FAST ESTIMATION OF THE NUMBER OF TEXTURE SEGMENTS USING CO-OCCURRENCE STATISTICS

Gou-Chol Pok[1], Jyh-Charn Liu[2], and Keun Ho Ryu[3]

[1]Computer Science Department, Yanbian University of Science and Technology,
Yanji City, Jilin Province, China 133000
[2] Department of Computer Science, Texas A&M University, College Station, Texas
[3]Database Laboratory, School of Electrical and Computer Engineering, Chungbuk National University,
Chongju, Chungbuk 361-763, Korea

ABSTRACT

Estimation of the number of clusters is an essential processing step for various applications. Existing approaches search for an optimal solution by computing and comparing a validity measure for all feasible configurations, and tend to under-estimate the number of clusters incorrectly. In this paper, we propose a fast and robust method to estimate the number of clusters without adopting the exhaustive search. Our scheme first extracts the relationship of neighboring features, and then uses this information to partition the clusters. The superb performance of the method is verified by the simulation results in determining the number of texture segments in the textured images.

1. INTRODUCTION

The objective of texture segmentation is to partition an image into homogeneous regions based on textural properties. Two most common approaches to segmentation are boundary detection of disjoint regions and clustering of regional features. Clustering is an unsupervised process without *a priori* information about class-specific references, where the number of clusters is explicitly specified as a user-defined parameter. For most clustering-based applications such as the unsupervised image segmentation, the number of clusters that fits the internal structure of a feature set needs to be automatically and accurately determined without user intervention. This *cluster validity problem* is a challenging issue with no known general solution [1].

Most of the currently existing techniques approach to the problem by computing and comparing a validity measure for all feasible configurations of clusters. Davies and Bouldin [2] developed a measure that searches for a configuration that minimizes the ratio of the average intra-cluster distance to the average inter-cluster distance. Havelicek and Tay [3] proposed a similar measure to compute the number of texture segments using wavelet transform-based texture features. Ray and Turi [4] also used the same measure to determine the number of clusters in color images. All these approaches adopt an exhaustive search for the optimal solution, and are not appropriate for real time applications such as texture segmentation or content-based image retrieval.

Instead of using the exhaustive search, our approach directly estimates the number of clusters using co-occurrence statistics of the features, which are captured via a 2-D histogram called a co-occurrence matrix. Usually texture features are represented as multi-dimensional vectors, and it is difficult to directly apply some useful scalar-oriented tools to analyze vectorial features. In order to address this issue, we apply the Kohonen's Self-Organizing Map (SOM) algorithm [5] to the multi-dimensional texture features, and obtain the transformed features that are encoded as scalars. This vector-to-scalar transform enables us to compute a 2-D histogram that provides useful information about the feature co-occurrences. Then, using the 2-D histogram, the grouping algorithm partitions the clusters and estimates the number of clusters. The main operations of our approach consist of five steps: 1) extraction of texture features using the Gabor filters, 2) quantization and encoding of the features by the SOM algorithm, 3) computation of co-occurrence matrix of the encoded features, 4) estimation of the optimal configuration of the clusters, and 5) verification of the final result with a validity measure. The proposed scheme is very fast and accurate, as evidenced by the experimental results.

The organization of the rest of this paper is as follows: in section 2, we briefly review the extraction of texture features; in section 3, we present a method to estimate the number of texture segments; in section 4, we report experimental results; finally, in section 5, we summarize the works presented in this paper.

2. TEXTURE FEATURES

2.1. Extraction of Texture Features

The multi-channel Gabor filters have been known to be closely related to the receptive field profiles within the human visual system, and efficient for extracting texture features [6]. Gabor features are obtained by convolving an image with a set of Gabor elementary functions,

$$h(x, y) = g(x', y') \cdot \exp[\, 2\pi j(Ux + Vy)],$$

where $(x', y') = (x\cos\theta + y\sin\theta, -x\sin\theta + y\cos\theta)$ denote coordinates oriented at the angle θ from the x-axis in the spatial domain, and (U,V) the filter location in the frequency-domain. A 2-D Gaussian $g(x,y)$ is specified by center frequency $F = sqrt(U^2 + V^2)$ and orientation $\theta = tan^{-1}(U/V)$ with the following form,

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

where σ_x and σ_y are scale factors characterizing the spatial extent of the Gaussian. By convolving an image $I(x,y)$ with a set of Gabor filters and then by taking the energy, one can obtain the Gabor features, $\boldsymbol{f} = (f_1, f_2, \ldots f_d)$, where d is the number of Gabor filters with different frequency F and orientation θ.

2.2. Clustering and Encoding of Texture Features

Gabor features, as described in the previous section, are represented as d-dimensional vectors. Therefore, it is difficult to directly apply certain useful scalar-oriented tools, e.g., histogram-based techniques, for analyzing the features. To address this issue, we employed the SOM algorithm which projects high-dimensional vectors onto a lower-dimensional space, v: $R^d \rightarrow \Lambda$, where $v\,(f_1, f_2, \ldots f_d) = \lambda$ and λ is an index in a feature map Λ. This mapping quantizes features and encodes the similarity of features as the spatial proximity in a feature map. An *encoded-feature image* $J(x,y)$ is generated by assigning each feature $(f_1, f_2, \ldots f_d)$ in the input image $I(x,y)$ to the corresponding index $\lambda = v\,(f_1, f_2, \ldots f_d)$ of the feature map for all (x,y). The value of pixels in J is in the range of 0 to K-1, where K is the dimension of Λ. We note that, at this preprocessing step, K determines only the quantization resolution for the purpose of encoding, and has little to do with the optimal number of clusters. Considering the textural properties of natural images, K=32 appears to be adequate. As needed, a conservative estimation of K=64 should be sufficient for most images.

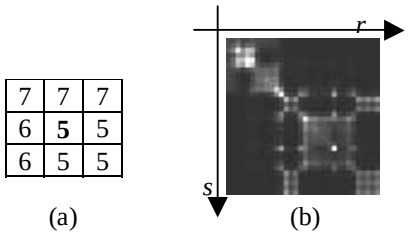


Figure 1. (a) Example of a 3×3 window, and (b) graphical representation of a 2-D histogram $h_{XY}(r,s)$ of size 64×64.

3. ESTIMATING THE NUMBER OF CLUSTERS

The essence of our approach is to capture the relationship of neighboring features, and then use this information to identify pixels of the same class. For the goal, a 2-D histogram called a co-occurrence matrix is first constructed with the encoded features, and then the grouping algorithm is applied to estimate the number of clusters.

3.1. Construction of Co-Occurrence Matrix

General co-occurrence matrix is a 2-D joint histogram of co-occurring pairs (r,s), where r and s are pixel values related by a displacement vector. We extend the basic notion of the co-occurrence matrix by replacing a replacement vector with a neighboring relation and by augmenting pair-wise relation to one-to-many relation. More specifically, given a 2-D histogram $h_{XY}(r,s)$, the row index $r \in X$ denotes any pixel values in J, and the column index $s \in Y$ denotes the pixel values within the neighborhood of pixels of value r. Here, both r and s denote the pixel values in J, and their notational difference reflects only the difference in the way of sampling. With fixed r_0 and s_0, the value of a histogram bin $h_{XY}(r_0,s_0)$ is computed as follows: for each pixel p with a value of r_0 in J, a neighborhood N_p of p consists of the pixels in an $n \times n$ window centered at p, excluding p. The number of pixels q with a value of s_0 within N_p is accumulated in the bin $h_{XY}(r_0,s_0)$. For example, if a configuration of $N_{p=5}$ in a 3×3 window is given as shown in Figure 1 (a), the values of $h_{XY}(5,5)$, $h_{XY}(5,6)$, and $h_{XY}(5,7)$ are incremented by 3, 2, and 3, respectively. The counting operation is performed for all pixels in J, and we obtain a 2-D histogram of size K×K. Figure 1 (b) shows an example of a 2-D histogram whose magnitudes are converted to 8-bit gray levels.

3.2. Estimation of the Number of Clusters

As aforementioned, if texture features are encoded using a feature map of dimension K, the 2-D histogram $h_{XY}(r,s)$

is constructed with the size of $K \times K$. Because the pixels in J takes a value from the index set $K = \{0, 1, \ldots, K\text{-}1\}$, the number of texture segments can be determined by the number of clusters in a partition on K. Constructing a partition of clusters on K is done by utilizing the co-occurring statistics of features, which is represented as the row and column index, i.e. r and s in $h_{XY}(r,s)$. The grouping algorithm takes K as the input, performs the following operations, and produces the output in a texture-class label C.

1. Initially, all indices in K are unlabeled, i.e. belong to the set $U = \{k: \forall k \in K\}$, and C is set to 1.

2. Among the unlabeled indices in U, select the one, say k_0, which occurs the greatest number of times in the encoded-feature image J, and set the class label of k_0 to C. Among the row indices r of $h_{XY}(r,s)$, find the one that corresponds to k_0 and denote it as r_C. Remove $k_0 = r_C$ from U to the set of the labeled indices, L.

3. From the column indices s of $h_{XY}(r,s)$, which are unlabeled, $s \in U$, find the ones that satisfy the criteria,

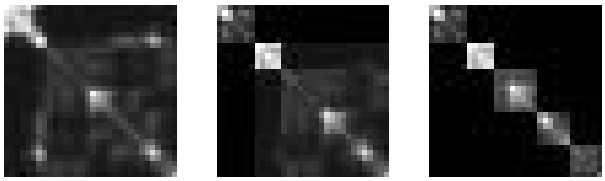
$$h_{XY}(r_C, s) \geq h_{XY}(r,s), \forall r \in U, r_C \neq r,$$

and assign C to all these s's. Thus, the selected indices are set to the same label as r_C that was processed at step 2. Remove the just selected indices from U and place them in L.

4. If there are unlabeled indices in U, then increase C by 1, and go to step 2 above. Otherwise, stop.

When the algorithm terminates, C attains the number of texture segments in the input image. The objective of step 2 is to select the index that plays the role of a seed for a new texture class. Generally this operation might not work for gray level-based approaches, because a gray level can be scattered over the entire image and hence the maximal occurrence of a gray level does not guarantee for those pixels to form a cluster. In our approach, however, the indices represent textural characteristics spanned over a region, and it does not matter whether or not the regions are scattered since scattered regions can form a cluster.

Once a seed index r_C for a new class is found at step 2, the indices that belong to the class are identified at step 3. The identification is carried out based on the idea that, by regional property, the indices that maximally co-occur with r_C will belong to the same class as r_C. The maximal co-occurrence can be checked by row-wise comparison for all unlabeled column indices $s \in U$ using the criteria $h_{XY}(r_C, s) \geq h_{XY}(r,s), \forall r \in U, r_C \neq r$.



(a) (b) (c)

Figure 2. Visualization of the dynamics of the algorithm; (a) A 2-D histogram of the image I_6 in Figure 4; (b) after the 1st iteration; (b) after the algorithm terminates.

This process can be graphically illustrated for visual understanding. At the first iteration, the identified indices, say $K_m = \{k_0, k_1, \ldots, k_{m-1}\}$, constitute an increasing, but not necessarily consecutive, sub-sequence of the entire sequence $S = <0, 1, \ldots, K\text{-}1>$. By the permutation, $P: k_i \leftrightarrow i$ for $0 \leq i \leq m\text{-}1$, those k's exchange their positions with the indices in the first m places in S, and become a consecutive sub-sequence. As a result, for all k_i and $k_j \in K_m$, the corresponding histogram bins permute their positions, $h_{XY}(k_i, k_j) \leftrightarrow h_{XY}(i,j)$, and form a squared region as shown at the left upper corner in Figure 2 (b) that is converted from Figure 2 (a). In this figure, the histogram bins representing texture boundaries, i.e. $h_{XY}(r,s)$ with $r \notin K_m$ or $s \notin K_m$, are removed for the clarity of illustration. The final result is illustrated in Figure 2 (c) where the number of texture segments, 5 in this case, is clearly shown by the five squares on the diagonal.

4. EXPERIMENTAL RESULTS

We evaluated the performance of the proposed algorithm with six synthetic texture images of size 256×256, I_1 to I_6, as shown in Figure 3. Texture features were computed by applying 24 Gabor filters with four different frequencies and six orientations, and encoded with the feature map of dimension K=32. For each image, evaluation was carried out for six 2-D histograms obtained by running $n \times n$ windows with odd n varying from 3 to 13. The intermediate results are summarized in Table 1. From these, we could correctly obtain the final result, i.e. the five texture segments, either by the majority rule or by applying a cluster validity measure (defined below) only for three cases, i.e., 4-, 5-, and 6-cluster partitions.

We also conducted experiments on exhaustive search-based approaches by applying the cluster validity measure defined in [3,4]. The measure produces an optimal solution at the minimum of the ratio $R=V/W$ of the intra-cluster scatter V to the inter–cluster separation W defined as,

$$V = \frac{1}{N}\sum_{i=1}^{K}\sum_{f \in Ci}\left\| f - z_i \right\|^2,$$

$$W = \min(\left\| z_i - z_j \right\|^2), 0 \le i, j \le K-1, i \ne j,$$

where N is the number of features, K the number of clusters, and z_i the center of cluster C_i. The experimental results are shown in Table 2 where the minimum is denoted in boldface and the second minimum in italic. As shown in the first column, the number of clusters is under-estimated as 3 for all the images, while 5 is correct. As noted by Ray and Turi [4], the measure's tendency for under-estimation suggests that the estimation of the number of clusters is not a trivial problem. As such, our simple but robust approach can be employed as an effective step for the sophisticated image analysis. Furthermore, computation in our approach can be done in a nearly constant time in the order of K, independent of the size N of the input image. Considering that $K<<N$, our approach saves a lot of computing time, compared with other exhaustive search-based approaches. More specifically, the average computing time on the 1.6GHz Pentium-IV PC shows the clear difference in two approaches. In our approach, the average processing for an image took 19 seconds (secs) with 18 secs for quantizing and encoding and less than 1 sec for estimating the number of clusters. In the exhaustive search-based approach, the search for the number of clusters was carried out for the 6 configurations, i.e. 3 to 8 clusters. The average processing time was 63 secs with 34 secs for clustering of the six configurations and 29 secs for determining the number of clusters. Even crude comparison of these results shows that our approach achieved 3 times of gain for the overall time and nearly 30 times of gain for the computation of the number of clusters.

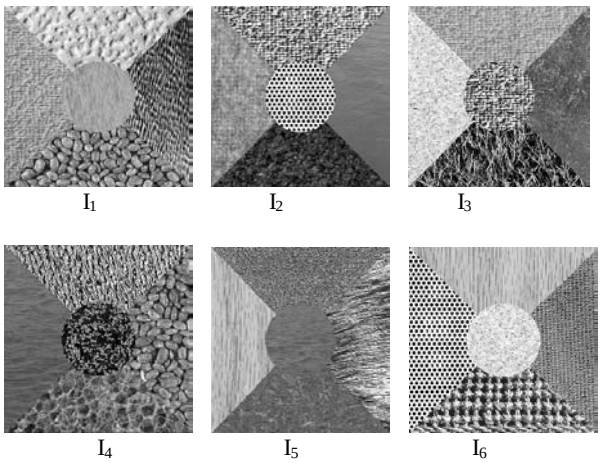


Figure 3. Five-texture images of size 256×256.

Table 1: Estimated number of texture segments in the five-texture images with varying window sizes

Image	Window size (n×n)					
	3×3	5×5	7×7	9×9	11×11	13×13
I_1	**5**	**5**	**5**	4	4	3
I_2	4	**5**	**5**	**5**	**5**	**5**
I_3	**5**	6	6	**5**	**5**	**5**
I_4	**5**	4	4	**5**	**5**	**5**
I_5	6	**5**	**5**	**5**	**5**	4
I_6	**5**	**5**	**5**	**5**	4	**5**

Table 2: Values of cluster validity measure R ($\times 10^{-3}$)

Image	Number of clusters (K)					
	3	4	5	6	7	8
I_1	**0.171**	0.645	*0.601*	1.435	3.226	11.868
I_2	**0.098**	10.101	*1.006*	2.054	27.038	9.878
I_3	**0.329**	1.726	*0.568*	6.853	6.853	5.126
I_4	**0.074**	1.166	*0.232*	1.303	1.845	13.250
I_5	**0.102**	1.535	*0.217*	2.512	1.770	2.913
I_6	**0.188**	1.584	*0.408*	1.190	10.566	7.360

5. CONCLUSIONS

In this paper, we presented a simple yet highly effective technique in determining the number of clusters, and verified its performance in estimating the number of texture segments. Unlike the traditional approaches, which perform an exhaustive search over all possible configurations, the proposed method directly estimates the number of clusters by using the co-occurrence statistics of the features. Experiments with texture images showed that considerable gains were achieved both for accuracy and for computing time.

6. REFERENCES

[1] D. A. Langan, J. W. Modestino, and J. Zhang, "Cluster validation for unsupervised stochastic model-based image segmentation," *IEEE Trans. Image Proc.*, Vol. 7, pp. 180-195, 1998.

[2] D. L. Davies and D. W. Bouldin, "A cluster separation measure," *IEEE Trans. Pattern Anal. Machine Intell.*, Vol. 1, pp. 224-227, 1979.

[3] J. P. Havelicek and P. C. Tay, "Determination of the number of texture segments using wavelets," *Proc. 16th Conf. on Applied Math.*, Univ. of Central Oklahoma, pp. 61-70, July 20, 2001.

[4] S. Ray and R. H. Turi, "Determination of number of clusters in k-means clustering and application in color image segmentation," *Proc. 4th Conf. on Pattern Recognition and Digital Tech.*, Calcutta, India, pp. 137-143, Dec 27-29, 1999.

[5] T. Kohonen, *Self-Organizing Maps*, Berlin: Springer, 1995.

[6] D. F. Dunn, W. E. Higgins, and J. Wakeley, "Texture segmentation using 2-D Gabor elementary functions," *IEEE Trans. Pattern Anal. Machine Intell.* ,Vol.16, pp.130-149, 1994.