01-US

Shu-Ching Chen, Mei-Ling Shyu, and Chengcui Zhang, "An Unsupervised Segmentation Framework For Texture Image Queries," The 25th IEEE Computer Society International Computer Software and Applications Conference (COMPSAC), Chicago, Illinois, USA, pp. 569-573, October 8-12, 2001.

An Unsupervised Segmentation Framework For Texture Image Queries*

Shu-Ching Chen
Distributed Multimedia Information System Laboratory
School of Computer Science
Florida International University
Miami, FL 33199, USA
chens@cs.fiu.edu

Mei-Ling Shyu
Department of Electrical and
Computer Engineering
University of Miami
Coral Gables, FL 33124, USA
shyu@miami.edu

Chengcui Zhang
Distributed Multimedia Information System Laboratory
School of Computer Science
Florida International University
Miami, FL 33199, USA
czhang02@cs.fiu.edu

Abstract

In this paper, a novel unsupervised segmentation framework for texture image queries is presented. The proposed framework consists of an unsupervised segmentation method for texture images, and a multi-filter query strategy. By applying the unsupervised segmentation method on each texture image, a set of texture feature parameters for that texture image can be extracted automatically. Based upon these parameters, an effective multi-filter query strategy which allows the users to issue texture-based image queries is developed. The test results of the proposed framework on 318 *texture images obtained from the MIT VisTex and Brodatz database are presented to show its effectiveness.*

1. Introduction

Segmentation is an important part of the computer vision and image analysis, wherein regions of interest are identified and extracted for future processing. The definition of suitable similarity and homogeneity measures is a fundamental task in many important applications, ranging from remote sensing to similarity-based retrieval in large image databases such as the query by image content (QBIC) system [4].

Texture segmentation involves the identification of uniform textured regions in an image. Many techniques have been used for the analysis of textures such as [5, 6]. With the restriction to a set of known textures, retrieval and segmentation problems are essentially reduced to a supervised classification task, which is amenable for standard techniques from pattern recognition and statistics. Techniques used for image segmentation include simple statistical models to obtain estimates of probability density functions [7], and intensity and texture measures [10], etc. Local statistics and edge information have also been used to segment and distinguish regions of interest from the background [9]. Segmentation techniques can be grouped under split and merge methods [8], region growing methods[1], and stochastic model based methods [2]. The main approach taken in most of the emerging techniques includes the step to choose a strategy to estimate the parameters of distributions, which is invariably to be the maximum likehood (ML) estimation or maximum a posteriori (MAP) estimation. However, computing the exact MAP estimate of the class label field is considered a hard problem. Also, no methods in the literature can compute the MAP estimates of the class parameters as well as the pixel labels simultaneously.

There exists a tight relationship between similarity-based texture image retrieval and unsupervised texture segmentation. Image retrieval often requires to select those images in a database which are most similar to a given query image, while the goal of segmentation is to partition a given image into maximally homogeneous regions. Therefore these tasks are closely related to similarity measures, since homogeneity can be defined as the average similarity between pairs of local texture patches within a region. In this paper,

*This research was supported in part by NSF CDA-9711582.

we propose an unsupervised texture segmentation method which can recognize the variability of content description depending on the complexity of the image regions and effectively address it. The proposed method considers the problem of segmentation as a joint estimation of the partition and class parameters. This class parameterization enables us to compute the optimal parameters using a simple least squares technique, and the class descriptions are amenable to direct estimation of their parameters without resorting to expensive numerical optimization procedures. By considering both the partition and the class parameters as random variables and estimating them jointly, their MAP estimates are computed simultaneously. In our framework, we first segment each of the texture images into classes (usually 2 classes), and extract texture features of each class simultaneously by generating the class parameters during the process of segmentation. Based on the database of the texture features, a multi-filter query mechanism is developed to filter out most of the biased texture images that are far different from the example query texture image at the very beginning of query, which can greatly reduce the overhead. The test results are based on the *318* texture images obtained from the MIT VisTex Texture database [11] and Brodatz database [12].

The rest of the paper is organized as follows. In Section 2, the unsupervised texture segmentation method is presented, and the feature parameters obtained by segmentation are described in details. Section 3 explains the query strategy. Section 4 gives the test results and the discussions. Conclusions and future work are given in Section 5.

2. Unsupervised Texture Segmentation

In the proposed unsupervised texture segmentation framework, the partition and the class parameters are treated as random variables. The method of partitioning a still image starts with a random partition and employs an iterative algorithm to estimate the partition and the class parameters jointly [3].

2.1. Segmentation Method

Suppose the image is of size $N_r \times N_c$ with intensities given by $Y = \{y_{ij}: 1 \leq i \leq N_r, 1 \leq j \leq N_c\}$ and there are two classes in the image. Let the partition variable be $\boldsymbol{c} = \{\boldsymbol{c}_1, \boldsymbol{c}_2\}$, and the classes be parameterized by $\boldsymbol{\theta} = \{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2\}$. Also, suppose all the pixel values y_{ij} belonging to class k $(k = 1, 2)$ are put into a vector $\mathbf{y_k}$. Each row of the matrix Φ is given by $(1, i, j, ij)$ and a_k is the vector of parameters $(a_{k0}, ..., a_{k3})^T$.

$$\begin{aligned} y_{ij} &= a_{k0} + a_{k1}i + a_{k2}j + a_{k3}ij, \forall (i,j)\ y_{ij} \in \boldsymbol{c}_k \\ \mathbf{y_k} &= \Phi a_k \end{aligned}$$

$$\hat{a_k} = (\Phi^T \Phi)^{-1} \Phi^T \mathbf{y_k}$$

The best partition is estimated as that which maximizes the *a posteriori probability (MAP)* of the partition variable given the image data Y. Now, the *MAP* estimates of $\boldsymbol{c} = \{\boldsymbol{c}_1, \boldsymbol{c}_2\}$ and $\boldsymbol{\theta} = \{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2\}$ are given by

$$\begin{aligned} (\hat{c}, \hat{\theta}) &= \operatorname*{Arg\,max}_{(\boldsymbol{c}, \boldsymbol{\theta})} P(\boldsymbol{c}, \boldsymbol{\theta} \mid Y) \\ &= \operatorname*{Arg\,max}_{(\boldsymbol{c}, \boldsymbol{\theta})} P(Y \mid \boldsymbol{c}, \boldsymbol{\theta}) P(\boldsymbol{c}, \boldsymbol{\theta}) \end{aligned}$$

Let $J(\boldsymbol{c}, \boldsymbol{\theta})$ be the functional to be minimized. With appropriate assumptions, this joint estimation can be simplified to the following form:

$$(\hat{c}, \hat{\theta}) = \operatorname*{Arg\,min}_{(\boldsymbol{c}, \boldsymbol{\theta})} J(\boldsymbol{c}_1, \boldsymbol{c}_2, \boldsymbol{\theta}_1, \boldsymbol{\theta}_2)$$

$$J(\boldsymbol{c}_1, \boldsymbol{c}_2, \boldsymbol{\theta}_1, \boldsymbol{\theta}_2) = \sum_{y_{ij} \in \boldsymbol{c}_1} -\ln p_1(y_{ij}; \boldsymbol{\theta}_1) + \sum_{y_{ij} \in \boldsymbol{c}_2} -\ln p_2(y_{ij}; \boldsymbol{\theta}_2)$$

The algorithm starts with an arbitrary partition of the data and computes the corresponding class parameters. Using these class parameters and the data, a new partition is estimated. Both the partition and the class parameters are iteratively refined until there is no further change in them. After the segmentation, a set of parameters describing both of the two classes is obtained automatically, and part of the parameters are selected for future query use.

2.2. Initial Partitions for Segmentation

The proposed segmentation method starts with a randomly generated initial partition. Hence, different initial partitions yield to different local minima. The smallest local minimum among them gives the desired solution though it may not be the global minimum. In the proposed framework, a number of local minima (e.g., *20*) are computed and the smallest local minimum is used. Since the computational requirement for each local minimum is very little, the overall computation needed for the best local minimum is not much. Two methods are used to generate those twenty initial partition candidates. By the *straight-line partition method*, the area of the original texture images is partitioned by an arbitrarily generated straight-line across the whole image area. Different areas separated by the straight-line represent different classes. In many cases, the randomly generated straight-line partitions are good enough to get the desired initial partition, but in many other cases it cannot work well. In order to obtain a good initial partition as quickly as possible, the *predefined template method* is also

used to generate the initial partitions. Eight predefined templates are selected as candidates in the selection of the desired initial partition.

Another important issue about the initial partition is how to select the "best" one among those candidates. The criteria for evaluating the candidates involve two aspects. One is the local minimum, and the other is the standard deviation of each class within a texture image. Two candidates are chosen when each of them has either the lowest local minimum or the lowest standard deviation. Then, the global minima of these two candidates are computed and the one with the lower global minimum is chosen as the final partition.

3. Query Strategy

After the segmentation on each texture image, a set of parameters for each image is obtained automatically. Some of these parameters are selected for query use. Since the proposed segmentation method uses the functions of the spatial coordinates of the pixels as the mathematical description of a class, those parameters related to spatial information should be able to represent the spatial distribution features of textures.

- Parameter AK: After the segmentation, each pixel within a texture has its class identification. For example, the class identification for each pixel is either 1 or 2 when there are two classes. As mentioned earlier, each class is parameterized by a vector of parameters $(a_{k0}, ..., a_{k3})^T$. In other words, this parameter vector contains not only the spatial distribution information of the texture, but also the information of intensity values within that class. Furthermore, among the four parameters in the vector, a_{k0} is usually far more larger than the other three. Therefore, given the number of classes is 2, two AK parameters (one for each class) are obtained for each texture.

- Parameter CV: It is the covariance matrix of matrix $cvecs\_n$ (n=1 or 2). This parameter represents the spatial distribution pattern of each class.

$$\begin{aligned} CV &= (CV\_1, CV\_2); \\ CV\_n &= (cvecs\_n \times cvecs\_n^T)/Nkn; \\ cvecs\_n &= (stkr\_n, stkc\_n)^T - mn\_n \times ones\_Nkn \end{aligned}$$

  where $stkr\_n$ and $stkc\_n$ are column vectors with each row being the $i - coordinate$ and $j - coordinate$ of $y_{ij} \in \boldsymbol{c}_n$, respectively. Here, mn_n is a column vector with 2 elements representing the means of the $i\_coordinates$ and $j\_coordinates$ of $y_{ij} \in \boldsymbol{c}_n$, and $ones\_Nkn$ is a unity vector of Nkn elements (i.e., all of them have the value **1**).

- Parameter VAR_MEAN: During the process of segmentation, the low-level features such as the variance and mean value for each class can also be obtained, which does not cause any excessive computation cost. Since the texture image is well segmented after the segmentation phase, using the low-level features of each class as the query criteria is expected to achieve good query results.

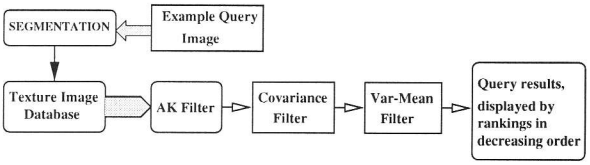


Figure 1. The multi-filter query architecture.

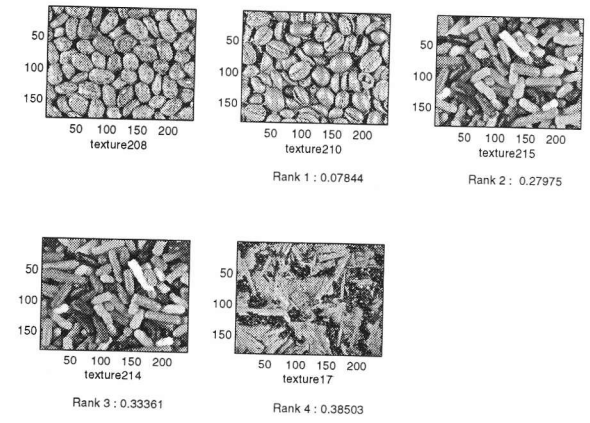
Since we use Euclidean distance for comparing two feature vectors, the smaller the dimension of vector is, the better the performance is. Notice that for each texture image, it has only two AK parameters. Though the information included in parameter AK may not be enough to achieve good query results, however, if it is used as the first level filter in the query strategy, the overall computation cost can be reduced significantly. Hence, a multi-filter query mechanism is developed in the proposed framework. Figure 1 shows the architecture of the multi-filter query strategy. As can be seen from this figure, the multi-filter query mechanism includes the AK filter, Covariance filter, and Var_Mean filter. The idea is to use the spatial distribution information obtained through segmentation to filter out those "bias" texture images, and use the classified VAR_MEAN to rank the retrieved images.

The ranking of the retrieved texture images is relatively simple. The sum of the weighted Euclidean distances on the VAR_MEAN for each class and the overall VAR_MEAN between the query image and the retrieved image is used to determine the ranking. The weights are derived from the experimental results.

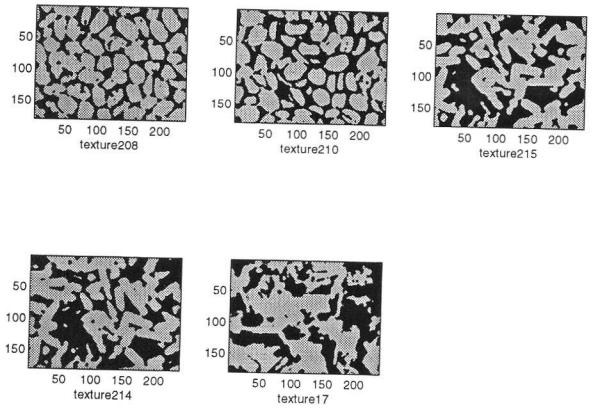
4. Test Results and Discussions

4.1. Image Retrieval Results

In order to test the performance of the proposed framework, *318* natural texture images mostly obtained from the MIT VisTex Texture database and Brodatz database are used. For the images from Brodatz, we partition each of the 512×512images into 6 subimages (with overlap). Each texture image is of size *240* rows and *180* columns. In the proposed framework, the similarity query is used. An example of the query looks like "Show me more texture images which are similar in texture patterns with the query image."



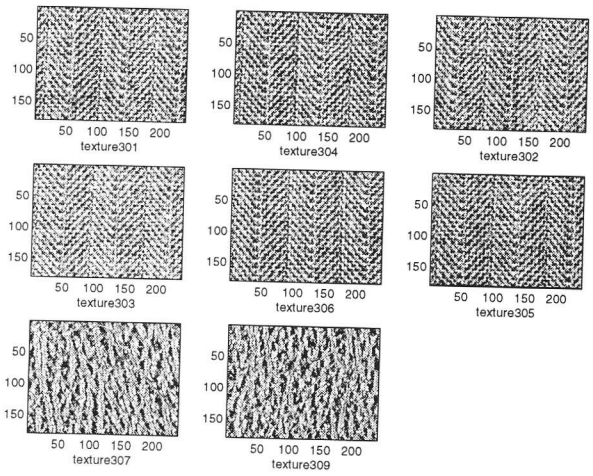
(a) Query results for query image texture208.



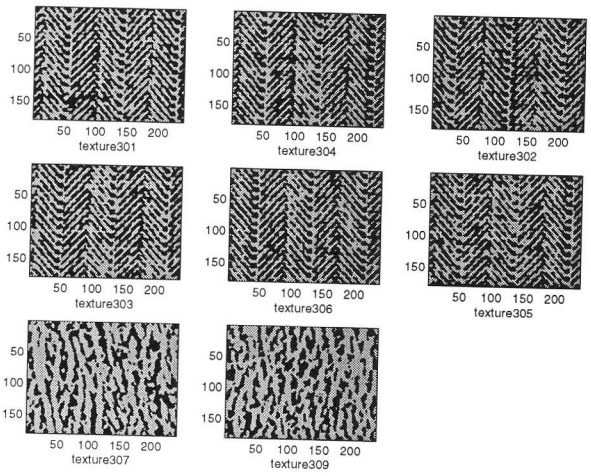
(b) Segmentation results

Figure 2. Texture query results after the segmentation. Example query image texture208 is on the top left. Matches of the images are listed from top left to bottom right in decreasing order of their similarities.

Figures 2(a)-(b) show the query results for example query image **texture208**, which is an image from MIT VisTex database. Figure 2(a) shows the first four original texture images being retrieved. The example query image **texture208** is on the top left, and the matches are listed from top left to bottom right in decreasing order of their similarities. The corresponding ranks of the matches are also given below the name of each original texture image as shown in Figure 2(a). The rank indicates how similar it is to the example query image. Figure 2(b) shows the segmentation results of those texture images in Figure 2(a). From the observations of the segmentation results, we can see that the texture pattern of image **texture210** is the closest to the query image **texture208**. The spatial distributions within each class are very similar to each other, as well as the variance_mean features. As for the retrieved images **texture215** and **texture214**, they have similar spatial distributions in texture patterns with the query image, but their variance_mean distributions are quite different from that of the query image. In addition, the spatial distribution of image **texture17** looks close to the query image, but not as close as **texture215** and **texture214** do. Another observation is that since the **texture210** is the closest to the query image, its corresponding rank value is almost four times higher than that of image **texture215**, which is significant enough to represent its high similarity with the query image.



(a) Query results for query image texture301.



(b) Segmentation results

Figure 3. Texture query results after the segmentation. Example query image texture301 is on the top left. Matches of the images are listed from top left to bottom right in decreasing order of their similarities.

Figure 3 shows another query results for example query image **texture301** which comes from the Brodatz database. The recall number is 8. It is clear that the top 5 matches include all the subimages which come from the same original image as the query image.

By analyzing the query results for the example query image, it is very promising to see that the proposed framework for texture segmentation and query can reasonably retrieve those texture images that have the similar texture patterns with the example query image. Moreover, since the proposed segmentation method is an unsupervised simultaneous partition and class parameter estimation algorithm, all the needed feature parameters can be obtained automatically and indexed offline without any user interactions. In the experiments, the accuracy of segmentation results for texture images exceeds **85** percent. In addition, the use of multi-filters (AK, CV and VAR_MEAN) greatly reduces the number of retrieved images at each step, which is essential to reduce the computation cost and get quick answers for the issued queries. For example, when **texture208** is used as the example query image, the number of retrieved images sharply dropped over **70** percent after the AK filter.

5. Conclusion and Future Work

In this paper, an unsupervised segmentation framework for texture image queries was proposed. By using a novel and effective segmentation method, a set of feature parameters for each class within an image is extracted automatically without any user interference. Based on these feature parameters, the proposed framework supports texture image queries effectively. Moreover, a multi-filter mechanism is used in the query procedure to greatly reduce the number of image candidates and at the same time, reduce the query processing time. Furthermore, applying the segmentation method on partitioning the natural image also gives good results.

One of the potentials of the proposed segmentation method is that it can also deal with the situation of multiple classes (more than two). The idea is to consider the number of classes s as another random variable. Our future work will focus on generalizing the proposed framework to handle the cases when the number of classes is more than two so that it can partition the image more reasonably and precisely, which is essential to the accuracy of the queries.

References

[1] J. M. Beulieu and M. Goldberg, "Hierarchy in picture segmentation: A stepwise optimization approach," *IEEE Trans. on PAMI*, 11(2):150-163, February 1989.

[2] R. Chellappa and A. Jain. *Markov Random Fields: Theory and Applications*. New York: McGraw-Hill Book Company, 1993.

[3] S.-C. Chen, S. Sista, M.-L. Shyu, and R. L. Kashyap, "Augmented Transition Networks as Video Browsing Models for Multimedia Databases and Multimedia Information Systems," *11th International Conference on Tools with Artificial Intelligence (ICTAI'99)*, pp. 175-182, Nov. 1999.

[4] M. Flickner et al., "Query by image and video content: The QBIC system," *IEEE Computer*, pp. 23-32, Sept. 1995.

[5] R. M. Haralick, "Statistical and structural approaches to texture," *Proceedings of IEEE*, vol. 67, pp. 786-804, 1979.

[6] W. Y. Ma and B. S. Manjunath, "Texture Features and Learning Similarity" *Proc. IEEE International Conference on Computer Vision and Pattern Recognition*, San Francisco, CA, pp. 425-430, June 1996.

[7] D. Nair and J. K. Aggarwal, "A focused target segmentation paradigm," *4th European Conference on Computer Vision*, vol. 1, pp. 579-588, Cambridge, UK, April 1996.

[8] T. Pavlidis. *Structural Pattern Recognition*, Springer-Verlag, 1991.

[9] K. Price, "Image segmentation: A comment on studies in global and local histogram-guided relaxation algorithms," *IEEE Trans. on PAMI*, 6(2):247-249, March 1984.

[10] M. Spann and A. Grace, "Adaptive segmentation of noisy and textured images," *Pattern Recognition*, 27(12): 1717-1733, December 1994.

[11] http://www-white.media.mit.edu/vismod/imagery/VisionTexture/vistex.html

[12] http://www.ux.his.no/ tranden/brodatz.html