International Journal of Pattern Recognition and Artificial Intelligence
Vol. 15, No. 8 (2001) 1231–1250

UNSUPERVISED TEXTURE SEGMENTATION BY DETERMINING THE INTERIOR OF TEXTURE REGIONS BASED ON WAVELET TRANSFORM*

KUEN-LONG LEE and LING-HWEI CHEN†

Department of Computer and Information Science, National Chiao Tung University, 1001 Ta Hsueh Road, Hsinchu, Taiwan 30050, R.O.C.
†*lhchen@cc.nctu.edu.tw*

Traditional approaches for texture segmentation via wavelet transform usually adopt textural features to achieve segmentation purposes. However, for a natural image, the characteristics of the pixels in a texture region are not similar everywhere from a global viewpoint, and over-segmentation often occurs. To deal with this issue, an unsupervised texture segmentation method based on determining the interior of texture regions is proposed. The key idea of the proposed method is that if the pixels of the input image can be classified into interior pixels (pixels within a texture region) and boundary ones, then the segmentation can be achieved by applying region growing on the interior pixels and reclassifying boundary pixels. Based on the fact that each pixel P within a texture region will have similar characteristics with its neighbors, after applying wavelet transform, pixel P will have similar response with its neighbors in each transformed subimage. Thus, by applying a multilevel thresholding technique to segment each subimage into several regions, pixel P and its neighbors will be assigned to the same region in most subimages. Based on these segmented results, an interior pixels finding algorithm is then provided to find all interior pixels of textural regions. The algorithm considers a pixel which is in the same region as its neighbors in most subimages as an interior pixel. The effectiveness of this method is proved by successfully segmenting natural texture images and comparing with other methods.

Keywords: Texture segmentation; wavelet transform; multiresolution segmentation; unsupervised clustering.

1. Introduction

Texture segmentation has long been an important topic in image processing. Basically, it aims at segmenting a textured image into several regions with the same texture features. An effective and efficient texture segmentation method will be very useful in applications like the analysis of aerial images, biomedical images and seismic images as well as the automation of industrial applications. Like the

*This research was supported in part by the National Science Council of R.O.C. under contract NSC-89-2213-E-009-102.
†Author for correspondence.

other segmentation problems, the segmentation of textures requires the identification of proper texture-specific features with good discriminative power. Generally speaking, texture feature extraction methods can be classified into three major categories, namely, statistical, structural and spectral. In statistical approaches,[2,8,9] texture statistics such as the moments of the gray-level histogram, or statistics based on gray-level co-occurrence matrix are computed to discriminate different textures. For structural approaches,[16,20] "texture primitive", the basic element of texture, is used to form more complex texture patterns by grammar rules which specify the generation of texture patterns. Finally, in spectral approaches,[1] the textured image is transformed into frequency domain. Then, the extraction of texture features can be done by analyzing the power spectrum. Various texture descriptors have been proposed in the past. In addition to the aforementioned methods, Law's texture energy measures,[12] Markov random field models,[13] texture spectrum,[10] etc. are some other texture descriptors.

Recently, multichannel and multiresolution-based approaches have drawn a lot of attention in the field of texture analysis. Several successful applications of these approaches to unsupervised texture segmentation have been reported. Multichannel-based approaches[3,11,19] use a bank of preselected Gabor filters in terms of frequencies, orientations and bandwidths to filter an input image. The features extracted from the responses of the filtered images are then used for texture classification or segmentation. Wavelet transform is used in multiresolution-based approaches. Most of the wavelet-based methods[14,15,18] use a pyramidal type of decomposition to transform the input image into an image of wavelet coefficient at different resolutions. The wavelet coefficients are then transformed into texture-specific features. Based on the features, traditional clustering techniques such as c-means[7] are adopted to segment the image into texture regions. Then, the segmentation results under different resolutions are further integrated to produce the final segmentation result. For a natural texture, however, the characteristics of the pixels in the texture pattern are not similar everywhere from a global viewpoint. Thus, without using a good integration technique for the feature images resulting from wavelet transform, over-segmentation often occurs for these methods.

To circumvent the above-mentioned issue, in this paper, based on the fact that each interior point in a texture region must possess similar properties with its neighbors, a new wavelet-based texture segmentation method is proposed. The key idea is that if the pixels of the input image can be classified into interior pixels and boundary ones, the interior pixels stand for the interior parts of texture regions, then the segmentation can be achieved by applying region growing on the interior pixels. To implement this idea, a wavelet transform is first applied to get several subimages with different frequencies and orientations. Then, in each subimage, the wavelet coefficients are thresholded to get preliminary segmented results. As each pixel within a texture region will have similar response with its neighbors in the transformed images, it will be assigned to the same region as its neighbors in most subimages. An interior pixels finding algorithm is then provided to integrate the

segmented results of the subimages to separate texture interiors from their boundaries. Texture regions can therefore be extracted by region growing. Finally, the texture boundaries are reclassified and segmentation is achieved.

In Sec. 2, the proposed method is described in detail. Experimental results and discussion are presented in Sec. 3. Finally, in Sec. 4, we provide a conclusion.

2. The Proposed Method

The system block of the proposed method is shown as Fig. 1. Through packet-structured wavelet transform, a textured image is first decomposed into subimages with different frequency channels and orientations. Then, each subimage is smoothed to increase the homogeneity of each texture region. A multilevel thresholding procedure is then used to segment each subimage into several regions. Note that those pixels assigned to the same region will have similar responses in a transformed subimage. Some small fragmented regions might be produced after the thresholding. To eliminate these fragmented regions, a region editing method is applied to merge these regions with their most similar neighboring regions. Based on the segmented results and the fact that pixels within a texture region will have the same properties as its neighbors, an interior pixels finding algorithm is provided to separate the interior of texture regions from their boundary. Region growing is applied and boundary pixels are further classified to produce the final segmentation result. In the following, the proposed method will be described in detail.

2.1. *Wavelet decomposition*

Traditionally, the pyramid-structured wavelet transform[17] is used in texture classification and segmentation in terms of decomposition scheme. When decomposing the image from level to level, the LL subimage is always decomposed and the rest of the subimages are left intact. For natural textures, however, useful texture information might reside in LH, HL and HH subimages. Thus, the tree-structured wavelet decomposition scheme[5] is proposed. Packet-structured wavelet decomposition[17] is another scheme which provides the possibility of exploring the useful information contained in every subimage, but the number of processed subimages is increased as well. Thus, to adopt packet-structured decomposition scheme, the issue of how to integrate texture regions from different subimages needs to be dealt with. In our method, an interior pixels finding algorithm is provided to integrate the segmented results of different subimages. The packet-structured wavelet transform is first conducted and an input image is decomposed to two levels by using the 6-tap Daubechies[6] wavelet as the wavelet filter. An example is shown in Fig. 2.

2.2. *Subimage smoothing*

After conducting the wavelet transform, an input image is decomposed into subimages with different frequencies and orientations, and the wavelet coefficients are

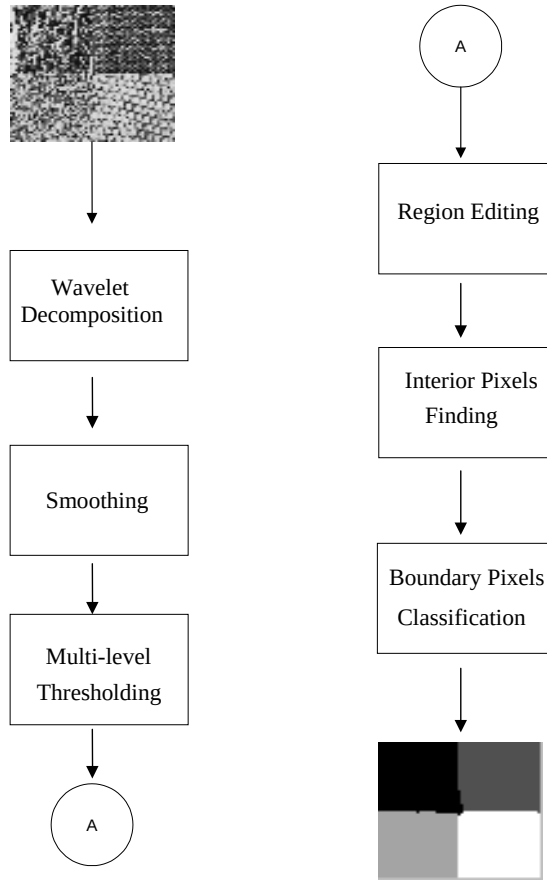
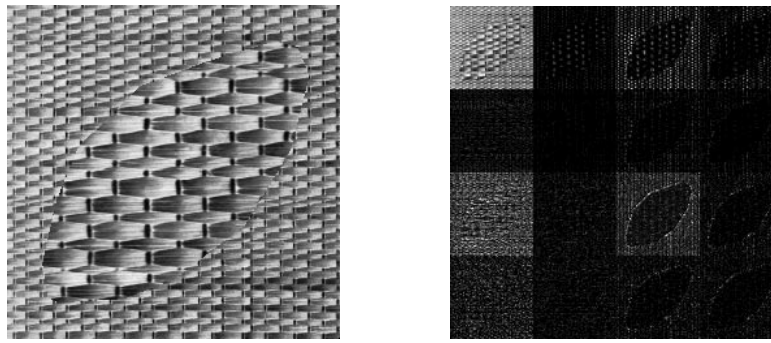


Fig. 1. The system block of the proposed method.



(a) (b)

Fig. 2. An input image and its transformed image via packet-structured wavelet transform. (a) A textured image with two texture regions. (b) The transformed image of (a) by packet-structured wavelet transform.

used as the base for segmentation. Since, irregularity is present inside the texture regions, to facilitate the segmentation, a smoothing operation is then applied on the wavelet coefficients to increase the homogeneity of each texture region. The smoothing operation is conducted by a Gaussian-like smoothing filter:

$$f(x,y) = \frac{1}{S} \sum_{(a,b)\in W_{x,y}} |g(a,b)| \times w(a,b) \tag{1}$$

where $f(x,y)$ denotes the smoothed coefficient at point (x,y), $g(a,b)$ is the original wavelet coefficient at point (a,b), and $W_{x,y}$ is an $n \times n$ window centered at point (x,y), $w(a,b)$ is the weighting function and S is the sum of all weightings within $W_{x,y}$. In our experiments, n is set as 7 and the corresponding weighting matrix is set as follows:

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 2 & 2 & 2 & 2 & 1 \\ 1 & 2 & 3 & 3 & 3 & 2 & 1 \\ 1 & 2 & 3 & 4 & 3 & 2 & 1 \\ 1 & 2 & 3 & 3 & 3 & 2 & 1 \\ 1 & 2 & 2 & 2 & 2 & 2 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Figure 3 shows the result of applying the smoothing operation to Fig. 2(b).

2.3. *Multilevel thresholding*

After performing smoothing on the wavelet coefficients of each subimage, our goal in this step is to divide each subimage into several texture regions, pixels in the same region will have similar transformed values. To achieve this, a multilevel thresholding method is provided.

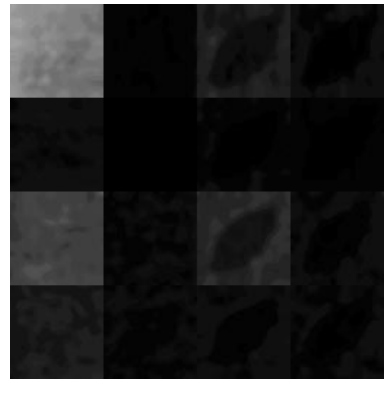
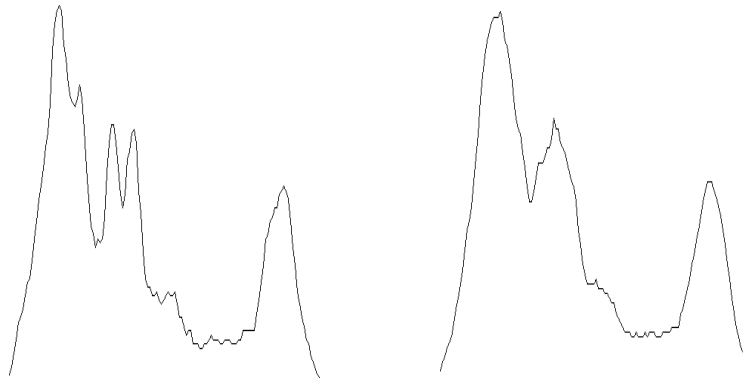


Fig. 3. The smoothed image of Fig. 2(b).



(a) (b)

Fig. 4. An example to explain the necessity of histogram smoothing. (a) A histogram with many granules. (b) Smoothed result.

Observing a smoothed subimage, we can find that the gray values of pixels in the same texture region are similar, thus they will be grouped as mountain-like patterns in the subimage histogram. On the other hand, the boundary pixels between two texture regions correspond to the histogram valleys. Thus, we can segment texture regions by locating the histogram valleys. In addition, as the subimage of the lowest frequency band (LL) corresponds to the blurred and subsampled version of the original image, and the other subimages correspond to the edge images of different orientations, we will develop two different valley locating methods, one is for LL subimage, and the other for LH, HL and HH subimages.

To locate the valleys, the histogram of a subimage is first constructed. For a natural texture image, the histogram usually has many granules [see Fig. 4(a)]. This will make the valley finding difficult. To reduce this granular phenomenon, we will provide two techniques. One is histogram smoothing, the other will be described later. The histogram smoothing is performed using the following formula:

$$\hat{h}(i) = \left(h(i) + \sum_{j=1}^{n} h(i-j) + h(i+j) \right) \Big/ (2n+1)$$

where $\hat{h}(i)$ and $h(i)$ are the smoothed and the original histogram frequencies of gray-level i, respectively. As human visual system cannot discriminate two gray levels with difference less than 10, the range of histogram smoothing is set to 11 and n is set as 5 accordingly. An example of the smoothed result is shown in Fig. 4(b).

As mentioned earlier, different valley location approaches are applied for subimage LL and the other subimages. After applying smoothing, the histogram valleys for LH, HL and HH subimages are located by using the following criterion:

Criterion C_1:

If $h(i-1) > h(i)$ and $h(i) < h(i+j)$, with $h(i) = h(i+k)$, $0 < k < j$, then let $i' = i + (j-1)/2$, and consider gray-level i' as a valley.

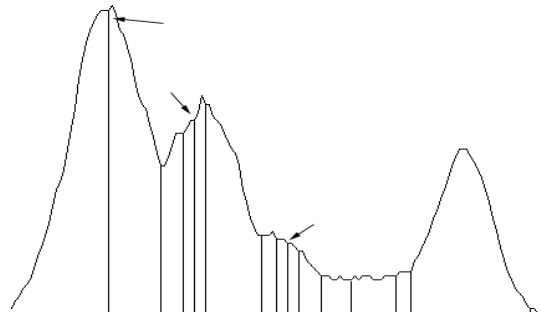


Fig. 5. An example with several false valleys marked.

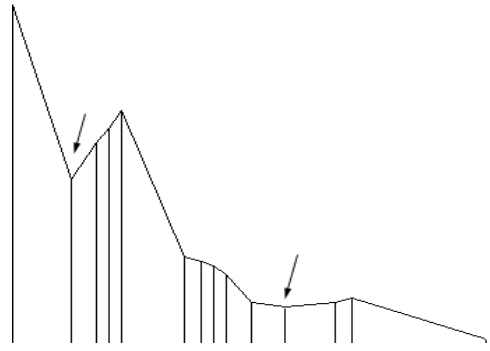


Fig. 6. The HV curve of Fig. 5 with two remaining valleys marked.

Based on C_1, the image histogram is scanned, the valley set V, $V = \{v_i, i = 1, 2, \ldots, m\}$, is then constructed for each subimage. For each subimage LH, HL and HH, V is used for thresholding. For subimage LL, however, we observe that some valleys in V, which are local minimum defined by C_1, actually are of high frequency in terms of histogram value. These valleys are considered as false ones. An example of false valleys is shown in Fig. 5. If we connect all valleys from left to right to form a curve HV (see Fig. 6), we can find that those false valleys mentioned above will reside on the mountain-like patterns of HV. By only locating the valleys of HV, we can eliminate those false valleys. In addition, let the valleys of HV be denoted as V_{HV}. We sometimes find some **valley groups** in V_{HV}, each group, $vs, vs = \{v_l, v_{l+1}, \ldots, v_m\}$, $l < m$, has the property $|v_k - v_{k+1}| < 10, \forall\, k, l < k < m$ or the number of pixels bounded by vs is less than a minimum number of pixels, $O_{\min}$, a reasonable region should have. As human visual system cannot tell the difference between two gray levels with difference less than 10, the valleys in vs should be merged. Besides, if the size of the region bounded by vs are less than $O_{\min}$, then vs needs to be merged as well. To merge a valley group, the valley with minimum histogram value is selected as threshold for that group. In summary, the aforementioned points can be implemented by the following algorithm.

Multilevel thresholding algorithm

Let $V = \{v_i, i = 1, 2, \ldots, m\}$ denote the valley set of subimage LL.

Build HV curve from V.

Find valleys set V_{HV} from HV.

While not end of V_{HV}

Scan V_{HV} to find the next valley group vs.

Choose case

Case 1: There is only one element in vs, then the element is considered as a threshold.

Case 2: There are more than one element in vs, locate the valley in vs with minimum histogram value as a threshold.

End choose case

Advance to the first valley next to the last vs found.

End

To extract texture regions in an image, the valley positions which separate texture regions are to be determined. With those determined valleys, pixels with gray levels between two valleys are considered to form a certain region, and all texture regions can be extracted accordingly. In our experiment, $O_{\min}$ is set as 5% of the subimage size. Once completing this procedure, we can use the final valley set to threshold each subimage. The spatial location of a region can be easily obtained by applying region growing on the thresholded subimage. However, some fragmented regions might still be present in the textured image. An example is shown in Fig. 7. To eliminate these fragmented regions, an automatic region editing technique is provided to do this job. On the other hand, some subimages might have

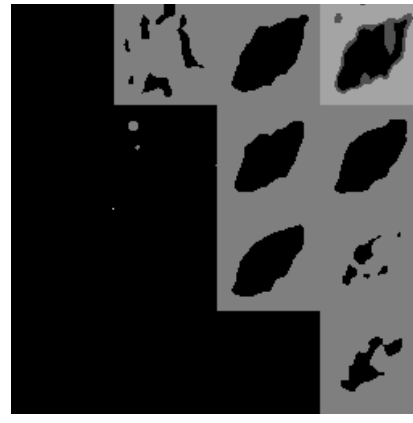


Fig. 7. Thresholded subimages of Fig. 3 with some fragmented regions.

only one homogeneous region after thresholding, these subimages do not contain useful information for segmentation purposes. Therefore, they are excluded from later steps to speed up the computation.

2.4. *Region editing*

As mentioned above, the fragmented regions are present after thresholding and have to be combined with their neighboring regions to achieve a better segmentation result. To achieve this, small regions with size less than $O_{\min}$ are located and merged with the most similar neighbor region in terms of gray value. Basically, the region editing process can be described as follows:

Region Editing Procedure:

Let s_i be the subimage considered.

(1) Perform a region growing procedure on s_i to get regions o_i, $i = 1, 2, \ldots, k$, and label these o_i by $g_i = \frac{(i-1)L}{k}$, $i = 1, 2, \ldots, k$, where L is the maximum of gray levels in the subimage.
(2) Compute the size A_i of each o_i, and locate a region R, whose size is less than $O_{\min}$. If no such region exists, then terminate. Otherwise, continue.
(3) Let the neighboring regions of R be denoted as $r_i, i = 1, 2, \ldots, n$.
Take r_j with

$$j = \arg\min_i |g_i - g_R| .$$

(4) Merge R with r_j. And iterate from Step (2).

Figure 8 shows the resulting image after applying region editing on Fig. 7. Fragmented regions are merged with their appropriate neighboring regions, the resulting regions provide a better base for segmentation.

Fig. 8. The resulting image of applying region editing on Fig. 7.

2.5. *Interior pixels finding*

Now we have done segmentation for each subimage. As each subimage corresponds to a certain frequency channel, it might have a good segmentation result for some textures, but not for all textures. Thus, the problem becomes how to integrate the segmentation results from different subimages.

To do the segmentation integration, the fact that each pixel within a texture region must have similar property as its neighbors is applied. To be more specific, a pixel within a texture region will have similar property with its neighbors in the transformed image, thus it can be considered as an interior pixel in the transformed image. For example, a pixel A within a pure vertically arranged texture will have a strong response in the HL subimage, but weak response in the LH subimage at a certain decomposition level, so will its neighbors. Therefore, the information in both HL and LH subimages support pixel A to be an interior point, but that is not the case for boundary pixels. As for a boundary pixel, it will not necessarily have similar property as its neighbors. Based on this fact, for each pixel $x(i, j)$ in each subimage s_k, we examine its neighbors. If for most of the subimages, most of the neighbor pixels of $x(i, j)$ are in the same region as $x(i, j)$, then $x(i, j)$ is deemed as the interior point of a texture region. In this way, we can tell whether a pixel is within the interior of a texture region or not, and separate the texture interior from its boundary. Now we will describe the interior pixels finding algorithm in more detail.

Algorithm: Interior Pixels Finding

For each subimage s_k do
Begin
  For each pixel $x(i, j)$ in s_k do
  Begin
    Check the $M \times M$ neighborhood $N(i, j)$ of $x(i, j)$
    If more than $p\%$ pixels of $N(i, j)$ are in the same region as $x(i, j)$ then
    $c(i, j) = c(i, j) + 1$
  End
End
For each counter $c(i, j)$ do
Begin
  If $c(i, j) > T_2$ then
    $CM(i, j) = 1$
  Otherwise
    $CM(i, j) = 0$
  End if
End

In our experiments, M is set as 7 and p is set as 80. Based on the above-mentioned fact, we can see that after applying the above algorithm, those interior pixels of texture regions will have high $c(i,j)$, and those pixels with low $c(i,j)$ will be boundary pixels. Therefore, we can separate the interior from texture boundary by thresholding on $c(i,j)$.

After applying the algorithm, pixels with value 1 in CM matrix represent those interior pixels of texture regions. On the other hand, pixels with value 0 represent boundary between texture regions. Therefore, we can perform a region growing on CM matrix to create an initial segmentation. Boundary pixels will be further classified in a later step.

One important issue in the above algorithm is how to determine the T_2 automatically. Note that $c(i,j)$ stands for the number of subimages supporting the pixel (i,j) to be an interior point of a certain texture region. Assume that the number of subimages used in the algorithm is K, then the possible value of T_2 is from 1 to K. To determine the best threshold for T_2, a two-step method is provided. The main idea is to test each candidate of T_2 and determine the best threshold based on the number of regions created as well as the separability among these regions. Considering these threshold candidates, the most frequent resulting region number is determined first. Then, the threshold resulting in this region number is chosen as the tentative candidate for T_2. If there are more than one most frequent resulting region numbers, then the threshold that results in larger class separability is selected as the final threshold. Before describing the algorithm for automatic threshold selection, some definitions will be given first.

Define the class separability, SP as the ratio of intra-distance, d_1 and inter-distance, d_2 of the texture regions, TR_j, $j = 1, 2, \ldots, m$. Let M_j denote the mean gray value of TR_j, and $p_j(x,y)$ be the value of the pixel (x,y) in TR_j. Then, d_1 and d_2 are defined as follows respectively.

$$d_1 = \frac{1}{m} \sum_{j=1}^{m} \frac{1}{A_j} \sum_{(x,y) \in TR_j} (p_j(x,y) - M_j)^2$$

and

$$d_2 = \frac{2}{m(m-1)} \sum_{k=1}^{m} \sum_{\substack{j=1 \\ j \neq k}}^{m} \frac{1}{A_j} \sum_{(x,y) \in TR_j} (p_j(x,y) - M_k)^2$$

where A_j is the area size of TR_j. And SP is defined as

$$SP = d_2/d_1 \,.$$

Based on these definitions, we will describe the threshold selection algorithm as follows:

Algorithm: Automatic Threshold Selection for Determining Texture Interior

Let $i = 1, 2, \ldots, K$ be the candidates of T_2.

For each i do
Begin
    Use i as the threshold to create CM_i.
    Perform a region growing on CM_i and produce texture regions,
    $TR_{ij},\ j = 1, 2, \ldots, m_i$.
    Compute the separability SP_i.
    $RC(m_i) = RC(m_i) + 1$.
End
$m = \max_j RC(j)$.
$MaxSP = 0$.
For each i do
Begin
  If $m_i = m$ then
    If $SP_i > MaxSP$ then
      $ThresholdCandidate = i$.
      $MaxSP = SP_i$.
    End if
  End if
End
$T_2 = ThresholdCandidate$.

Through the above algorithm, we can easily separate the interior of texture region from its boundary. The texture regions can then be extracted by performing a region growing on the CM matrix obtained by using the best threshold. Figure 9 shows the segmentation result of Fig. 8. Now we need to classify the boundary pixels to finish the segmentation. That is covered in the next section.

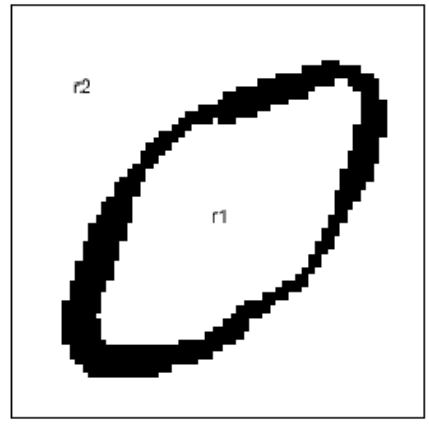


Fig. 9. The initial segmentation of Fig. 8 after integrating segmented subimages, r_1 and r_2 represent two regions, and black points stand for boundary.

2.6. *Boundary pixel classification*

After integrating the segmentation result in each subimage, we have separated the interior of texture regions from their boundaries, then the texture regions, TRS, are obtained by region growing. The boundary pixels can be classified based on the extracted texture regions. Given a boundary pixel p and a subimage s, locate the texture region r_p in s in which p resides, then determine the texture region in TRS which overlaps most with r_p. After checking all subimages, p is assigned to the texture region in TRS to which it is assigned most frequently. We describe this algorithm as follows:

Algorithm: Boundary Pixel Classification
For each boundary pixel p do
Begin
    For each subimage s_k do
    Begin
        Determine the region, r_k in s_k, to which p belongs based on the clusters of s_k
        Determine the region r_m of TRS, which overlaps with r_k most
        $c_m = c_m + 1$
    End
    $j = \arg\max_i c_i$
    Assign p to region r_j
End

After classifying the boundary pixels, we can obtain the final segmentation. Figure 10 shows the final segmentation result of Fig. 2(a), two texture regions are found.

Fig. 10. The final segmentation result of Fig. 2(a).

3. Experimental Results

Our method has been tested on several textured images from Brodatz album[4] and a natural image. All images used have 256×256 pixels and are gray-scale ones. Packet-structured wavelet transform is used up to two levels in all experiments. Besides, for comparison purposes, the same set of images are tested using the methods proposed in Refs. 14 and 15 as well. Figure 11 shows the segmentation results for an image which consists of three texture regions. The processing result of each step of our method is shown by sequence. Figure 11(d) shows the multilevel thresholded image with some homogeneous subimages. As these homogeneous subimages do not provide useful information for segmentation, they are excluded from later processing steps to accelerate the computation. Figure 11(e) shows the image after region

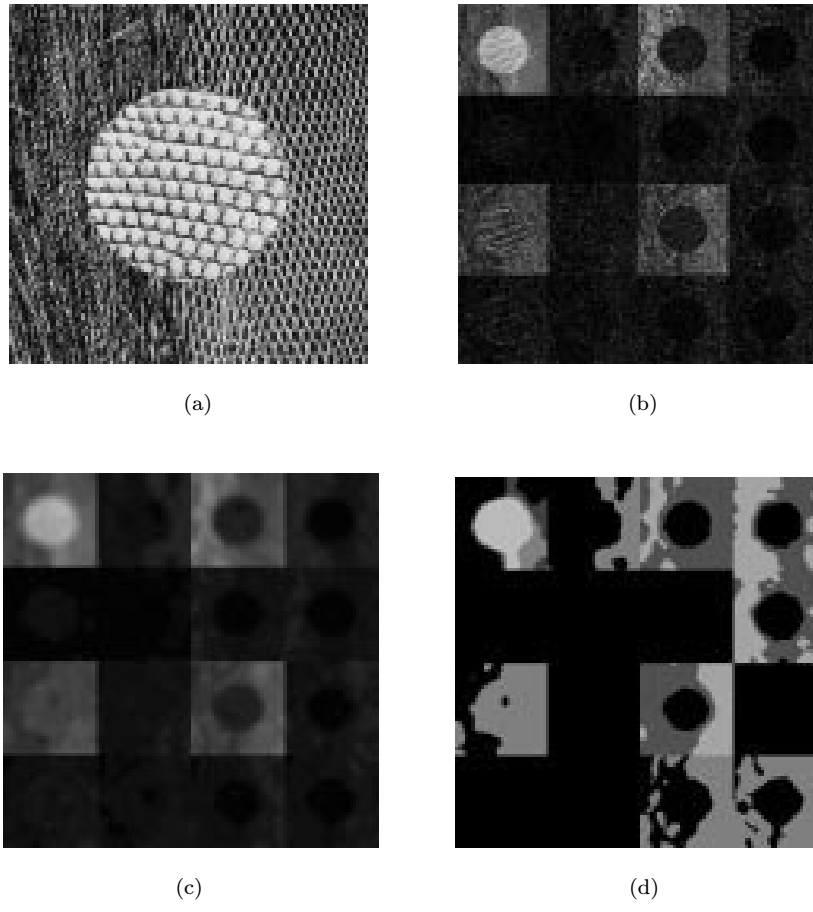
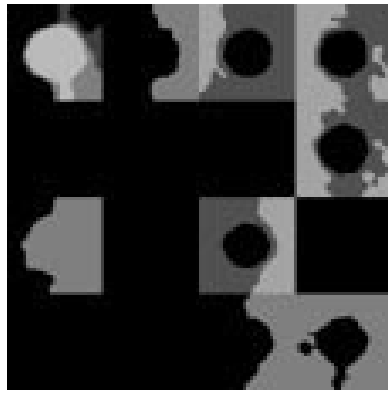
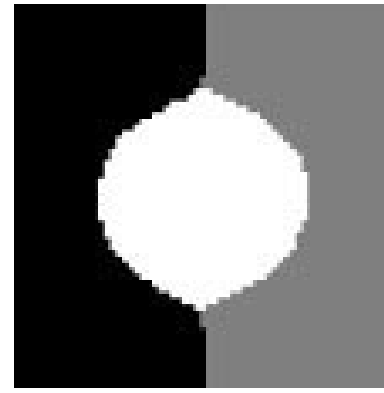


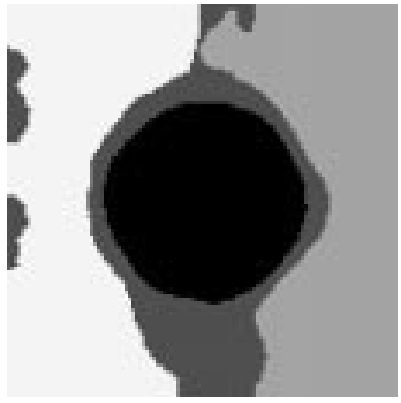
Fig. 11. Segmentation results of the proposed method and methods in Refs. 14 and 15 for three-category Brodatz textures. (a) An original image with three textures from Brodatz album. (b) The wavelet transformed result of (a). (c) The smoothed result of (b). (d) The multilevel thresholded result of (c).



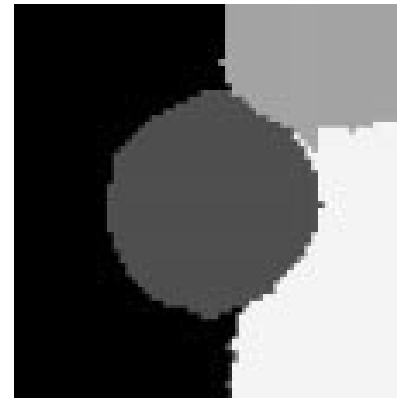
(e)



(f)



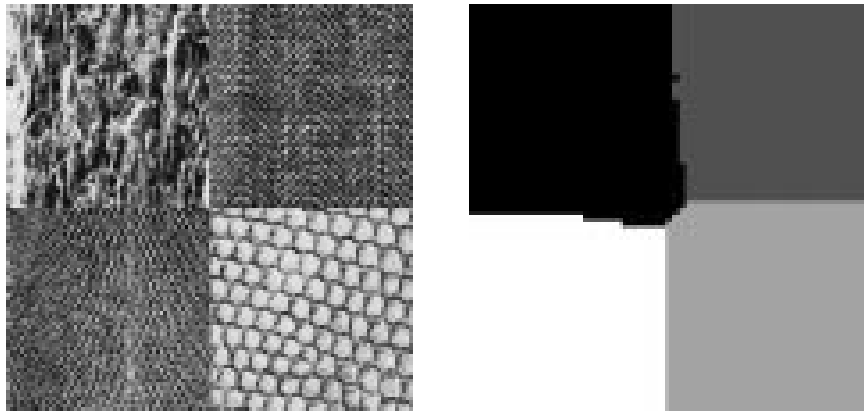
(g)



(h)

Fig. 11 (*Continued*). (e) The region edited result of (d). (f) The segmented result of the proposed method. (g) The result of applying the method proposed in Ref. 15. (h) The result of applying the method proposed in Ref. 14.

editing step, which eliminates some small regions in Fig. 11(d). It is noticed that some texture regions are segmented successfully in some subimages. But, it is hard to find a single subimage with satisfactory segmentation result. By integrating the segmentation results from subimages via finding the interior pixels, three textures can finally be obtained. As seen from the segmentation result in Fig. 11(f), three textures are successfully segmented, and the result is quite accurate. Figure 11(g) shows the result of applying the method proposed in Ref. 15, some pixels of the straw texture are misclassified and produce small regions. Besides, boundary pixels of the three textures are either misclassified, or segmented into another region. On the other hand, Fig. 11(h) shows the over-segmented result of Ref. 14, which segments Fig. 11(a) into four regions.



(a) (b)

(c) (d)

Fig. 12. Segmentation results of the proposed method and methods in Refs. 14 and 15 for four-category Brodatz textures. (a) An original image with four Brodatz textures. (b) The segmentation result of the proposed method. (c) The result of applying the method proposed in Ref. 15. (d) The result of applying the method proposed in Ref. 14.

Figure 12(a) shows another textured image obtained by collaging four Brodatz textures. Figure 12(b) shows the segmentation result which successfully separates these four texture patterns. Figure 12(c) shows the result of Ref. 15, the pixels of the boundary areas between texture patterns are misclassified. The result of Ref. 14 is shown in Fig. 12(d), it is roughly comparable to that of the proposed method.

Figure 13(a) shows a five-category textured image. The left and right patterns are very similar in terms of intensity and texture patterns. Figure 13(b) shows the segmentation result of the proposed method and the result is satisfactory. Figure 13(c) shows the result of Ref. 15, which wrongly classifies the boundary

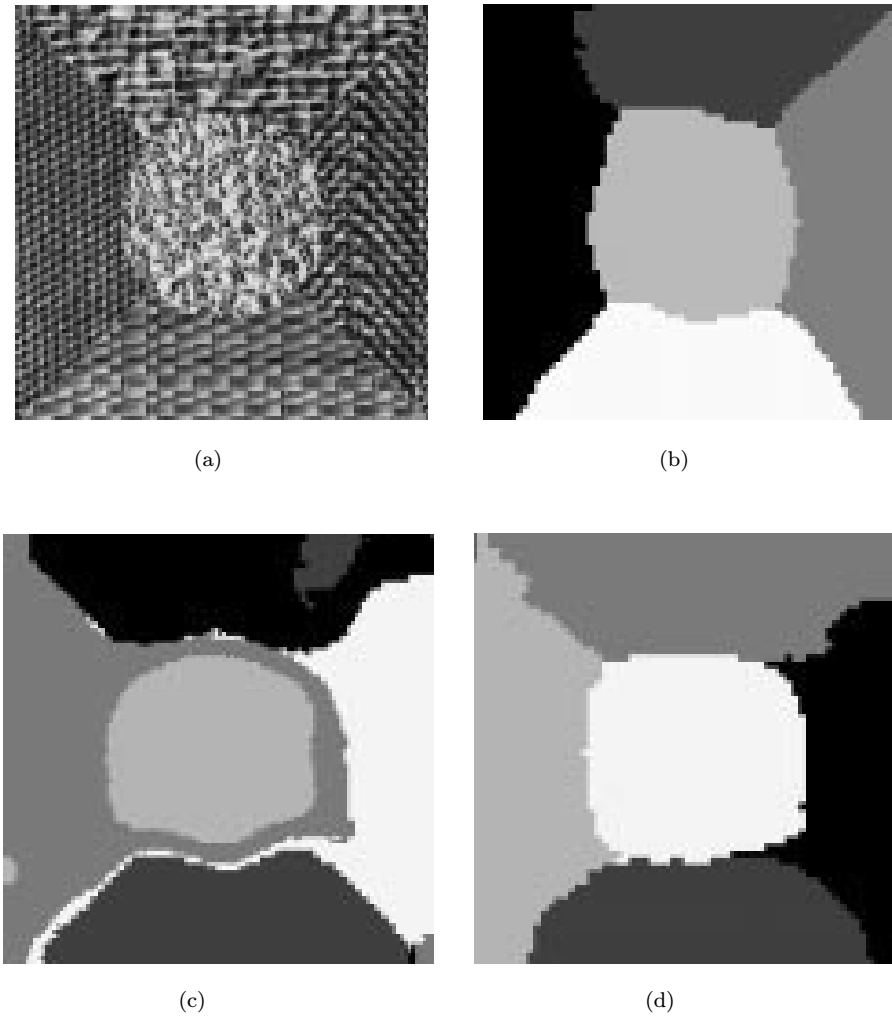
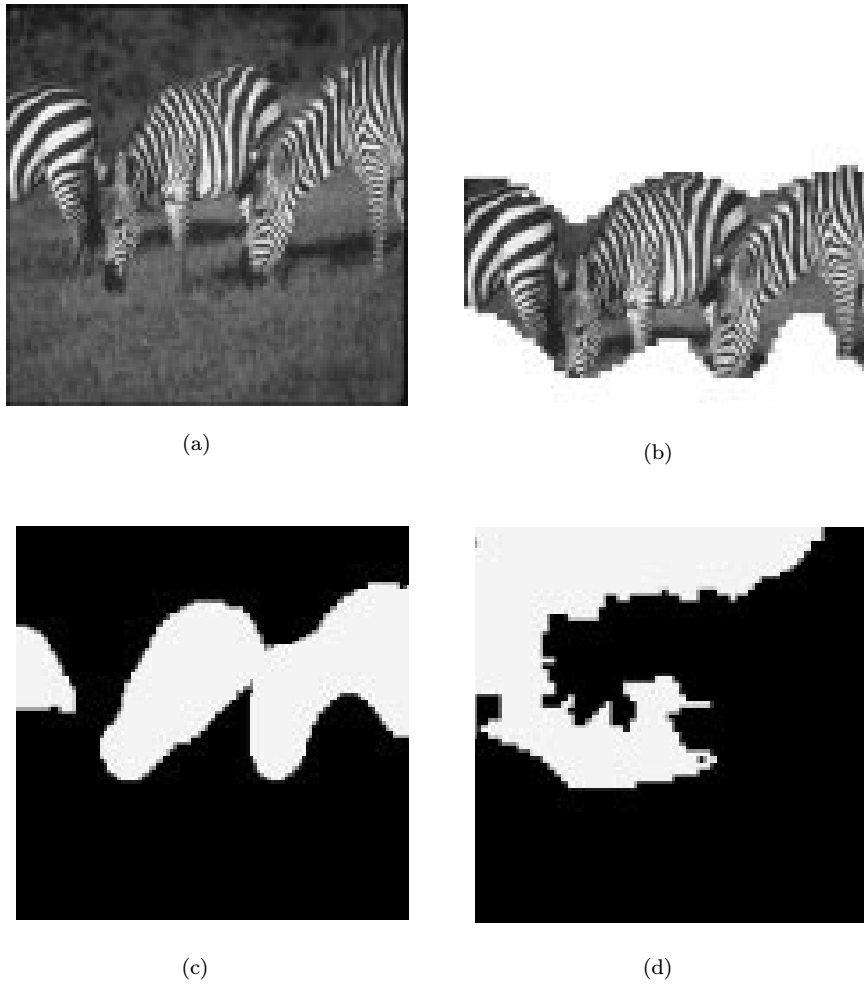


Fig. 13. Segmentation results of the proposed method and methods in Refs. 14 and 15 for five-category Brodatz textures. (a) An original image with five textures. (b) The segmented result of the proposed method. (c) The result of applying the method proposed in Ref. 15. (d) The result of applying the method proposed in Ref. 14.

pixels and makes the boundary pixels merge with the texture pattern on the left. Figure 13(d) shows the result obtained by in Ref. 14, it is roughly comparable to that of Fig. 13(b).

Figure 14(a) shows a natural image containing three zebras. The position of zebras is correctly extracted by the proposed method and shown in Fig. 14(b). Figure 14(c) shows the result obtained by applying the method proposed in Ref. 15, which cannot correctly extract the position of the three zebras. The result obtained in Ref. 14 is shown in Fig. 14(d), the zebras are not segmented out successfully.



(a) (b)

(c) (d)

Fig. 14. Segmentation results of the proposed method and methods in Refs. 14 and 15 for a natural image with three zebras. (a) A natural image with three zebras. (b) The extracted zebra position by applying the proposed method. (c) The result of applying the method proposed in Ref. 15. (d) The result of applying the method proposed in Ref. 14.

4. Conclusion

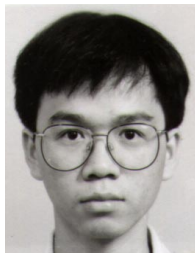
In this paper, based on the fact that an interior point in a texture region has similar properties as its neighbors, a new texture segmentation method has been presented. It can discriminate the interior parts of texture regions from boundaries. Natural textured images have been used to prove the effectiveness of the proposed method. Furthermore, some comparisons among the proposed method and two existing ones are made to show that our method is better than those two methods. The proposed texture segmentation method can be combined with other texture classification method to do texture image database retrieval.

Acknowledgments

The authors would like to express their appreciation to Dr. C. S. Lu who provided the results of his methods for comparisons with the proposed method, this greatly enriches the experimental results of this paper.

References

1. R. Bajcsy, "Computer description of texture surface," *Proc. 1973 Int. Conf. Artificial Intelligence*, Stanford, California, 1973, pp. 572–579.
2. R. Bajcsy and L. Lieberman, "Texture gradient as a depth cue," *Comput. Graph. Imag. Proc.* **5**, 1 (1976) 52–67.
3. A. C. Bovik, M. Clark and W. S. Geisler, "Multichannel texture analysis using localized spatial filters," *IEEE Trans. Patt. Anal. Mach. Intell.* **12**, 1 (1990) 55–73.
4. P. Brodatz, *Textures — A Photographic for Artists and Designers*, Dover, NY, 1966.
5. T. Chang and C. C. Kuo, "Texture analysis and classification with tree-structured wavelet transform," *IEEE Trans. Imag. Process.* **2** (1993) 429–441.
6. I. Daubechies, *Ten Lectures on Wavelets*, SIAM, Philadelphia, Pennsylvania, 1992.
7. P. A. Devijver and J. Kittler, *Pattern Recognition, A Statistical Approach*, Prentice Hall, 1982.
8. R. M. Haralick *et al.*, "Texture features for image classification," *IEEE Trans. Syst. Man Cybern.* **SMC-3**, 6 (1973) 610–621.
9. R. M. Haralick, *Statistical and Structural Approaches to Texture, Proc. 4th Int. Joint Conf. Pattern Recognition*, 1979, pp. 45–60.
10. D. C. He and L. Wang, "Textural filters based on the texture spectrum," *Patt. Recogn.* **24**, 12 (1991) 1187–1195.
11. A. K. Jain and F. Farrokhnia, "Unsupervised texture segmentation using Gabor filters," *Patt. Recogn.* **24**, 12 (1991) 1167–1186.
12. K. L. Laws, "Rapid texture identification," *Proc. SPIE* **238** (1980) 376–380.
13. S. Z. Li, *Markov Random Field Modeling in Computer Vision*, Springer-Verlag, NY, 1995.
14. C. S. Lu and P. C. Chung, "Wold features for unsupervised texture segmentation," *14th Int. Conf. Pattern Recognition* **II**, Special Topic in Computer Vision, Australia, 1988, pp. 1689–1693.
15. C. S. Lu, P. C. Chung and C. F. Chen, "Unsupervised texture segmentation via wavelet transform," *Patt. Recogn.* **30**, 5 (1997) 729–742.
16. S. Y. Lu and K. S. Fu, "A syntactic approach to texture analysis," *Comput. Graph. Imag. Proc.* **7**, 3 (1978) 303–330.
17. S. G. Mallat, "A theory of multiresolution signal decomposition: the wavelet transform," *IEEE Trans. Patt. Anal. Mach. Intell.* **11**, 7 (1989) 674–693.
18. E. Salari and Z. Ling, "Texture segmentation using hierarchical wavelet decomposition," *Patt. Recogn.* **28**, 12 (1995) 1819–1824.
19. A. Teuner, O. Pichler and B. J. Hosticka, "Unsupervised texture segmentation of images using tuned matched Gabor filters," *IEEE Trans. Imag. Process.* **4**, 6 (1995) 863–870.
20. Tomita *et al.*, "Description of texture by a structural analysis," *IEEE Trans. Patt. Anal. Mach. Intell.* **PAMI-4**, 2 (1982) 183–191.



Kuen-Long Lee received both the B.S. and M.S. degrees in computer and information science from National Chiao Tung University, Taiwan, in 1988 and 1990, respectively. He is currently a Ph.D. student in the Department of Computer and Information Science at National Chiao Tung University.

His research interests include image processing, pattern recognition, texture analysis and image retrieval.

Ling-Hwei Chen received the B.S. degree in mathematics and the M.S. degree in applied mathematics from the National Tsing Hua University, Hsinchu, Taiwan in 1975 and 1977, respectively, and the Ph.D. in computer engineering from National Chiao Tung University, Hsinchu, Taiwan in 1987.

From August 1977 to April 1979, she worked as a research assistant in the Chung-Shan Institute of Science and Technology, Taoyan, Taiwan, after which she worked as a research associate in the Electronic Research and Service Organization, Industry Technology Research Institute, Hsinchu, Taiwan. From March 1981 to August 1983, she worked as an engineer in the Institute of Information Industry, Taipei, Taiwan and is now a Professor at the Department of Computer and Information Science at the National Chiao Tung University.

Her current research interests include image processing, pattern recognition, document processing, image compression and image cryptography.