Texture Features and Learning Similarity

W. Y. Ma and B. S. Manjunath
Department of Electrical and Computer Engineering
University of California, Santa Barbara, CA 93106
wei@chandra.ece.ucsb.edu,manj@surya.ece.ucsb.edu

Abstract

This paper addresses two important issues related to texture pattern retrieval: feature extraction and similarity search. A Gabor feature representation for textured images is proposed, and its performance in pattern retrieval is evaluated on a large texture image database. These features compare favorably with other existing texture representations. A simple hybrid neural network algorithm is used to learn the similarity by simple clustering in the texture feature space. With learning similarity, the performance of similar pattern retrieval improves significantly.

An important aspect of this work is its application to real image data. Texture feature extraction with similarity learning is used to search through large aerial photographs. Feature clustering enables efficient search of the database as our experimental results indicate.

1 Introduction

Texture as a primitive visual cue has been studied for over twenty years now. Various techniques have been developed for texture segmentation, texture classification, shape from texture, and texture synthesis. In addition, much work has been done in the context of human texture perception and modelling the low level processing. Although texture analysis has a long history, its applications to real image data has been very limited to-date.

An important and emerging application where texture analysis can make a significant contribution is the area of content based retrieval in large image and video databases. Using texture as a visual feature, one can query a database to retrieve similar patterns. Example queries include: "retrieve all landsat images with less than 20% cloud cover" or "retrieve all citrus plantations which look like this image from the aerial photographs of southern California". Texture classification and segmentation schemes are important in answering such queries. However, what distinguishes 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, the input query image is used as an initial seed to browse through a large collection of data, and similarity based retrieval is needed in such cases. In this context the main contributions of this paper are:

- A Gabor texture feature set is proposed and a comprehensive evaluation and comparison between several texture features is provided on a large texture database. We conclude that our Gabor texture features provide the best overall retrieval accuracy.
- A simple hybrid learning algorithm is proposed to compare patterns in the texture feature space. This significantly enhances the retrieval performance.
- A practical application to image browsing is illustrated on a large collection of aerial photographs. Gabor features combined with clustering results in a natural hierarchical indexing data structure for fast browsing of image data.

This paper is organized as follows: The next section introduces a class of self-similar Gabor functions. A simple filter design strategy is suggested for selecting the filter parameters. In Section 3 a comprehensive comparison of different multiscale texture feature is provided. Section 4 discusses learning in the texture feature space and we conclude with some discussions in Section 5.

2 Gabor Texture Features

Gabor functions are Gaussians modulated by complex sinusoids. In two dimensions they take the form [1]:

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

The Gabor filter masks can be considered as orientation and scale tunable edge and line (bar) detectors. The statistics of these microfeatures in a given region can be used to characterize the underlying texture information. A class of such self-similar functions, referred to as Gabor wavelets in the following discussion, is now considered. This self-similar filter dictionary can be obtained by appropriate dilations and rotations of $g(x,y)$ through the generating function,

$$g_{mn}(x,y) = a^{-m}g(x',y'), \quad a>1, \quad m,n = \text{integer} \quad (2)$$

$$x' = a^{-m}(x\cos\theta + y\sin\theta), y' = a^{-m}(-x\sin\theta + y\cos\theta),$$

where $\theta = n\pi/K$, K being the number of orientations. Let U_l and U_h denote the lower and upper center frequencies of interest, and S be the number of scales in the multi-resolution decomposition. Then the following design ensures that the half-peak magnitude support of the filter responses in the frequency spectrum touch each other. Let $\sigma_u = 1/2\pi\sigma_x$ and $\sigma_v = 1/2\pi\sigma_y$. Then,

1063-6919/96 $5.00 © 1996 IEEE

$$a = (U_h/U_l)^{-1/(S-1)}, \quad (3)$$

$$\sigma_u = ((a-1)U_h)/((a+1)\sqrt{2\ln 2}), \quad (4)$$

$$\sigma_v = \tan\left(\frac{\pi}{2k}\right)\left[U_h - 2\ln 2\left(\frac{\sigma_u^2}{U_h}\right)\right]\left[2\ln 2 - \frac{(2\ln 2)^2\sigma_u^2}{U_h^2}\right]^{-\frac{1}{2}}, \quad (5)$$

$$W = U_h, \ \theta = n\pi/K \text{ and } m = 0, 1, \ldots, S-1 . \quad (6)$$

In the experiments on the Brodatz album, we use four scales ($S = 4$) and six orientations ($K = 6$).

2.1 Feature Representation and Distance Measure

Given an image $I(x, y)$, we compute

$$W_{mn}(x, y) = \int I(x_1, y_1)\, g_{mn}^*(x - x_1, y - y_1)\, dx_1 dy_1, \quad (7)$$

where * indicates the complex conjugate. A texture region is now characterized by a the mean μ_{mn} and the standard deviation σ_{mn} of the energy distribution of the transform coefficients.

$$\mu_{mn} = \iint |W_{mn}(x, y)|\, dxdy , \quad (8)$$

$$\sigma_{mn} = \sqrt{\iint (|W_{mn}(x, y)| - \mu_{mn})^2\, dxdy} \quad (9)$$

A feature vector is now constructed using μ_{mn} and σ_{mn} as components. Recall that $S = 4$ and $K = 6$, resulting in a feature vector

$$\vec{f} = \left[\mu_{00}\ \sigma_{00}\ \mu_{01}\ \cdots\ \mu_{35}\ \sigma_{35}\right]^T . \quad (10)$$

Other representations such as using higher order or complex moments are also possible. However, our extensive experiments on a large data set indicate that the marginal improvement obtained by using higher order moments does not really justify the additional complexity. The complexity increases both in terms of added computations as well as increase in the feature dimensionality.

2.2 Retrieval Performance

Texture Database: The texture database used in the experiments consists of 116 different texture classes. Each image is 512×512 pixels. Each image is divided into 16 non-overlapping subimages, each 128×128 pixels in size, thus creating a database of 1856 texture images. These images are used in comparing the performance of different texture features.

In the experiments here, a query pattern is defined to be any one of the 1856 patterns in the database. This pattern is then processed to compute the feature vector as in (10). Note that there are 16 images per texture class in the database. Consider two image patterns i and j, and let $\vec{f}^{(i)}$ and $\vec{f}^{(j)}$ represent the corresponding feature vectors. Then the distance between the two patterns in the feature space is defined to be $d(i,j) = \sum_m \sum_n d_{mn}(i,j)$, where

$$d_{mn}(i,j) = \left|\frac{\mu_{mn}^{(i)} - \mu_{mn}^{(j)}}{\alpha(\mu_{mn})}\right| + \left|\frac{\sigma_{mn}^{(i)} - \sigma_{mn}^{(j)}}{\alpha(\sigma_{mn})}\right| \quad (11)$$

$\alpha(\mu_{mn})$ and $\alpha(\sigma_{mn})$ are the standard deviations of the respective features over the entire database. The distances are then sorted in increasing order and self matches are excluded. In the ideal case all the top 15 retrievals are from the same large image. The performance is measured in terms of the average retrieval rate which is defined as the average percentage number of patterns belonging to the same image as the query pattern in the top 15 matches. On the average 74.37% of the correct patterns are in the top 15 retrieved images.

2.3 Comparison with Other Texture Features

A detailed comparison with some of the other recently proposed multiresolution texture image features is made here. For the Gabor feature case, the entire 24×2 component feature vectors are used. The comparisons are made with the conventional pyramid-structured wavelet transform (PWT) features, tree-structured wavelet transform (TWT) features, and multiresolution simultaneous autoregressive model (MR-SAR) features.

The Gabor features give the best overall performance at close to 74% retrieval. This is closely followed by the MR-SAR features at 73% (using normalized Euclidean distance instead of Mahalanobis distance gave only 65%). As can be expected, the TWT features perform better (69.4%) than the PWT features (68.7%) but only by a 0.7% margin. Figure 1 shows a graph illustrating this retrieval performance as a function of number of top matches considered. More details can be found in [3].

3 Similarity Measures and Learning

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. Simple distance measures, such as the normalized Euclidean distance, on these features may not preserve the perceptual similarity. Computing the appropriate similarity measure can be treated as a learning problem. The goal of learning is to partition the original feature space into clusters of visually similar patterns. A large number of labeled image data and the associated feature vectors are used during the learning phase. When a texture pattern 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.

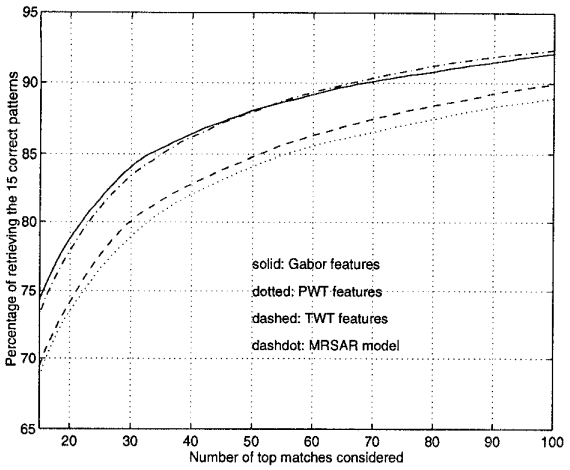


Figure 1: Comparison of retrieval performance of different texture features.

3.1 Stage I: Kohonen Feature Map

In the first stage of the learning process, a self-organizing feature-mapping algorithm [7] is used. The network consisting of two layers transforms the input features of arbitrary dimension into a two-dimensional discrete map. 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 feature map achieves dimensionality reduction while preserving topology.

The weights in the network are adjusted based on competitive learning [8]. The training of the network is performed by randomly presenting a feature vector x to the input layer of the network and adjusting the connection weight vectors $\boldsymbol{m}_i$ according to the updating rules given below. In the absence of any additional information about cluster formation, the minimum Euclidean distance criterion is used to decide which neuron is activated. Let us denote the fired neuron as c, then

$$c = \arg\min_i \|x(n) - \boldsymbol{m}_i\|, \; i = 1, 2, \ldots, N. \tag{12}$$

The updating of the weights associated with the neurons is only performed within a neighborhood set $N_c(n)$ of the neuron c. The updating rule can be formulated as:

$$\boldsymbol{m}_i(n+1) = \begin{cases} \boldsymbol{m}_i(n) + \alpha(n)\,[x(n) - \boldsymbol{m}_i(n)], i \in N_c(n) \\ \boldsymbol{m}_i(n), i \notin N_c(n) \end{cases} \tag{13}$$

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 our experiments, 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.

3.1.1 Stage II: Learning vector quantization (LVQ)

Following the self-organizing stage, a labelled set of training feature vectors are presented to the network. Class labels are assigned to the second layer of neurons in the feature map by majority voting. This is followed by a fine tuning of the network to improve the classification accuracy. Fine tuning is done using a learning vector quantization (LVQ3) [7, 8] algorithm which 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 the class label associated with the ith neuron be represented by C_i. 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 x is considered to be within the window and the following updates are computed if the relative distances satisfy the following condition $\min(d_i/d_j, d_j/d_i) > \gamma$.

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

$$\begin{aligned} \boldsymbol{m}_i(n+1) &= \boldsymbol{m}_i(n) - \alpha(n)\,[\boldsymbol{x}(n) - \boldsymbol{m}_i(n)] \\ \boldsymbol{m}_j(n+1) &= \boldsymbol{m}_j(n) + \alpha(n)\,[\boldsymbol{x}(n) - \boldsymbol{m}_j(n)] \end{aligned} \tag{14}$$

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

$$\boldsymbol{m}_k(n+1) = \boldsymbol{m}_k(n) + \varepsilon\alpha(n)\,[x(n) - \boldsymbol{m}_k(n)], k \in \{i, j\} \tag{15}$$

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 $\varepsilon = 0.3$.

3.2 Performance Evaluation

We (again) use the Brodatz database to evaluate the performance of the learning algorithms for texture similarity. The training dataset is obtained by partitioning the 512x512 images into 49 subimages (with overlap), and choosing a subset of them. The subimages are all 128x128 pixels, centered on a 7x7 grid over the original image. The first 33 subimages are used for the training set and the last 16 for testing.

The 116 texture images are grouped into 32 different classes, each class containing between 2-6 similar textures. This classification was done manually by researchers in the laboratory and Table 1 shows these various classes and the corresponding textures. While it is possible that different groups of people 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.

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		

Table 1. Texture clusters used in learning similarity measures (identified by human subjects as being visually similar within each cluster).

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 classes (clusters) to be learnt. After the training stage, the majority rule is used to assign a class label to each of the neurons. Within a given cluster, the Euclidean distance is used to compute the similarity.

Figure 2 illustrates an evaluation now based on the 32 clusters that were used for training and labelling. Here the differences are quite striking. Whereas 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 3 shows some retrieval examples with and without learning similarity measures, which clearly illustrate the superiority of the learning algorithm. Perhaps it is not surprising to expect much better results as the network uses additional information not used when patterns were retrieved based only on weighted Euclidean distance. Another important point to note is that the clustering can be performed in a hierarchical manner, thus creating a tree representation. Searching through a tree is much more efficient, and resolves some of problems related to indexing in high dimensional feature space. Traditional data structures do not generalize well into high dimensions (greater than 20) and the feature vectors we are using are 48-dimensional. Search mechanisms such as the R* trees under such conditions are no better than a linear search [9].

3.3 Browsing Large Aerial Photographs

Query based on texture properties will have many applications in image and multi-media databases. Here we describe with an example our current work on incorporating these features for browsing large satellite images and air photos. Typical images in such a database range from few megabytes to hundreds of megabytes, posing challenging problems in image analysis and visualization of data. Con-

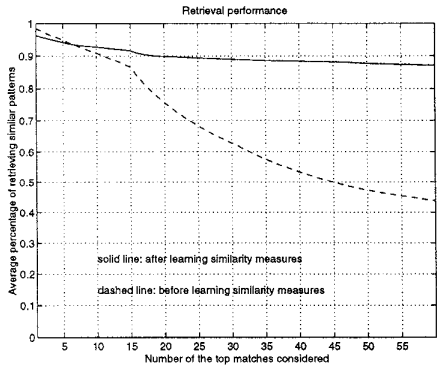
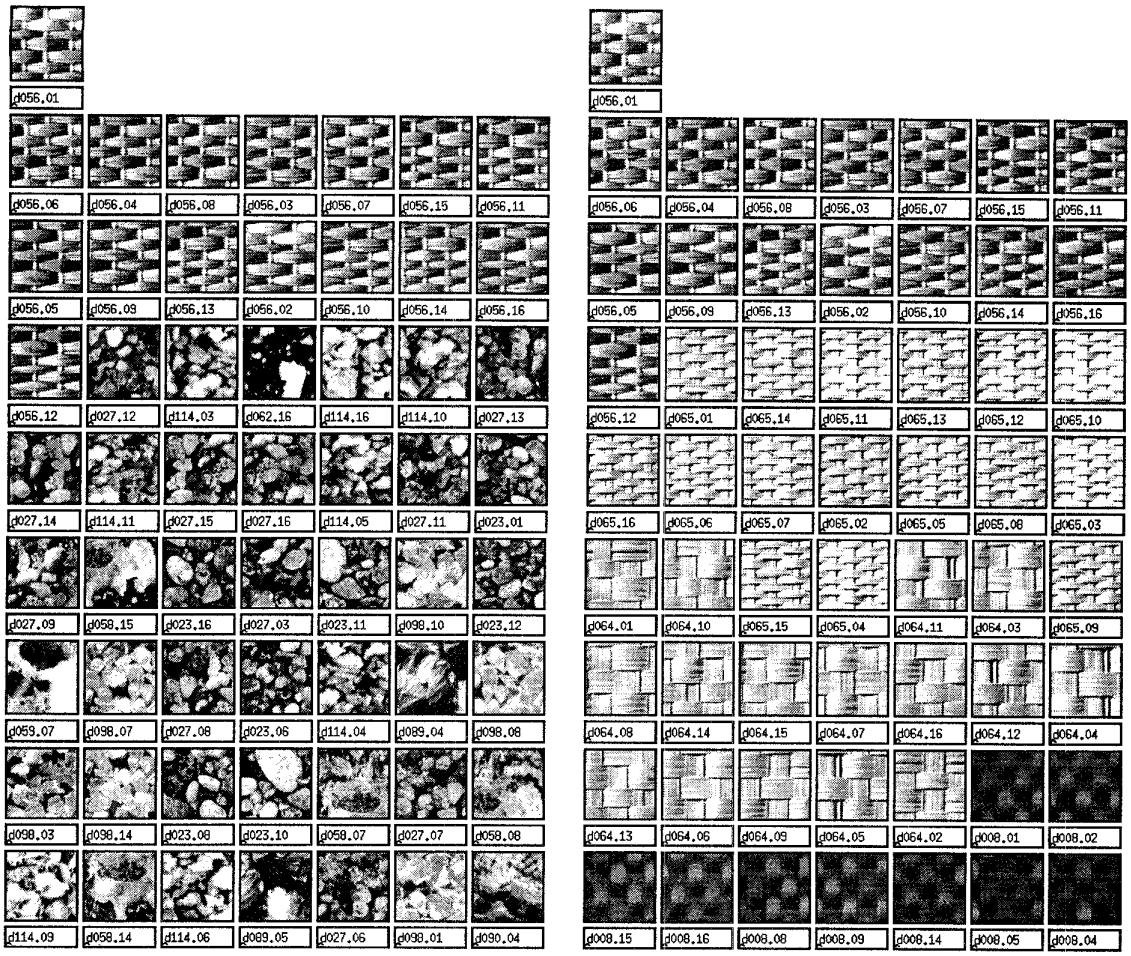


Figure 2: Retrieval performance before and after learning similarity measures. Notice the significant improvement in similarity based retrieval with learning.

tent based retrieval will be very useful in this context in answering queries such as "Find a vegetation patch that looks like this region".

We are currently investigating the use of texture primitives to accomplish rapid content based browsing within an image or across similar images. Figure 4 shows an example of browsing 5248 x 5248 air photos. Initially, a coarse subsampled version of the image is loaded on the display monitor. The original image is analyzed in blocks of pixels (e.g., 128x128 or 64x64) and the texture features are computed and stored as part of the image "meta-data". The user can select any position and display the full resolution version at that location, and use that pattern to search for similar looking regions.

The current database now consists of about 40 such large airphotos, with more than 250,000 *texture objects* to search and retrieve. Our initial results in browsing such a large dataset using the learning approach are very encouraging. In our current implementation, search and retrieval is almost *real time* (2-3 seconds, including search and sorting, on a SUN Sparc20 machine). In contrast, a linear search



(a) before learning similarity measures (b) after learning similarity measures

Figure 3: 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.

would have taken 3-5 minutes (and indexing structures would not have been very useful either, as noted earlier). Ground truth is not available to provide a quantitative evaluation similar to the Brodatz album. However, as the retrieval results in Figure 4 indicate, our approach using Gabor features appears to be quite successful on a wide range of image patterns. Retrieval examples include vegetation patches (such as citrus plantations), parking lots with cars, highways and intersections, and even some imprinted text on the images (see the web pages at http:// vivaldi.ece.ucsb.edu for more examples). The texture features are now being incorporated into a database system which allows both textual and texture pattern queries.

4 Conclusions

A Gabor wavelet based texture analysis scheme is proposed and its application to a large texture image database is demonstrated. A comprehensive performance evaluation of the method is given using a large number of textures, and a comparison with some of the well known multiresolution texture classification algorithms is made. The experimental results indicate that the Gabor feature are quite robust and compare favorably with other texture features for pattern retrieval.

Comparing patterns in the feature space is another important issue that is addressed. We propose the use of a hybrid neural network learning algorithm to cluster feature vectors while preserving the topology and visual similarity. As demonstrated by the experimental results, this combina-

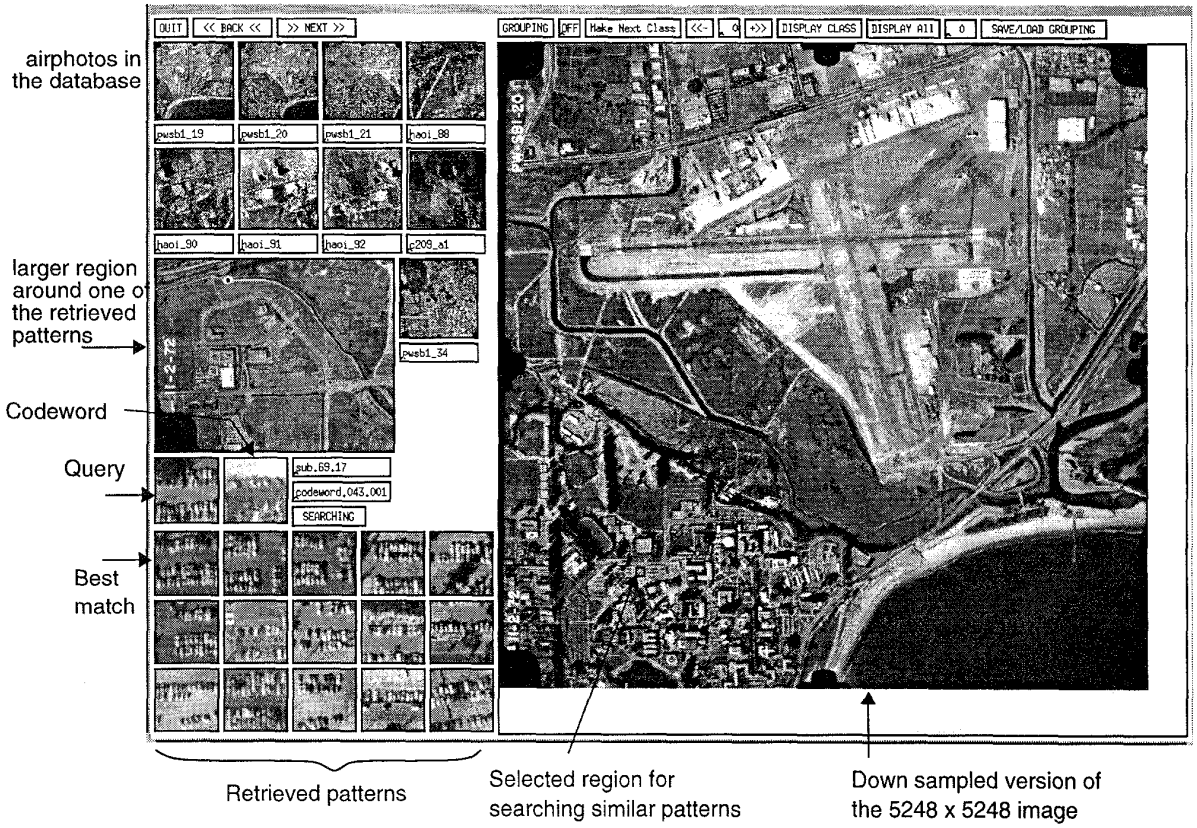


Figure 4: Snapshot of an aerial photograph browsing demonstration. The example shown indicates a query pattern containing a parking lot. Next to the query is the image codeword used to index the database. The browser can retrieve almost 99% of all the parking lots in the aerial photo database.

tion of Gabor features and learning significantly enhance the retrieval performance. The resulting hierarchical representation, which can be easily generalized into a tree structure, also facilitates fast browsing on a large image collection. The application to browsing large aerial photographs is among the first of its type to demonstrate the usefulness of texture features for content based retrieval on real image data. Our current research is to extend these methods to deal with scale and rotation invariance [10] and incorporating automated segmentation of images into homogeneous texture regions.

Acknowledgments: We thank Professor Picard for providing the software for computing the MR-SAR texture features. This research was supported in part by a grant from NSF IRI94-11330 and by NASA under NAGW 3951.

5 REFERENCES

[1] J. G. Daugman, "Complete discrete 2-D Gabor transforms by neural networks for image analysis and compression," IEEE Trans. ASSP, vol. 36, pp. 1169-1179, July 1988.

[2] T. Chang, C.-C. Jay Kuo, "Texture analysis and classification with tree-structured wavelet transform," IEEE Trans. Image Processing, vol. 2, no. 4, pp. 429-441, October 1993.

[3] B. S. Manjunath and W. Y. Ma, "Texture features for browsing and retrieval of image data," CIPT TR-95-06, July 1995 (http://vivaldi.ece.ucsb.edu). also to appear in the special issue on Digital Libraries, IEEE T-PAMI, November 1996.

[4] F. Liu, R. W. Picard, "Periodicity, directionality, and randomness: Wold features for image modeling and retrieval," MIT Media Lab Technical Report No. 320.

[5] R. W. Picard, T. Kabir, and F. Liu, "Real-time recognition with the entire Brodatz texture database," Proc. IEEE Conf. CVPR '93, New York, pp. 638-639. June, 1993.

[6] W. Y. Ma, B. S. Manjunath, "A comparison of wavelet features for texture annotation," Proc. IEEE Intl. Conf. on Image Processing '95, Washington D.C., vol. II, pp. 256-259, October, 1995.

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

[8] S. Haykin, Neural Networks, Macmillan Publishing, 1994.

[9] A. D. Alexandrov, W. Y. Ma, A. El Abbadi, B. S. Manjunath, "Adaptive filtering and indexing for image databases," Proc. of SPIE, Vol. 2420, pp. 12-23, San Jose, CA, February 1995.

[10] G. M. Haley, B. S. Manjunath, "Rotation invariant texture classification using the modified Gabor filters," Proc. IEEE Intl. Conf. on Image Processing '95, Washington D.C., vol. I, pp. 262-265, October, 1995.