture Class
1	D1, D6, D14, D20, D49	12	D62, D88, D89	23	D19, D82, D83, D85
2	D8, D56, D64, D65	13	D24, D80, D81, D105, D106	24	D66, D67, D74, D75
3	D34, D52, D103, D104	14	D50, D51, D68, D70, D76	25	D101, D102, O1
4	D18, D46, D47	15	D25, D26, O1, D96	26	D2, D73, D111, D112
5	D11, D16, D17	16	D94, D95, O6	27	D86, O3
6	D21, D55, D84	17	D69, D71, D72, D93	28	D37, D38
7	D53, D77, D78, D79	18	D4, D29, D57, D92, O4	29	D9, D109, D110, D116
8	D5, D33, O5	19	D39, D40, D41, D42	30	D107, D108
9	D23, D27, D28, D30, D54, D98	20	D3, D10, D22, D35, D36, D87	31	D12, D13
10	D7, D58, D60	21	D48, D90, D91, D100	32	D15, D97
11	D59, D61, D63	22	D43, D44, D45		

* These have been identified by human subjects as being visually similar within each cluster. See Figure 3 for the texture images.

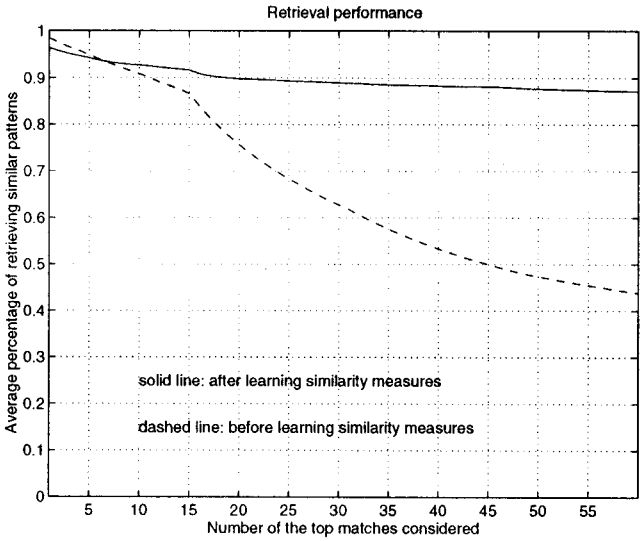


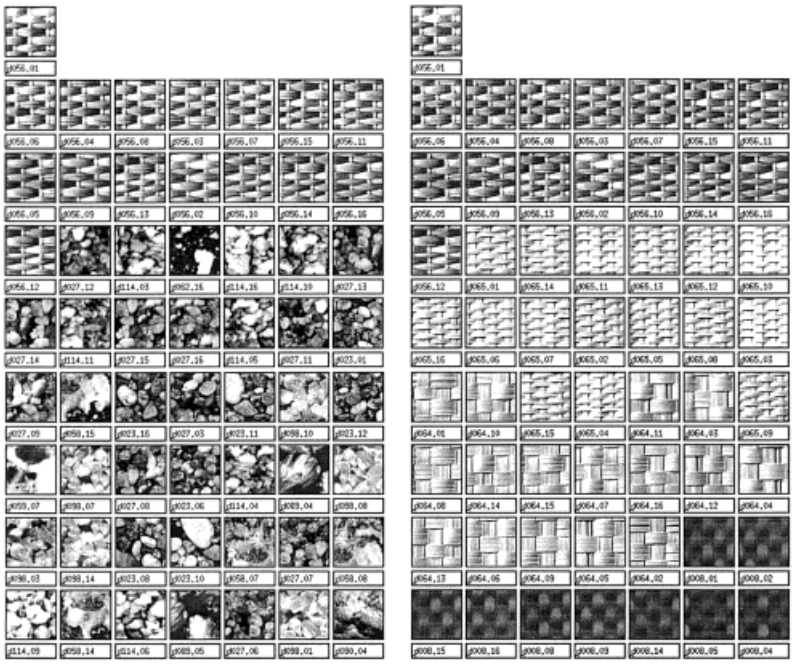
FIG. 4. Retrieval performance before and after learning similarity measures. Notice the significant improvement in similarity-based retrieval with learning.

posed to browse through the image database. This model can be visualized as an image counterpart of the traditional thesaurus for text search. It creates the information links among the stored image data based on a collection of codewords and sample patterns obtained from the training set. Similar to parsing text documents using a dictionary or thesaurus, the information within images can be classified and indexed via the use of a texture thesaurus. The design of the texture thesaurus has two stages. The first stage uses the learning similarity algorithm to combine the human perceptual similarity with the low level feature vector information, and the second stage utilizes a hierarchical vector quantization technique to construct the codewords. The salient features of this model include:

- The texture thesaurus is domain-dependent and can be designed to meet the particular need of a specific image data type by exploring the training data.
- It provides an efficient indexing tree while maintaining or even improving the retrieval performance in terms of human perception.
- The visual codeword representation in the thesaurus can be used as information samples to help users browse through the database.

4.1. Texture Thesaurus Construction

A texture thesaurus is constructed using aerial photographs of Santa Barbara and nearby regions taken at dif-



(a) Without learning similarity measures (b) With learning similarity measures

FIG. 5. Pattern retrieval with and without learning. Each query pattern has 15 other similar patterns in the database. The input query (d056.01) is shown at the top of the column in each case. With or without learning, the Gabor features provide a very good representation in retrieving all the other 15 images from the same texture class. However, note the degradation in visual similarity after that for the case without learning. The images are ordered according to decreasing similarity from left to right, and top to bottom. For the case with learning similarity, the performance continues without any marked degradation in perceptual similarity, even after 50 patterns are retrieved.

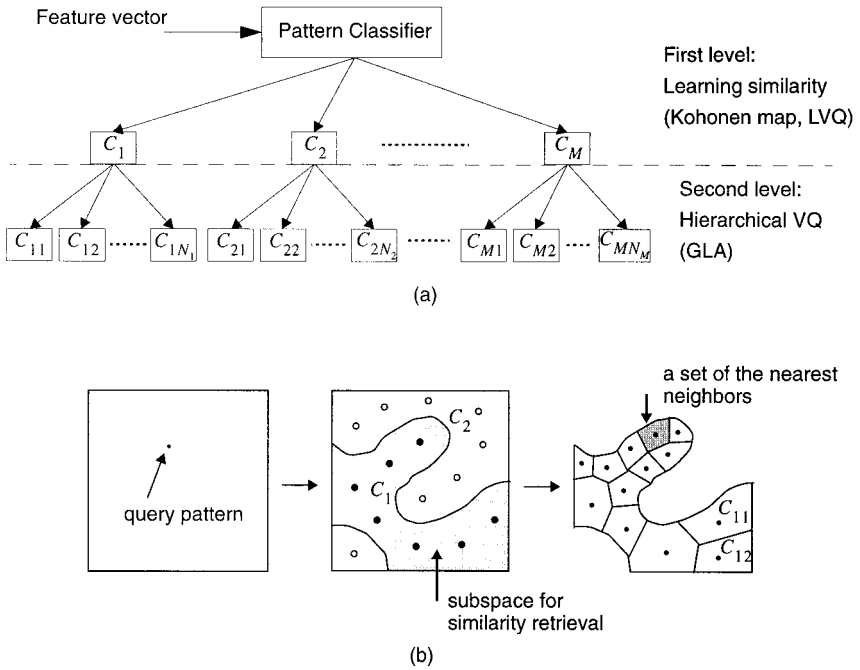


FIG. 6. **a:** The construction of the two-level indexing tree using a learning similarity algorithm and a hierarchical vector quantization technique. The first level partitions the original feature space into many visually similar subspaces. Within each subspace, the second level of the tree further divides it into a set of smaller clusters. **b:** An example of indexing a 2-D image feature.

ferent times. A subset of the image data is manually labeled, and their corresponding texture features are used as training data for the hybrid neural network algorithm, described in Section 3, to create the first level of the indexing tree. The purpose of this stage is to partition the original feature space into many visually similar subspaces. Within each subspace, a hierarchical vector quantization technique is used to further partition the space into many smaller clusters. The centroids of these clusters are used to form the codewords in the texture thesaurus, and the training image patterns of these centroids are used as icons to visualize the corresponding codewords.

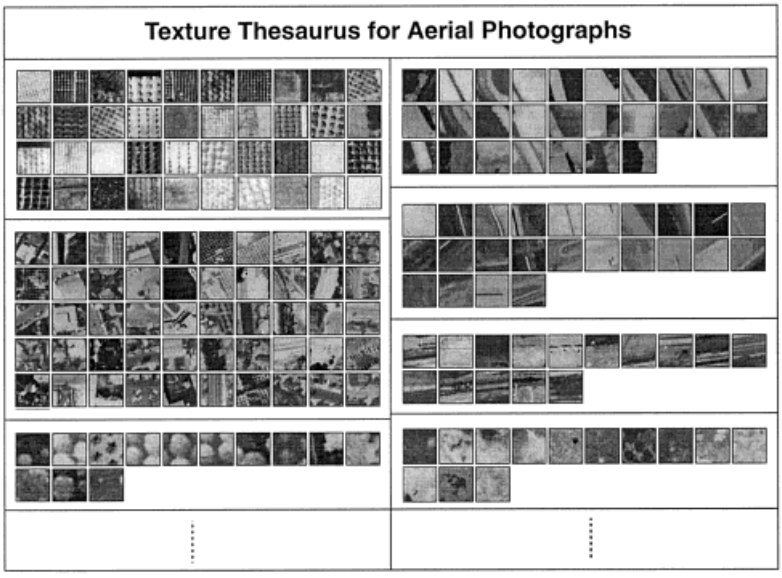


FIG. 7. Examples of the codewords obtained for the aerial photographs. The patterns inside each block belong to the same class.

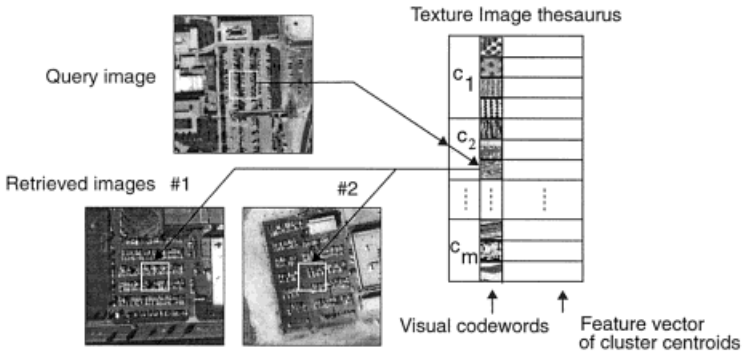


FIG. 8. A texture image thesaurus for content-based image indexing. The image tiles shown here contain parking lots.

Let M be the number of pattern classes in the training data. Thus, the first level of the indexing tree will partition the feature space into M subspaces. Each of these subspaces is further partitioned using the generalized Lloyd algorithm (GLA) (Gersho & Gray, 1992), which is summarized in the following:

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

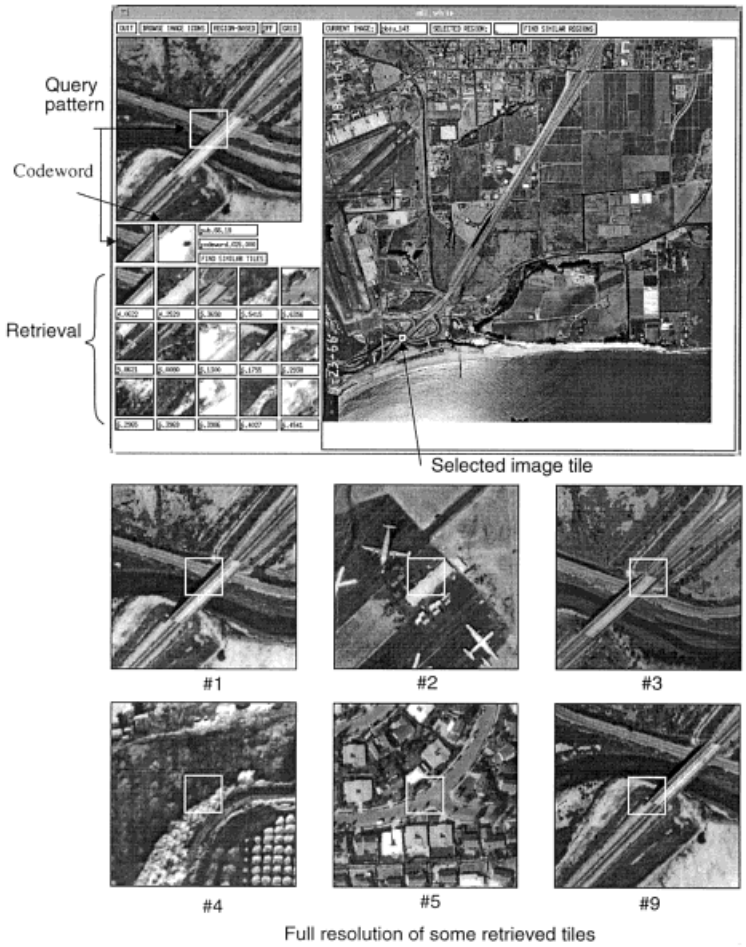


FIG. 9. Tool for browsing large aerial photographs using Gabor features and the texture thesaurus. Users can select any image tile or region of interest to see the original resolution of that area (shown in the upper-left corner) and request the system to retrieve similar images from the database. The example shown here is the tile-based search on a ''bridge'' pattern. The best 15 matches are shown in a row-scan order. The full resolution subimages in the surrounding neighborhood of some of the tiles are also illustrated.

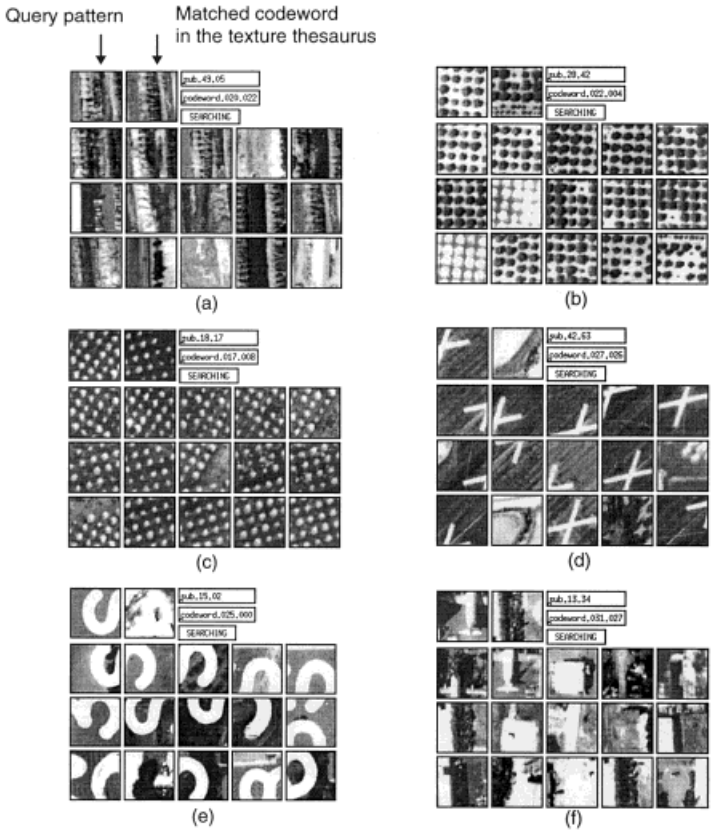


FIG. 10. Examples of the tile-based search. **a–c:** From the vegetation areas. **d:** The crossmark from the runway of an airport. **e:** A portion of the marked letter ''S'' in the image. **f:** An airplane. As can be seen, the top two matches also contain airplanes.

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 criterion to form the nearest neighbor cells:

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

3. Use the following centroid condition to update the codebook:

$$C_{m+1} = \{\text{centroid}(R_i);\ i = 1, 2, \ldots, N\},$$

which is the optimal reproduction codebook for the cells just found.

4. Compute the average distortion for C_{m+1}, which is defined as $D = \Sigma_{i=1}^{N} \Sigma_{x \in R_i} \|x - y_i\|$. If the change in distortion between the codebooks C_{m+1} and C_m is less than a certain threshold, then stop. Otherwise set $m + 1 \rightarrow m$ and go to step 2. In our experiments, we used the criterion $|D(C_{m+1}) - D(C_m)|/(D(C_m)) < 0.001$ to stop the iterations.

The initial codebook C_1 is generated based on the technique proposed by Katsavounidis, Kuo, & Zhang (1994). This starts by selecting the training vector with the largest norm as the first codeword in the codebook. Distance between this initial codeword and all the remaining training feature vectors are computed, and the vector with the largest distance is selected as the next codeword. This process is repeated in selecting additional codewords: Let $d_{ic} = \min\{d_{ik},\ k \in codebook\}$ be the distance between ith training feature vector and the codebook vectors. The next entry to the codebook will be the training vector with the maximum distance to the codebook. This procedure stops when the desired size of the codebook is reached.

Notice that the above procedure is used to construct codewords for each of the subspaces. This algorithm results in a set of cluster centroids. This centroid set is then used to construct the second level of indexing tree and the codewords. This scheme for the texture thesaurus design is in spirit similar to the tree-structured vector quantization (TSVQ) for image coding. However, instead of trying to minimize the image reconstruction error, the goal here is to minimize the recognition or classification error according to the class labels provided by human observers. In order to help users visualize the codewords, image patterns with feature vectors closest to the cluster centroids are used as the iconic representations for the corresponding codewords.

The number of codewords obtained from each subspace is dependent on the number of training data classified into the corresponding subspace and their distribution

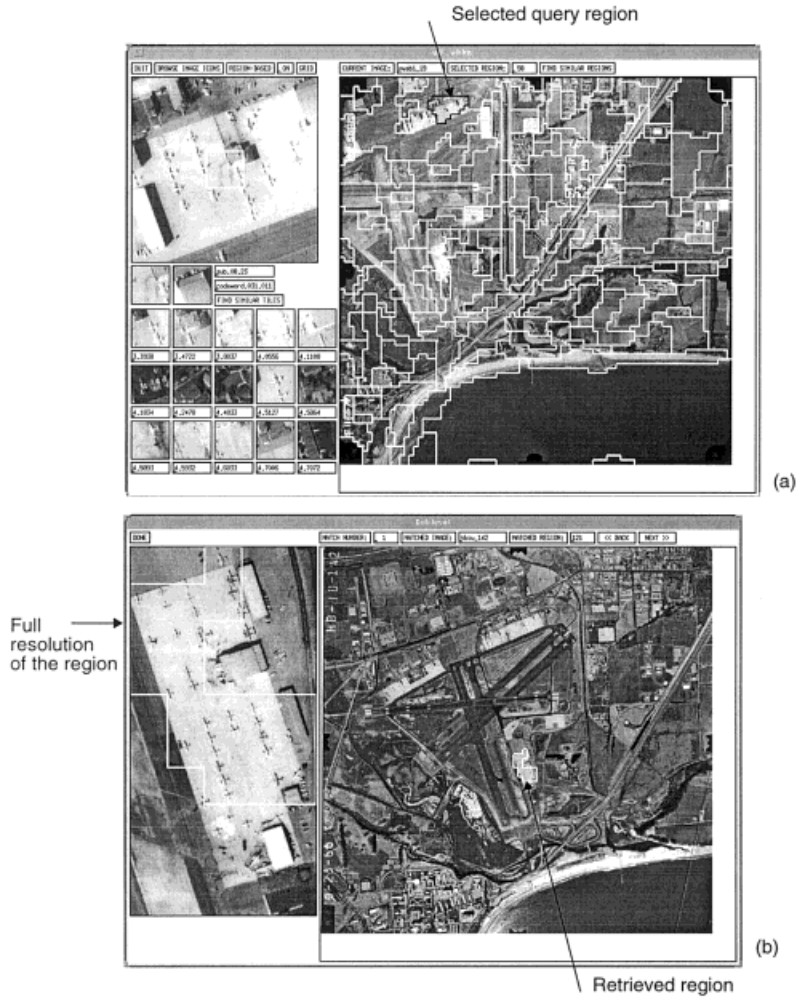


FIG. 11. An example of the region-based search. **a:** The down-sampled version of a segmented large airphoto. **b:** Retrieval result. Both the query and retrieval patterns are from the airport areas.

within the subspace. If a class has a large number of data points, it requires more codewords to represent all samples. This results in an unbalanced tree structure for searching and indexing (Fig. 6a). An example of indexing the 2-D image features is shown in Figure 6b. As can be seen, the goal of the first level of the indexing tree is to identify a subspace within which the search and retrieval should be constrained in terms of pattern similarity. On the other hand, the second level of the indexing tree mainly focuses on exploring the data distribution (or density) within the subspace, so a set of the nearest neighbors (within the smaller cluster) can be quickly identified and retrieved.

4.2. Experimental Results

In the current implementation, the texture thesaurus contains 60 similarity classes and about 950 codewords. In constructing the first level of the indexing tree, we use the Kohonen feature map (Section 3.1.1) with 900 neurons organized on a two-dimensional lattice. The number of neurons used here was empirically determined by considering the number of similarity classes in the training data. These neurons classify the input feature vectors into one of the 60 subspaces, and within each subspace the number of codewords obtained by GLA ranges from 5 to 60. Figure 7 shows some examples of the visual codewords in the texture thesaurus designed for airphotos. As we can see, the patterns within the same class are visually similar.

This texture thesaurus is used to process all the aerial photographs in the database in the following way:

- When an airphoto is ingested into the database, the texture features of the image tiles (64×64 blocks of pixels) and segmented regions are extracted.
- These features are compared with the codewords in the thesaurus. Once the best match is identified, a two-way link between the image tile (or region) and the corresponding codeword is created and stored as the image meta-data.

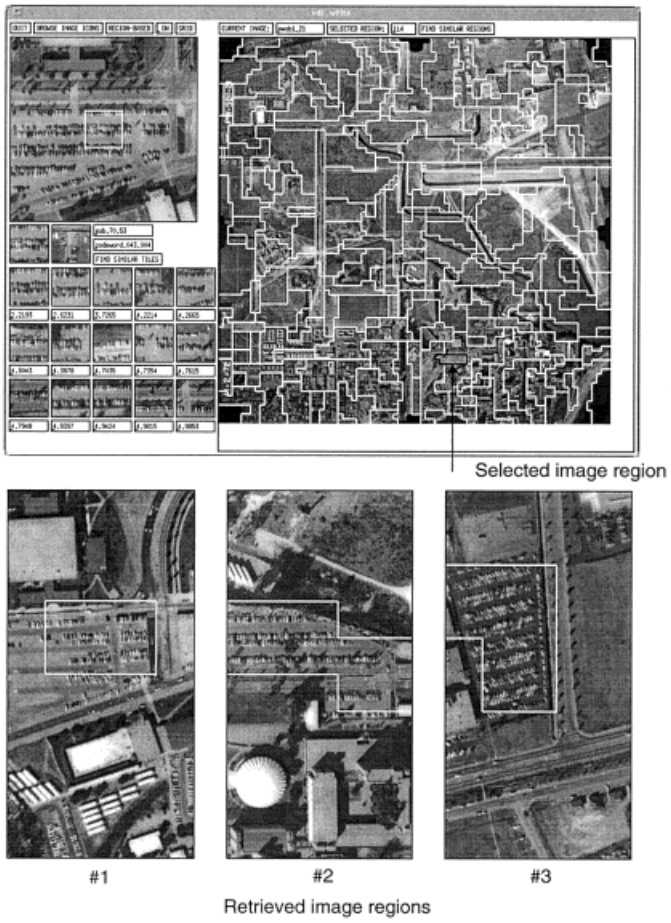


FIG. 12. An example of region-based retrieval on the parking lots areas.

- During query time, the feature vector of the selected pattern is used to search for the best matching codeword, and by tracing back the links to it, all candidate matches in the database can be retrieved.
- In the final stage, the Euclidean distance measure on the feature vector is used to rank-order the similarity with the query pattern, and the best matches are shown to the user.

Figure 8 illustrates the idea of using the texture thesaurus for content-based image indexing. In the current implementation, we have 40 large aerial photographs in the database which contain about 280,000 image tiles and 6,000 regions. Using the texture thesaurus model, searching for similar image tiles takes only a couple of seconds. Contrast this with a sequential search which would have taken 5–10 minutes. Note that implementing this on a traditional database system would not have resulted in any improvement because of the high dimensionality of the feature vectors.

Figure 9 shows a snapshot of the image browsing tool. Initially, this system displays a down-sampled version of the large airphotos (about 5K $\times$ 5K) on the screen, and users can select any region of interest to see it at the original resolution. The current implementation provides both tile-based and region-based search capabilities using texture image primitives. Tile-based retrieval is useful in searching for local image features, such as highway intersections and bridges, which are shown in Figure 9. Figure 10 shows more examples of the tile-based search. By clicking on any retrieved tiles, a full-resolution surrounding neighborhood and the corresponding large airphotos will be displayed on the screen. On the other hand, the region-based search is appropriate for larger geographic features. Figure 11 shows an example of retrieving a similar region from an airport; Figures 12 and 13 illustrate some more region-based retrievals, including image patterns of parking lots, marked numbers, highways, houses, and vegetation areas.

5. Discussion

An implementation of a texture-based image retrieval system for browsing large-scale aerial photographs is presented. The components required for constructing such a system are identified and a prototype system has been implemented. A hardware implementation of this feature

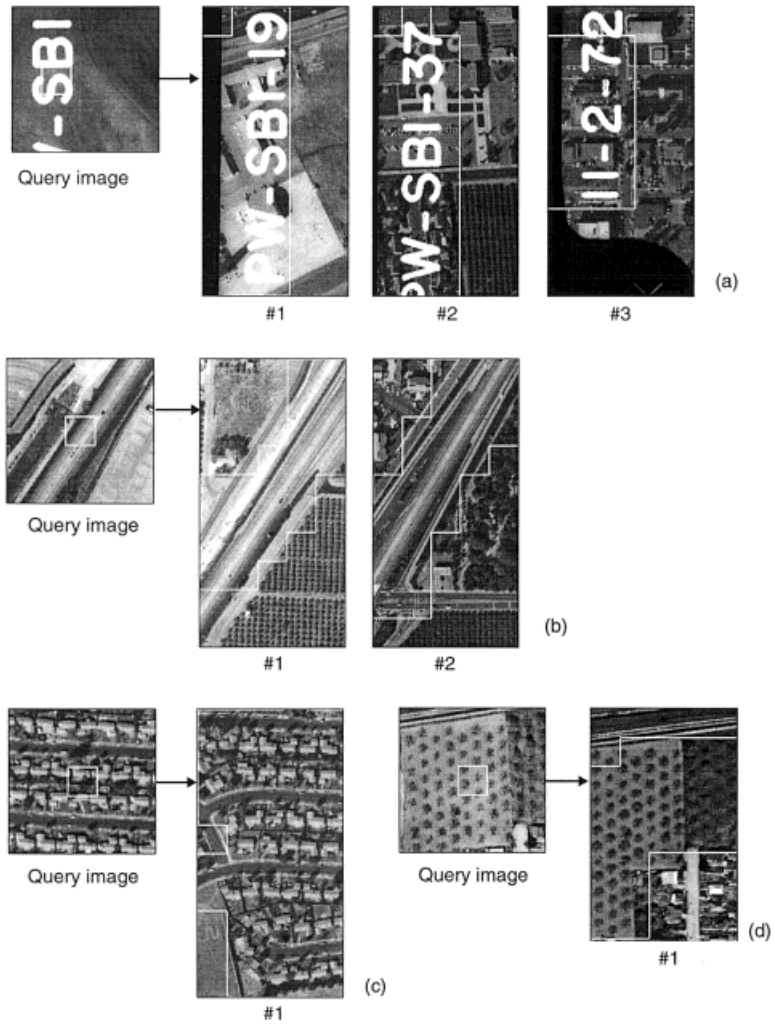


FIG. 13. **a:** The region retrievals from areas containing an image identification number. **b:** A highway scene. **c:** Houses in residential areas. **d:** An example of retrieving a similar vegetation patch.

extraction algorithm is under progress. A novel image segmentation and grouping algorithm, which is appropriate for large images and database retrieval applications, is proposed. Currently, we are extending these techniques to deal with color and multispectral data, as well as rotation invariance (Haley & Manjunath, 1995). Some examples of color image segmentation can be found at *http://vivaldi.ece.ucsb.edu/projects.html.* To account for scale change, we are currently investigating the use of pixel resolution information (available for most geographic images) in designing the pre-processing filters for texture feature detection.

Similarity measures in the feature space is an important issue in database applications. We have proposed the use of a hybrid neural network learning algorithm to cluster feature vectors while preserving the topology and visual similarity. As demonstrated by a performance evaluation on the Brodatz texture database, the proposed learning algorithm enhances the retrieval performance significantly. The resulting hierarchical representation also facilitates fast search on a large image collection. Further research is needed on metric representations of perceptual similarity.

By combining the learning similarity algorithm and a hierarchical vector quantization scheme, we have created a texture image thesaurus to process and annotate all the aerial photographs stored in the database. The resulting two-level indexing tree provides a very efficient search while maintaining a good retrieval performance. Although it is currently implemented using texture features for browsing airphotos, the concept of this image thesaurus model can be extended to other types of image attributes and spatio-temporal groupings of basic features. We are currently extending this approach to include a larger variety of data, including SPOT and LANDSAT images.

Acknowledgments

This research was supported, in part, by an NSF Digital Library grant IRI-94-11330.

References

Alexandrov, A. D., Ma, W. Y., Abbadi, A. E., & Manjunath, B. S. (1995). Adaptive filtering and indexing for image databases. *Proceedings of SPIE Conference on Storage and Retrieval for Image and Video Databases*—III, San Jose, CA (pp. 12–23).

Brodatz, P. (1966). *Textures: A photographic album for artists and designers.* New York: Dover.

Gersho, A., & Gray, R. M. (1992). *Vector quantization and signal compression.* Kluwer Academic Publishers.

Haley, G. M., & Manjunath, B. S. (1995). Rotation-invariant texture classification using modified Gabor filters. *Proceedings of IEEE International Conference on Image Processing,* Washington, DC (Vol. I, pp. 262–265).

Haykin, S. (1994). *Neural networks.* New York: Macmillan.

Katsavounidis, I., Kuo, C.-C. J., & Zhang, Z. (1994). A new initialization techniques for VQ codebook design. *Proceedings of the 28th Asilomar Conference on Signal, System & Computers,* Pacific Grove, CA (pp. 706–710).

Kohonen, T. (1982). Self-organized formation of topologically correct feature maps. *Biological Cybernetics, 43,* 59–69.

Kohonen, T. (1990). The self-organizing map. *Proceedings of IEEE, 78*(9), 1464–1480.

Krzanowski, W. J. (1995). *Recent advances in descriptive multivariate analysis* (chap. 2). Oxford Science Publications.

Ma, W. Y., & Manjunath, B. S. (1995). Image indexing using a texture dictionary. *Proceedings of SPIE Conference on Image Storage and Archiving System,* Philadelphia, PA (Vol. 2606, pp. 288–298).

Ma, W. Y., & Manjunath, B. S. (1996). Texture features and learning similarity. *Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition,* San Francisco, CA (pp. 425–430).

Ma, W. Y., & Manjunath, B. S. (1997). *Edge flow: A framework of boundary detection and image segmentation. Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition,* Puerto Rico. (pp. 744–749).

Manjunath, B. S., & Ma, W. Y. (1996). Texture features for browsing and retrieval of image data. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 18*(8), 837–842.

Minka, T. P., & Picard, R. W. (1995). *Interactive learning using a ''society of models''* (Tech. Rep. No. 349). MIT Media Laboratory.

Niblack, W., et al. (1993). The QBIC project: Querying images by content using color, texture, and shape. *Proceedings of SPIE Conference on Storage and Retrieval for Image and Video Databases,* San Jose, CA (Vol. 1908, pp. 173–187).

Niblack, W., et al. (1995, September). Query by image and video content: The QBIC system. *IEEE Computer,* pp. 23–32.

Pentland, A., Picard, R. W., & Sclaroff, S. (1994). Photobook: Tools for content based manipulation of image databases. *Proceedings of SPIE Conference on Storage and Retrieval for Image and Video Databases*—II, San Jose, CA (Vol. 2185, pp. 34–47).

Picard, R. W. (1995). Light-years from lena: Video and image libraries of the future. *Proceedings of IEEE International Conference on Image Processing,* Washington, DC (Vol. I, pp. 310–313).

Smith, T. R. (1996, May). A digital library for geographically referenced materials. *IEEE Computer, 29,* 54–60.

Strobel, N., Mitra, S. K., & Manjunath, B. S. (1995). An approach to efficient storage, retrieval, and browsing of large scale image databases. *Proceedings of SPIE Conference on Image Storage and Archiving System,* Philadelphia, PA (Vol. 2606, pp. 324–335).