Image and Texture Segmentation Using Local Spectral Histograms

Xiuwen Liu, *Member, IEEE*, and DeLiang Wang, *Fellow, IEEE*

***Abstract*—We present a method for segmenting images consisting of texture and nontexture regions based on local spectral histograms. Defined as a vector consisting of marginal distributions of chosen filter responses, local spectral histograms provide a feature statistic for both types of regions. Using local spectral histograms of homogeneous regions, we decompose the segmentation process into three stages. The first is the initial classification stage, where probability models for homogeneous texture and nontexture regions are derived and an initial segmentation result is obtained by classifying local windows. In the second stage, we give an algorithm that iteratively updates the segmentation using the derived probability models. The third is the boundary localization stage, where region boundaries are localized by building refined probability models that are sensitive to spatial patterns in segmented regions. We present segmentation results on texture as well as nontexture images. Our comparison with other methods shows that the proposed method produces more accurate segmentation results.**

***Index Terms*—Filtering, image segmentation, integral histogram image, local spectral histogram, spectral histogram, texture segmentation.**

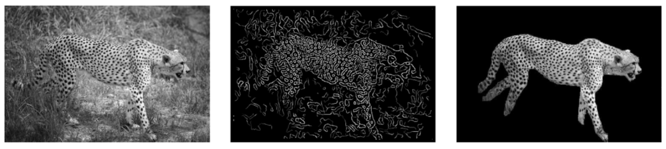

(a) (b) (c)

Fig. 1. Illustrative example. (a) Natural image of a cheetah. (b) Typical edge detection result. (c) Manually segmented result.

I. Introduction

SCENE analysis is a central task in numerous applications including autonomous robots, intelligent human computer interfaces, and content-based image and video retrieval systems. Its essential goal is to derive a meaningful description of the input. While a hypothetical solution can be constructed by building a look-up table with one entry for each possible input (see, e.g., [19]), the complexity of labeling the table entries, storing the table, and finding a match for a given input makes it impossible to implement.[1] Clearly, the complexity is a key requirement for scene analysis and the focus of research is thus on developing efficient and effective models and algorithms.

To illustrate the plausible approaches to scene analysis, Fig. 1(a) shows a natural image of a cheetah. One choice is to detect objects by exhaustively searching over all possible variations, including locations, scales, and orientations; this strategy is used by some face detection algorithms (e.g., [25]). This approach, however, is not effective given the large number of possible objects and their variations in the input. It seems that a more promising approach is to decompose scene analysis into two stages: an initial generic scene segmentation followed by some iterative recognition and model-specific segmentation loop. In this paper, we study the initial or bottom-up scene segmentation problem. Note that the initial segmentation stage is critical for the decomposition to be effective; otherwise, it would reduce to exhaustive detection. In this setting, the goal of scene segmentation is not to obtain an ideal segmentation—in fact, it was argued that an ideal segmentation like the one shown in Fig. 1(c) is not feasible without specific models [21]—but to provide candidate regions to initiate recognition, classification, or other higher level processes.

This formulation makes clear the goal of generic scene segmentation, as well as its constraints. Clearly, generic scene segmentation should not utilize object-specific models as they are not available. Also, a solution should work on different kinds of images such as texture and nontexture images. There are two broad categories of approaches to bottom-up scene segmentation. The first category is to group basic elements that can be computed easily (e.g., edges) to form more meaningful boundaries. However, meaningful elements may not be obtained readily due to complex textures of natural objects. For example, Fig. 1(b) shows a typical edge-detection result for Fig. 1(a) using the Canny edge detector [4]. It is clear that grouping these edges into meaningful boundary segments is a complicated task, if not infeasible. The other category is to identify regions based on statistics of local features, including region growing, split-and-merge, and so on. Here, segmentation can be defined as a constrained partition problem, where each partitioned region should be as homogeneous as possible and neighboring ones should be as different as possible. Two fundamental issues can be identified. The first one is the underlying image model that defines region homogeneity and thus specifies what a good segmentation should be, and the second one is to design an algorithm to compute a solution, exactly or approximately.

Manuscript received January 24, 2005; revised January 13, 2006. The work of X. Liu was supported in part by the National Science Foundation under Grants IIS-0307998 and CCF-0514743 and in part by the ARO under Grant W911NF-04-01-0268. The work of D. L. Wang was supported in part by the AFOSR under Grant FA9550-04-1-0117 and in part by the AFRL. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Eli Saber.

X. Liu is with the Department of Computer Science, Florida State University, Tallahassee, FL 32306-4530 USA (e-mail: liux@cs.fsu.edu).

D. Wang is with the Department of Computer Science and Engineering and Center for Cognitive Science, The Ohio State University, Columbus, OH 43210 USA (e-mail: dwang@cse.ohio-state.edu).

Digital Object Identifier 10.1109/TIP.2006.877511

[1]For example, assuming that the input is a gray-level image of size 256×256 with pixel levels from 0 to 255, the table would have about $10^{157,826}$ entries.

There are numerous algorithms for segmentation. A key difference among them is the underlying segmentation criterion, either explicitly or implicitly defined. For gray-level images with piecewise smooth regions, the Mumford-Shah model [16] is representative in that criteria used in most existing segmentation algorithms are its special cases [15]. While their model has led to numerous optimization algorithms (see, e.g., [5] and [7]), its success is limited to nontexture images; for texture regions, local smoothness is not adequate as a homogeneity criterion. In this paper, we extend the Mumford-Shah model to images consisting of piecewise smooth texture and nontexture regions using local spectral histograms [10], [12]. Because a local spectral histogram of an image window consists of histograms of response images of chosen filters, it captures local spatial patterns through filtering and global patterns through histograms and constraints among different filters. Assuming that a representative spectral histogram is available for each region, we derive a segmentation algorithm as follows: 1) estimating a probability model for each region and classifying image windows to obtain an initial segmentation; 2) iteratively updating segmentation and local spectral histograms of pixels along region boundaries based on the derived probability models; and 3) further localizing region boundaries using refined probability models derived based on spatial patterns in segmented regions. Additionally, we address the issues of automatically identifying representative regions. While histograms of filtered images have been used for texture modeling [2], [8], [10], [28], texture classification [1], [10], [13], and segmentation based on clustering [9], the proposed method leads to robust and accurate segmentation by using adaptive feature extraction and boundary localization, which have not been studied and used. The main contributions of this paper are 1) we give a segmentation method that is effective for both texture and nontexture regions, justified by experimental results on different kinds of images; 2) we give new ways to estimate probability models, which lead to more accurate segmentation results in comparison with normalized cut method [20] and other methods [6], [17]; and 3) we give a novel algorithm that computes spectral histograms of local windows efficiently.

The rest of the paper is organized as follows. Section II introduces the local spectral histogram representation and discusses its properties; we also present an algorithm that computes spectral histograms of local windows using a fixed number of arithmetic operations regardless of window size. In Section III, we present our segmentation algorithm which couples the feature detection and segmentation steps together by extracting features based on the currently available segmentation result. In Section IV, we propose a novel method to further localize region boundaries. Section V presents an algorithm for identifying regional features in homogeneous texture regions. Section VI provides experimental results and comparisons, and Section VII concludes the paper with a summary.

II. Local Spectral Histogram Representation

Given a window $\mathbf{W}$ in an input image $\mathbf{I}$ and a chosen bank of filters $\{F^{(\alpha)}, \alpha = 1, 2, \ldots, K\}$, we compute, for each filter $F^{(\alpha)}$, a sub-band image $\mathbf{W}^{(\alpha)}$ through a linear convolution, i.e., $\mathbf{W}^{(\alpha)}(\vec{v}) = F^{(\alpha)} * \mathbf{W}(v) = \sum_{\vec{u}} F^{(\alpha)}(\vec{u})\mathbf{W}(\vec{v} - \vec{u})$, where

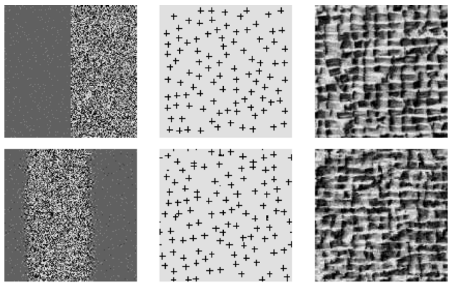


Fig. 2. Different types of images characterized by spectral histograms. In each column, the top row shows an observed image and the bottom a typical image that shares the same spectral histogram. (a) Gray-level image consisting of two piecewise constant regions with additive Gaussian noise. (b) Texton image consisting of cross elements. (c) Stochastic texture image.

$\vec{u}$ and $\vec{v}$ specify pixel locations. For $\mathbf{W}^{(\alpha)}$, we define its histogram, a bin of which is given by

$$H_{\mathbf{W}}^{(\alpha)}(z_1, z_2) = \sum_{\vec{v} \in \mathbf{W}} \int_{z_1}^{z_2} \delta(z - \mathbf{W}^{(\alpha)}(\vec{v}))dz \tag{1}$$

where z_1 and z_2 specify the range of the bin. We then define the spectral histogram with respect to the chosen filters as

$$H_{\mathbf{W}} = \frac{1}{|\mathbf{W}|}\left(H_{\mathbf{W}}^{(1)}, H_{\mathbf{W}}^{(2)}, \ldots, H_{\mathbf{W}}^{(K)}\right). \tag{2}$$

The spectral histogram of an image window is essentially a vector consisting of marginal distributions of filter responses and the size of the window is called *integration scale*. Because they consist of probability distributions, a similarity measure between two spectral histograms can be defined as χ^2-statistic (e.g., [9]) given by

$$\chi^2(H_{\mathbf{W}_1}, H_{\mathbf{W}_2}) = \frac{1}{|\mathbf{W}|}\sum_{\alpha=1}^{K}\sum_{z} \frac{\left(H_{\mathbf{W}_1}^{(\alpha)}(z) - H_{\mathbf{W}_2}^{(\alpha)}(z)\right)^2}{H_{\mathbf{W}_1}^{(\alpha)}(z) + H_{\mathbf{W}_2}^{(\alpha)}(z)}. \tag{3}$$

The spectral histogram provides a normalized feature statistic to compare image windows of different sizes. The input image windows do not need to be aligned; misalignment is a serious problem for approaches that use filter responses directly as features, such as those studied in [17], due to the inhomogeneity of filter responses. When proper filters are chosen, the spectral histogram is sufficient in characterizing texture appearance. Fig. 2 shows three types of images, where the typical images are generated using a Gibbs sampler [27]. In Fig. 2(a), the spectral histogram captures the perceptual appearance of both regions. Given that the circular boundary is used for a typical image, the typical image represents closely the observed one. Fig. 2(b) shows a texton image, where the spectral histogram captures the texton elements and the element density. Fig. 2(c) shows a stochastic texture and the spectral histogram captures the perceptual appearance well. In this paper, eight filters are used: the intensity filter, two gradient filters, LoG with two scales and three Gabor filters with different orientations; they are not tuned for particular images but chosen to capture general aspects of texture as well as nontexture regions (see Section VI-C for comparison of different filters).

Note that the spectral histogram is defined on any type of images. Piecewise-constant images with additive Gaussian noise are a special case where the spectral histogram has a unique pattern. In addition, it can also characterize patterns with topological structures (e.g., a face) with appropriate boundary conditions [11]. The spectral histogram representation was first suggested in psychophysical studies on texture modeling [2], and has been used in texture modeling and synthesis [8], [28], [10], texture classification [1], [10], [13], and modeling human texture discrimination [12]. Histograms of local fields have also been used for object recognition and detection [11], [18], [25]. While spectral histogram representations have been used for classification where labeled training samples are assumed [11], [13], this paper addresses image segmentation where no labeled training samples are available. For robust segmentation, we introduce adaptive feature extraction and boundary localization, which are not studied in earlier classification and segmentation work based on similar features [1], [9], [13].

A. Fast Implementation

By definition, segmentation is to compute homogeneous regions and thus requires evaluations of local spectral histograms of different windows when they are used (see Section III for details). To characterize textures, $\mathbf{W}$ should be relatively large and therefore fast computation is needed. As shown in (1), calculating the histogram of local window $\mathbf{W}$ of a particular filter requires summation over all the pixels in $\mathbf{W}$. Let $\mathbf{W}$ be given by $(x_0, y_0) \leq \vec{u} \leq (x_1, y_1)$ and $\mathbf{I}^{(\alpha)}$ be the filtered image of image $\mathbf{I}$ using filter $F^{(\alpha)}$. By rewriting (1) in terms of $\mathbf{I}^{(\alpha)}$, we have

$$\begin{aligned} H_{\mathbf{W}}^{(\alpha)}(z_1, z_2) &= \sum_{(x_0,y_0)\leq\vec{v}\leq(x_1,y_1)} \int_{z_1}^{z_2} \delta(z - \mathbf{I}^{(\alpha)}(\vec{v}))dz \\ &= \sum_{(0,0)\leq\vec{v}\leq(x_1,y_1)} \int_{z_1}^{z_2} \delta(z - \mathbf{I}^{(\alpha)}(\vec{v}))dz \\ &\quad + \sum_{(0,0)\leq\vec{v}\leq(x_0-1,y_0-1)} \int_{z_1}^{z_2} \delta(z - \mathbf{I}^{(\alpha)}(\vec{v}))dz \\ &\quad - \sum_{(0,0)\leq\vec{v}\leq(x_0-1,y_1)} \int_{z_1}^{z_2} \delta(z - \mathbf{I}^{(\alpha)}(\vec{v}))dz \\ &\quad - \sum_{(0,0)\leq\vec{v}\leq(x_1,y_0-1)} \int_{z_1}^{z_2} \delta(z - \mathbf{I}^{(\alpha)}(\vec{v}))dz. \end{aligned} \tag{4}$$

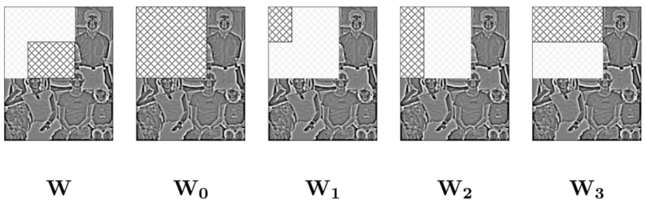


Fig. 3. Illustration that any rectangular $\mathbf{W}$ can be written as $\mathbf{W} = \mathbf{W_0} + \mathbf{W_1} - \mathbf{W_2} - \mathbf{W_3}$. In each image, the corresponding region is marked as hatched.

Here, $(0,0)$ specifies the top-left corner of the image. If we let $\mathbf{W_0}$ be the rectangular window given by $(0,0) \leq \vec{u} \leq (x_1, y_1)$, $\mathbf{W_1}$ by $(0,0) \leq \vec{u} \leq (x_0-1, y_0-1)$, $\mathbf{W_2}$ by $(0,0) \leq \vec{u} \leq (x_0-1, y_1)$, and $\mathbf{W_3}$ by $(0,0) \leq \vec{u} \leq (x_1, y_0-1)$, we have $\mathbf{W} = \mathbf{W_0} + \mathbf{W_1} - \mathbf{W_2} - \mathbf{W_3}$, as illustrated in Fig. 3. This gives the following:

$$\begin{aligned} H_{\mathbf{W}}^{(\alpha)}(z_1, z_2) &= H_{\mathbf{W_0}}^{(\alpha)}(z_1, z_2) \\ &\quad + H_{\mathbf{W_1}}^{(\alpha)}(z_1, z_2) - H_{\mathbf{W_2}}^{(\alpha)}(z_1, z_2) - H_{\mathbf{W_3}}^{(\alpha)}(z_1, z_2) \end{aligned} \tag{5}$$

according to (4). Now we define an *integral histogram image* $HI_{(z_1,z_2)}^{(\alpha)}$ for the bin, where the value at pixel (x,y) is $H_{\mathbf{W}'}^{(\alpha)}(z_1, z_2)$ and $\mathbf{W}'$ is the rectangular window given by $(0,0) \leq \vec{u} \leq (x, y)$. In other words, we define $HI_{(z_1,z_2)}^{(\alpha)}(x,y)$ as

$$HI_{(z_1,z_2)}^{(\alpha)}(x,y) = \sum_{(0,0)\leq\vec{v}\leq(x,y)} \int_{z_1}^{z_2} \delta(z - \mathbf{I}^{(\alpha)}(\vec{v}))dz. \tag{6}$$

Through the integral histogram image, as shown in (5), we can compute $H_{\mathbf{W}}^{(\alpha)}$ using L additions and $2 \times L$ subtractions, where L is the number of the bins in the histogram. To compute the integral histogram image, we use (7), shown at the bottom of the page. By defining $RS_{(z_1,z_2)}^{(\alpha)}(x,y) = \sum_{(0,y)\leq\vec{v}\leq(x,y)} \int_{z_1}^{z_2} \delta(z - \mathbf{I}^{(\alpha)}(\vec{v}))dz$, we have

$$\begin{aligned} &RS_{(z_1,z_2)}^{(\alpha)}(x,y) \\ &= \begin{cases} RS_{(z_1,z_2)}^{(\alpha)}(x-1,y) + 1, & \text{if } z_1 \leq \mathbf{I}^{(\alpha)}(x,y) < z_2 \\ RS_{(z_1,z_2)}^{(\alpha)}(x-1,y), & \text{otherwise} \end{cases} \end{aligned} \tag{8}$$

when $x \geq 1$. When $x = 0$, we have

$$RS_{(z_1,z_2)}^{(\alpha)}(0,y) = \begin{cases} 1, & \text{if} z_1 \leq \mathbf{I}^{(\alpha)}(x,y) < z_2 \\ 0, & \text{otherwise.} \end{cases} \tag{9}$$

Now, integral histogram image $HI_{(z_1,z_2)}^{(\alpha)}$ can be computed quickly by first computing $RS_{(z_1,z_2)}^{(\alpha)}(x,y)$ using (9) and (8) and then $HI_{(z_1,z_2)}^{(\alpha)}$ using (7). This has been shown in [22] but for computing special filter responses. Note that $\mathbf{I}^{(\alpha)}$ only

$$HI_{(z_1,z_2)}^{(\alpha)}(x,y) = \begin{cases} HI_{(z_1,z_2)}^{(\alpha)}(x,y-1) + \sum_{(0,y)\leq\vec{v}\leq(x,y)} \int_{z_1}^{z_2} \delta(z - \mathbf{I}^{(\alpha)}(\vec{v}))dz, & \text{if } y \geq 1 \\ \sum_{(0,0)\leq\vec{v}\leq(x,0)} \int_{z_1}^{z_2} \delta(z - \mathbf{I}^{(\alpha)}(\vec{v}))dz, & \text{if } y = 0 \end{cases} \tag{7}$$

needs to be computed once and field programmable gate array (FPGA) devices can be used for fast implementation [23].

The computation for computing local spectral histograms can be further reduced when the size of the image is known. For example, for a 256×256 image, the number in any bin can be at most $256 \times 256 = 65,536 \leq 2^{16}$, which means that we only need 16 bits to represent any bin in the integral histogram image. By using 128-bit words,[2] we can encode 8 bins in one word; this reduces the number of operations to compute the local histogram feature of a filter to $3 \times \lceil L/8 \rceil$. For $L = 8$, as is the case in all the experiments shown here, we only need three arithmetic operations to compute $H_{\mathbf{W}}^{(\alpha)}$ and 24 to compute $H_{\mathbf{W}}$ (for eight filters) for any window size.

III. Segmentation Algorithm

As discussed in Section I, the goal of segmentation is to partition an input image into regions so that each one is as homogeneous as possible and neighboring ones are as different as possible. To be able to model both texture and nontexture regions, we use local spectral histograms as local features and measure homogeneity of a region using distance among spectral histograms given by (3). The problem now becomes an optimization one and could be solved using algorithms developed for the Mumford and Shah model (e.g., [5] and [7]). However, due to the large integration scale for texture regions, these algorithms would give inaccurate region boundaries as windows crossing multiple regions do not give accurate features. To overcome this problem, here, we develop an approximation algorithm that couples the feature extraction and segmentation iteratively, implemented in three stages.

To be more specific, let $\mathcal{L}$ be a grid defined on a planar domain and R_i, $i = 1, \ldots, n$, be a disjoint subset of $\mathcal{L}$. We assume that a feature $\mathcal{F}_i$ is associated with each region R_i, and the feature can be given manually or selected automatically (see Section V); each feature vector $\mathcal{F}_i$ is extracted from window W^s centered at a given or detected pixel location. Here, W^s is the integration scale for segmentation, which is assumed to be given. We also define R_0, which is called *background* [24], as $R_0 = \mathcal{L} - (R_1 \cup \cdots R_n)$, consisting pixels that cannot be segmented into one of the given regions. Under this formulation, further assuming feature vectors $\mathcal{F}_i$s are sufficiently accurate, homogeneous regions can be obtained by classification. However, features computed using windows that are centered at pixel locations are inaccurate at locations along region boundaries, because the windows of such locations straddle multiple regions (see Fig. 6). Hence, we decompose the segmentation into three stages. We first estimate probability models for each region and obtain an initial segmentation by classification. Then we iteratively update segmentation by coupling the feature extraction and segmentation. The third stage performs an additional boundary localization stage to be described in Section IV.

To estimate probability models and parameters T_i— homogeneity thresholds for regions—we compute the spectral histogram centered at a pixel location; to save computation, it is done only at subsampled pixels. The χ^2-statistic distance may

[2]Available in SSE2 and SSE3 instructions on recent Intel processors.

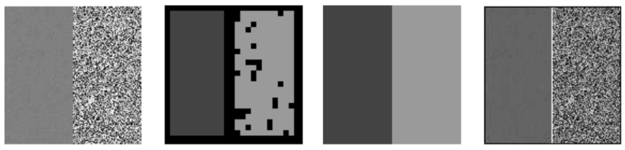


Fig. 4. Gray-level image segmentation using spectral histograms. Integration scale W^s for spectral histograms is a 15×15 square window. (a) Synthetic image with size 128×128. The image is generated by adding zero-mean Gaussian noise with different σ's at left and right regions. (b) Initial classification result. Two features are given at (32, 64) and (96,64), where the first element in each pair indicates the x or horizontal dimension. (c) Segmentation result. Each region is represented by a manually assigned gray value. (d) Segmentation result shown by region boundary (white). Note that the original image is dimmed to make the boundary more visible.

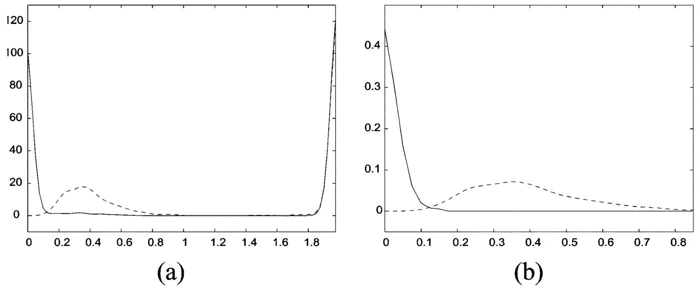


Fig. 5. Histograms and derived probability models of χ^2-statistic for the given region features in Fig. 4. Solid lines stand for left region and dashed lines stand for right region. (a) Histogram of the χ^2-statistic between the given feature and the computed ones at a coarse grid. (b) Derived probability model for the left and right regions.

not provide an accurate measure close to region boundaries due to inhomogeneity between different regions. For example, in the image shown in Fig. 4, the left region is homogeneous and the variation allowed should be small. In the right region, the variation allowed should be relatively large. To overcome this problem and provide a more accurate model, we estimate a probability model of the χ^2-statistic for each given feature vector $\mathcal{F}_i$. This is done by computing the histogram of the χ^2-statistic between the computed spectral histograms at all the pixels of the image and the given spectral histogram $\mathcal{F}_i$ for the region. Fig. 5(a) shows the two histograms of the χ^2-statistic for two given features corresponding to the two regions of the image in Fig. 4(a). Parameter T_i for each region is determined to be the first (smallest) local minimum position, which provides natural separation to include pixel locations whose features are close to the given one. This is done by detecting zero-crossings using a one-dimensional (1-D) LoG filter [14]. We then assign zero probability for histogram values larger than T_i for the region and obtain the probability model by normalizing the sum to 1. The derived probability models for Fig. 4(a) are given in Fig. 5(b). Then, the input image is classified using a maximum likelihood classifier using the derived probability models; the pixels whose minimum distance is larger than T_i from all regions, are classified as background. The classification result is used as the initial segmentation. For the image in Fig. 4(a), the corresponding initial segmentation result is shown in Fig. 4(b). Note that the classification is done at subsampled pixels.

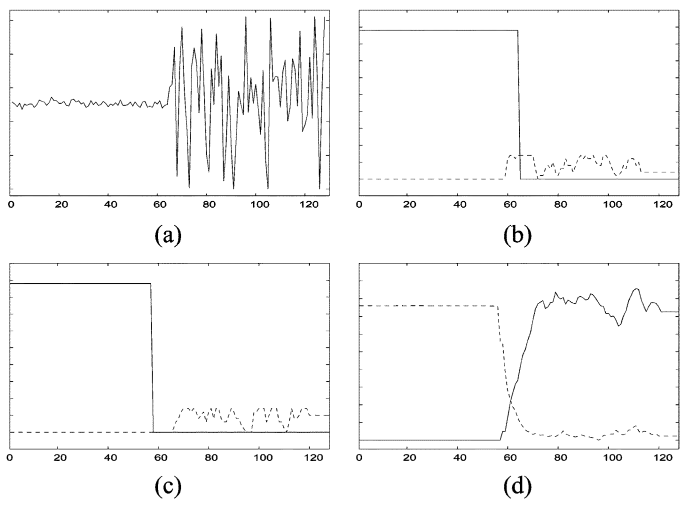


Fig. 6. Effectiveness of derived probability models. In (b)–(d), solid lines stand for the left region and dashed lines the right region. The true edge is located between pixels 63 and 64. (a) The 64th row from Fig. 4(a). (b) Probability of the two given regional features using asymmetric windows to compute spectral histograms. (c) Similar to (b) but using windows centered at the pixel to compute spectral histogram. Here, the edge point cannot be localized. (d) χ^2-statistic from the two given regional features using centered windows.

After we obtain the initial segmentation and probability models, the segmentation is then iteratively updated based on the following local updating rule at pixel (x, y):

$$\pi_i(x, y) = P(\chi^2(H_{W^s(x,y)}, \mathcal{F}_i)). \tag{10}$$

For a given pixel (x, y) to be updated, one can estimate the spectral histogram using a window centered at the pixel. However, as noted earlier, features estimated using centered, symmetric windows can be noisy near region boundaries. To alleviate this problem, we propose to use asymmetric windows for feature extraction. Specifically, we use square windows of size W^s where the given pixel is a corner. Because there are four windows to choose at pixel (x, y), for each $\mathcal{F}_i$, we use the window that has the most number of labeled pixels in R_i, and thus the window or feature at pixel (x, y) for different labels can be different. The new label of (x, y) is assigned the one that gives the maximum $\pi_i(x, y)$. As in the initial segmentation stage, a pixel remains in the background if the minimum distance to all regions R_i, $1 \leq i \leq n$, is larger than the corresponding T_i using asymmetric windows. Fig. 4(c) shows the segmentation result for the image in Fig. 4(a), and the resulting region boundary is shown in Fig. 4(d). Here, all the pixels are segmented correctly.

To illustrate the effectiveness of the derived probability models and asymmetric windows, Fig. 6(a) shows a row from the image shown in Fig. 4(a). Fig. 6(b) shows the probability of the two labels at each pixel using asymmetric windows. We can see that the edge point is localized precisely at the true location. Fig. 6(c) show the probability using symmetric windows centered at pixels. There is an interval where labels can not be decided because the spectral histogram computed in the interval does not belong to either of the regions. For comparison, Fig. 6(d) shows the result using χ^2-statistic directly for centered windows. Here, boundaries are systematically shifted by several pixels because χ^2-statistic distance favors homogeneous regions.

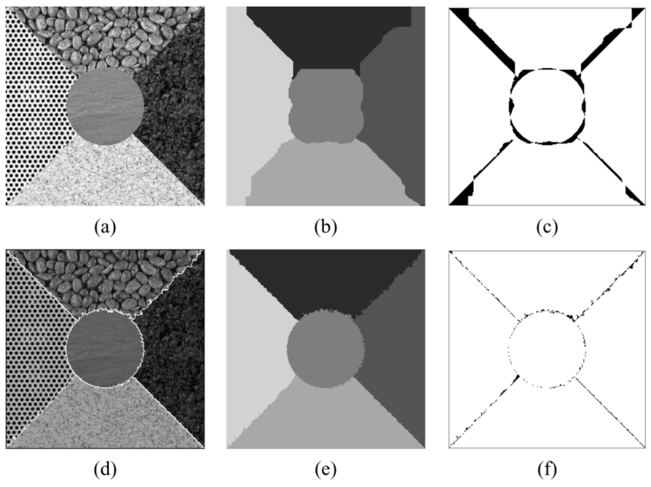


Fig. 7. (a) Texture image with size 256 × 256. (b) Segmentation result using spectral histograms. (c) Wrongly segmented pixels of (b), represented in black with respect to the ground truth. Segmentation error is 6.55%. (d), (e) Refined segmentation result shown by region boundaries and by regions respectively. (f) Wrongly segmented pixels of (e), represented in black as in (c). Segmentation error is 0.95%.

IV. Localization of Region Boundaries

Because textures need to be characterized by spatial relationships among pixels, relatively large integration windows are needed in order to extract meaningful features. A large integration scale however results in large errors along texture boundaries due to the uncertainty introduced by large windows [4]. By using asymmetric windows for feature extraction, the uncertainty effect is reduced. However, for arbitrary texture boundaries, the errors along boundaries can be large even when the overall segmentation performance is good. For example, Fig. 7(b) shows a segmentation result on the image shown in Fig. 7(a) using spectral histograms. While the segmentation error is only 6.55%, the segmentation result is visually intolerable due to large errors along texture boundaries, as shown in Fig. 7(c).

In order to accurately localize texture boundaries, we build a refined probability model using given m pixels from each region. To capture the spatial relationship, we choose for each texture region a window as a template; the template is the same window from which the region feature $\mathcal{F}_i$ is extracted. For selected m pixels, we define their distance to a texture region as the minimum mean square distance between those pixels and the template. Using these distances of segmented regions, we build a refined probability model for each texture region as in the first step. Then we iteratively update the segmentation again using (10) but using the refined probability models. Intuitively, if the m pixels belong to a texture region, it should match the spatial relationship among pixels when they are aligned with the texture structure. These probability models are sensitive to alignments and thus should produce more accurate region boundaries than those based on the spectral histograms.

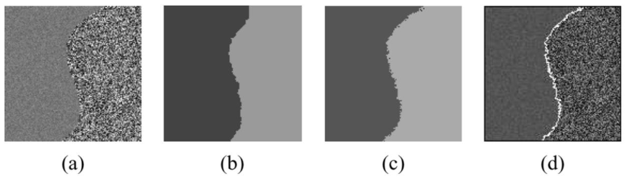


Fig. 8. (a) Synthetic image with size 128×128. (b) Segmentation result. (c), (d) Refined segmentation result shown as regions and as the region boundary respectively.

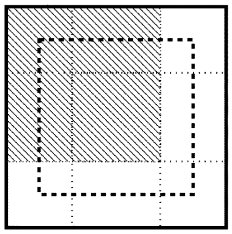


Fig. 9. Windows for automated seed selection. Here, the solid window is W^a, dashed is the central W^s and the dotted are four corner ones. For clarity, the top-left corner window is shown hatched.

Fig. 7(e) shows the refined segmentation result with $m = 11$ pixels and Fig. 7(d) shows the corresponding region boundaries. The segmentation error is reduced to 0.95%, and visually, the segmentation result is improved significantly. Fig. 7(f) shows the wrongly segmented pixels of the refined segmentation. Similar images were used by Hofmann *et al.* [9]. Compared to their results, ours is comparable to their best without expensive optimization. Fig. 8(a) shows an intensity image with a curvy boundary between two regions with the same mean but different variance. Fig. 8(b) shows the segmentation result, Fig. 8(c) shows the refined result, and Fig. 8(d) shows the region boundary. It is clear that the boundary between two regions is improved significantly, especially at the top and bottom borders of the image.

V. Automated Seed Selection

In the segmentation algorithm presented above, we assume that representative pixels or seeds located near centers of distinct regions are given. This assumption can limit the use of the proposed method in autonomous systems. Here, we present an algorithm for identifying homogeneous texture regions in a given image. Within a homogeneous texture region relative to an integration scale, spectral histograms calculated at different windows should be similar. Based on this observation, we try to identify homogeneous texture regions based on distance measures with respect to two integration scales. Let W^a be an integration scale 1.5 times larger than W^s (in terms of number of pixels; see Fig. 9), the integration scale for segmentation. We define two distance measures at pixel (x, y) with respect to W^s and W^a, $\psi^B_{W^s}$ and $\psi^I_{W^s}$, given by

$$\psi^B_{W^s}(x,y) = D(H_{W^s(x,y)}, H_{W^a(x,y)}) \quad (11)$$

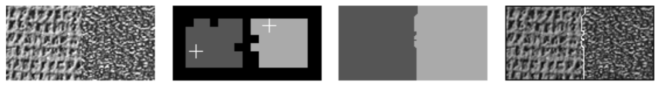


Fig. 10. Texture image segmentation with representative pixels identified automatically with $W^s = 29 \times 29$ and $m = 11$. (a) Input texture image. (b) Initial classification result. Here, the representative pixels (marked as white '+') are detected automatically. (c), (d) Segmentation result and the resulting region boundary.

and

$$\psi^I_{W^s}(x,y) = \max_{W^s(x_1,y_1) \subset W^a} D(H_{W^s(x,y)}, H_{W^s(x_1,y_1)}). \quad (12)$$

Here, $\psi^B_{W^s}$ measures the difference between the spectral histograms of W^s and W^a. Within a homogeneous texture region, $\psi^B_{W^s}$ should be small because $H_{W^s}(x,y)$ and $H_{W^a}(x,y)$ should be similar. Similarly, $\psi^I_{W^s}$ measures the maximum variation among different windows of scale W^s within W^a; it should be also small within a homogeneous region. To save computation, $\psi^I_{W^s}$ is approximated in implementation using four corner windows within W^a as illustrated in Fig. 9. Additionally, we want to choose features that are as different as possible from those already chosen. Suppose we have chosen k features already, where, $\mathcal{F}_i = H_{W^s(x_i,y_i)}$, for $i = 1, \ldots, k$, we define

$$\psi^F(x,y) = \max_{1 \le i \le k} D(H_{W^s(x,y)}, \mathcal{F}_i) \quad (13)$$

which gives a distance measure between the candidate feature at (x, y) and all other chosen ones. Combining these together, we have the following saliency measure:

$$\begin{aligned}\psi(x,y) = (1-\lambda_C)\big(&\lambda_A \times \psi^B_{W^s}(x,y) \\ &+(1-\lambda_A)\times \psi^I_{W^s}(x,y)\big) - \lambda_C \times \psi^F(x,y). \quad (14)\end{aligned}$$

Here, λ_A and λ_C are parameters to determine the relative contribution of each term, and we set $\lambda_A = 0.2$ and $\lambda_C = 0.1$ for all the experiments, determined empirically over a set of images. Intuitively, $\psi(x,y)$ should be small in a homogeneous region that has not been represented. Therefore, we select seed windows according to $\psi(x,y)$ until it becomes large.

Fig. 10 shows a seed selection example, where an image with two texture regions is shown in Fig. 10(a). Fig. 10(b) shows the initial classification result using automatically detected feature vectors. Fig. 10(c) shows the segmentation result. The texture boundary is localized well even though the two textures are similar in local intensity values.

VI. Experimental Results and ComparisonS

A. *Segmentation Results*

Figs. 11 and 12 show the segmentation results for a set of texture images. For each image, first the feature vectors are identified automatically and an initial result is then obtained using a maximum likelihood classifier with the found region features.

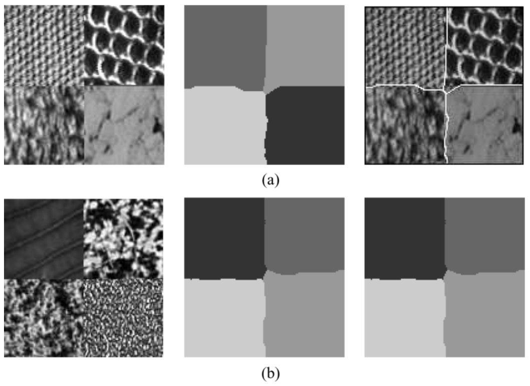


Fig. 11. Texture image segmentation examples with $W^s = 29 \times 29$. In each panel, the left is the input image, the middle the final segmentation result, and the right the final region boundaries. All representative pixels are detected automatically. (a) $m = 11$. (b) $m = 15$.

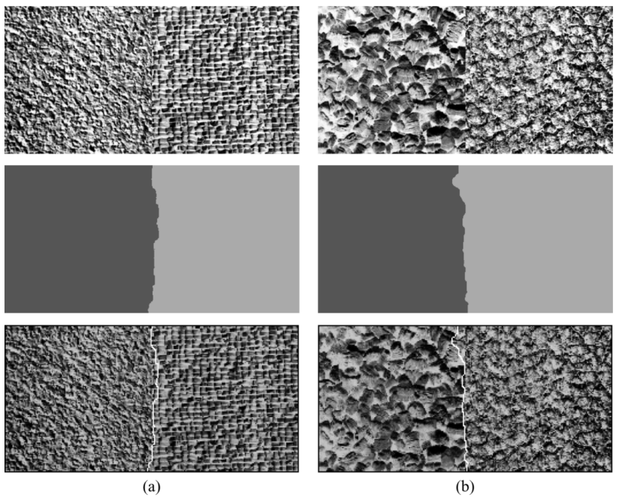


Fig. 12. Texture image segmentation examples with $W^s = 29 \times 29$ and $m = 11$. In each column, the top is the input image, the middle the segmentation result and the bottom the region boundaries. All representative pixels are detected automatically.

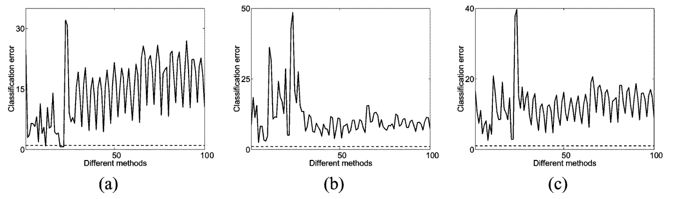


Fig. 13. Classification error (%) of 100 methods used in [17]. In each plot, the dashed line is the corresponding segmentation error of our results. (a) For image in Fig. 12(a). (b) For image in Fig. 12(b). (c) Average performance on the two images.

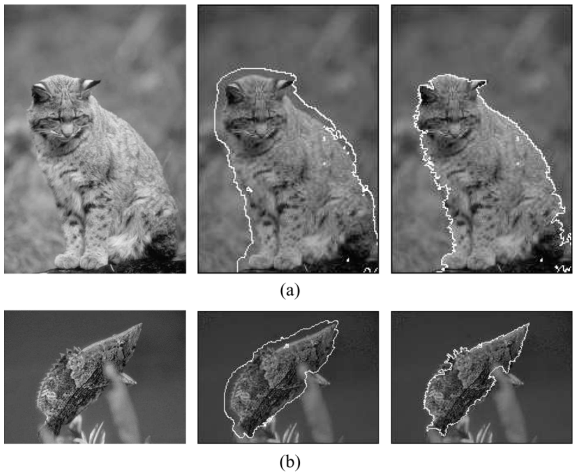


Fig. 14. Natural image segmentation examples with $W^s = 29 \times 29$ and $m = 25$. In each panel, the left is the input image, the middle segmentation result before boundary localization, and the right the result after boundary localization. (a) Cat image with size 305 × 450. (b) Fish image with size 438 × 321.

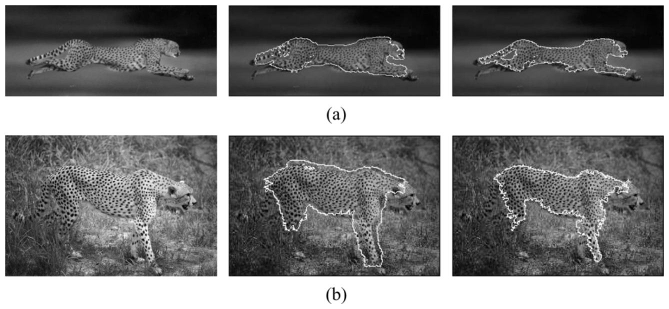


Fig. 15. More natural image segmentation examples with $W^s = 29 \times 29$ and $m = 25$. See Fig. 14 for figure legend. (a) Cheetah image with size 795 × 343. (b) Another cheetah image with size 486 × 324.

The segmentation is then updated iteratively and the final result is obtained by applying the boundary localization algorithm. As shown in these examples, the texture boundaries are localized well and all the homogeneous texture regions are identified by the seed selection algorithm. The inaccuracy of the boundaries is mostly due to the similarity of the textures along the boundaries as well as large texture structures and variations in these examples. Two examples shown in Fig. 12 are also used by Randen and Husoy [17] in their comparative study of 100 texture-classification methods. Their experiments were set up as supervised classification, where training features were provided first. Thus, the segmentation problem studied here is essentially more difficult.[3] Fig. 13(a) and (b) show the classification error of the 100 texture classification methods for Fig. 12(a) and (b) respectively. In both cases, the segmentation error of our results is 1.0%. For Fig. 12(a), only four methods perform slightly better (one 0.9% and three 0.7%) than ours. However, these four methods along with others perform significantly worse than our method on Fig. 12(b). To show this, Fig. 13(c) shows the average performance on both images and clearly our method gives the lowest error rate. This significant improvement in performance is due to the desirable properties of the spectral histogram representation and the derived probability models.

Natural images in general consist of regions that are not homogeneous and we are often interested in some meaningful regions only. This can be achieved in our system by identifying only a few region features. As mentioned before, no assumption is made in our algorithm regarding the distributions and properties of background regions, and thus we avoid building models for them. We apply the same algorithm but with one region identified and Figs. 14 and 15 show some examples. The left column shows the input images and the middle shows the segmentation result before boundary localization. The final boundary-localized result is shown in the right column. To show the accuracy of segmented boundaries, they are embedded in the original

[3]For texture classification comparisons, see [13].

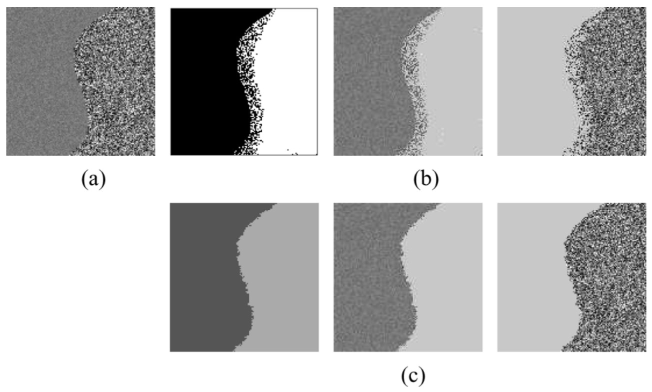


Fig. 16. Comparison between normalized cut and proposed method on a gray-level image. (a) Input image, as shown in Fig. 8(a). (b), (c) Segmentation result from normalized cut and our method. In each panel, the left is the segmentation result and the rest are the components. Here, a gray color is used to indicate pixels not in the current component.

image. As these examples show, our algorithm gives accurate region boundaries. Some parts of the boundaries are not localized well due to the similarity between the region of interest and the background. If we had models for recognition, the cheetah in Fig. 15(a) and (b) could be recognized due to its distinctive skin pattern and then the segmentation result could be further improved using top-down information. Given that our system is generic and there is no image specific training, our results are comparable with best available results.

B. Comparisons

To provide additional empirical justifications of our method, we have compared our segmentation method with the normalized cut algorithm proposed by Shi and Malik [20] and then with a feature fusion method by Clausi and Deng [6], which is mainly for texture segmentation. Normalized cut is a general framework for image segmentation based on the graph theoretic formulation of grouping. Each pixel in the image is viewed as a vertex in a graph and the edges between pixels (vertices) are constructed based on similarity measures between the corresponding feature vectors. To deal with texture as well as non-texture regions, Shi and Malik include features based on intensity, color, and filter responses. After the edges are established, the segmentation problem is posed as a graph cutting problem of the constructed graph. Using a normalized cut criterion, Shi and Malik proposed an efficient algorithm by converting the cutting problem into an eigen one [20] based on the spectral graph theory. The algorithm has been applied to segmenting gray-level images and texture images.

Here, we used an implementation by Shi.[4] For each case shown below, we have tried different combinations of parameters to obtain the best performance we can. First, we applied the normalized cut on the gray-level image shown in Fig. 8(a) and Fig. 16(b) shows the segmentation result. Following Shi and Malik, Fig. 16(b) also shows the segmented components from normalized cut and Fig. 16(c) shows our corresponding segmentation and components. In this case, the normalized cut successfully segments two regions out. However, the region boundary is not well localized because the normalized cut does not exploit the spatial connectivity in segmented regions and

[4]Obtained from http://www.hid.ri.cmu.edu/Hid/software_ncutPublic.html.

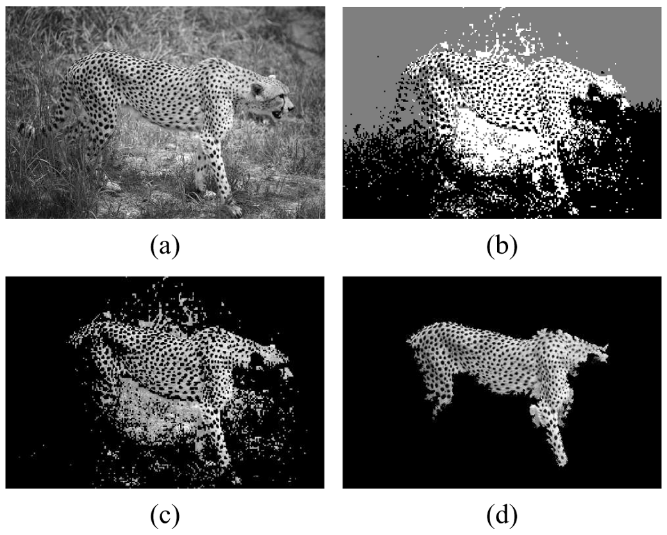


Fig. 17. Comparison between normalized cut and the proposed method on the cheetah image [Fig. 15(b)]. In (c) and (d), black is used to indicate pixels not in the current component. (a) Input image. (b) Segmentation result from normalized cut. (c) Cheetah segment from (b). (d) Cheetah segment from our method.

does not have a refined model for boundary localization. Even without boundary localization [see Fig. 8(b)], our result still gives more accurate region boundary.

Fig. 17 compares the normalized cut method and ours on the cheetah image shown in Fig. 15(b). Here, we compare the segment corresponding to the cheetah region. While both methods are able to segment the cheetah out, our segment is localized more accurately even without boundary localization [see Fig. 15(b)]. Note that the normalized cut gives the cheetah region as one of its segments and does not localize its boundary.

Fig. 18(b) shows the normalized cut segmentation and its major components for the image shown in Fig. 7(a). For comparison, we have shown our segmentation result in the same format in Fig. 18(c). As in the previous example, the major regions are segmented out by their algorithm. While boundaries between very different regions are localized well, boundaries are not accurate when neighboring regions are similar. In several cases, regions from different textures are mixed. On the other hand, our segmentation gives accurate region boundaries; note that without boundary localization [see Fig. 7(b)], ours still gives more accurate boundaries. We attribute the difference to the derived probability models and to the fact that the spectral histogram is more reliable to characterize texture regions than filter responses, which are used directly in the normalized cut method.

Based on these examples, we can see that our results generally give more accurate region boundaries. On the other hand, our method can be combined with the normalized cut method in several ways. First instead of using the filter responses to characterize textures, the normalized cut can use local spectral histograms and distance between them to establish edges in the underlying graph and can even use the probability models we have. Second, our boundary localization algorithm can be used to localize segments from normalized cut. Additionally, instead of using the minimum-distance classification to obtain an initial segmentation for our method, we may use the normalized cut to generate an initial segmentation.

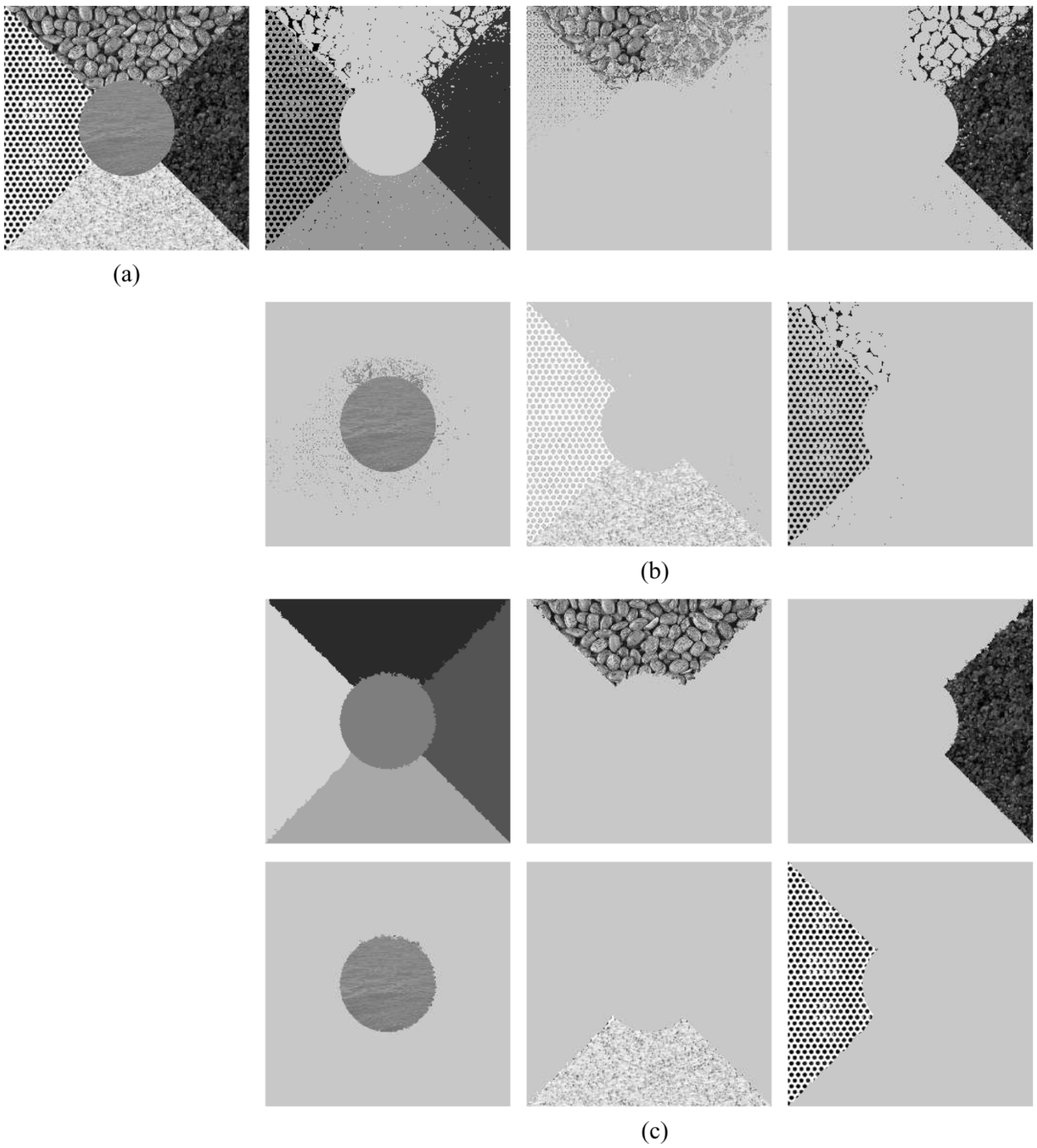


Fig. 18. Normalized cut segmentation result and its major components for a texture image. (a) Input image as shown in Fig. 7(a). (b), (c) Segmentation result and major segmented components of normalized cut and the proposed method respectively. Here, a distinctive gray color is used to indicate pixels not in the current component.

While our method can be applied to segmentation of texture as well as nontexture regions, we demonstrate here that this generality does not lead to less accurate segmentation for texture images through an additional comparison for texture segmentation [6]. To improve texture recognition and segmentation, Clausi and Deng [6] fuse Gabor features and gray-level co-occurrence probability features so that lower, mid, and higher frequency texture information is all captured. The fused features, as well as Gabor and gray-level co-occurrence probability features, are applied to segmentation using the standard K-means clustering method. They provide quantitative results on three texture images (shown in Fig. 19) and Table I shows the accuracy of best segmentation results from [6]. We apply our method on the images and our segmentation results (without and with boundary localization) are shown in the last two columns of Fig. 19, the accuracy of which is summarized in Table I; here, the segmentation accuracy is computed relative to the provided ground truth segmentation. The same eight filters used earlier are applied to Fig. 19(a) and (b), while only the intensity and two gradient filters are applied to Fig. 19(c) due to the presence of narrow-shaped parts in some texture regions. The comparison shows that our boundary localization improves boundary accuracy and our results with boundary localization are more accurate in all the cases. We attribute the improvement to the use of iterative feature extraction and segmentation, a key difference between clustering-based methods and ours.

C. *Further Evaluations*

As texture is a board perceptual concept, a given texture segmentation method likely works well on some textures but not well on others, depending the features and the parameters used. Important aspects of a segmentation method include whether it is applicable to different kinds of images and how sensitive it is to particular choices of parameter values. To answer these questions, we have performed further quantitative analysis with respect to two important aspects of our method:

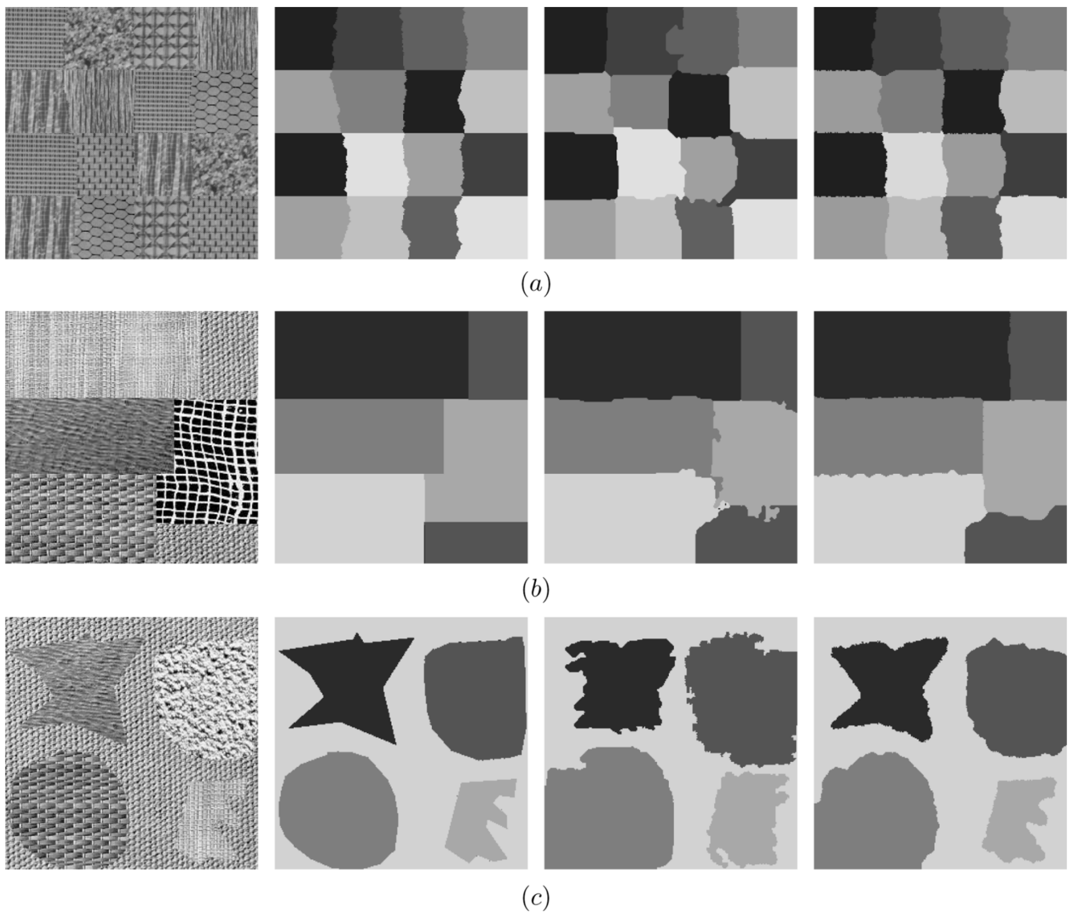


Fig. 19. Results on texture images used in [6]. In each row, the left image shows the original texture image, the middle-left shows the ground truth segmentation, and the right two images show our segmentation results without and with boundary localization respectively. (a) Image used in [3], consisting of seven textures and 16 regions. Note that the boundaries are not straight lines in the ground truth. (b) Image with six regions of different sizes. (c) Image with irregular-shaped regions.

TABLE I
SEGMENTATION ACCURACY (%) FOR THE FEATURE FUSION [6] AND THE PROPOSED METHODS

Image	Best (%) from [6]		Proposed method (%)	
	without fusion	with fused features	without localization	with localization
Fig. 19(a)	87.2	84.9	90.8	97.4
Fig. 19(b)	82.5	90.3	94.8	96.3
Fig. 19(c)	89.5	93.6	84.8	94.2

TABLE II
SEGMENTATION ACCURACY (%) FOR THE PROPOSED METHOD IN FIG. 7(a) USING DIFFERENT INTEGRATION SCALES

Integration scale	Accuracy (%)	
	without localization	with localization
43×43	93.7	98.8
35×35	93.4	99.1
29×29	94.6	99.1
23×23	94.7	99.1
19×19	96.1	99.2
15×15	96.2	98.5
13×13	93.4	99.1

the integration scale and the filters chosen. The integration scale corresponds to the window size used to compute local spectral histogram features. In general, the larger the window, the larger the uncertainty along region boundaries. The use of an asymmetric window in our method reduces the uncertainty by limiting the influence of pixels from other texture regions (see Fig. 6). On the other hand, when the window size is too small, the variation within the same texture type becomes large, leading to inaccurate segmentation. To show this, we apply our method on the image in Fig. 7(a) by using different integration scales. Table II summarizes the results, which show an "optimal" integration scale for the image without boundary localization. Our boundary localization algorithm, however, largely eliminates this sensitivity by using spatial patterns. This shows that our method is not sensitive to particular choices of integration scale.

Regarding the filters used in computing local spectral histograms, they are clearly important for segmentation results but

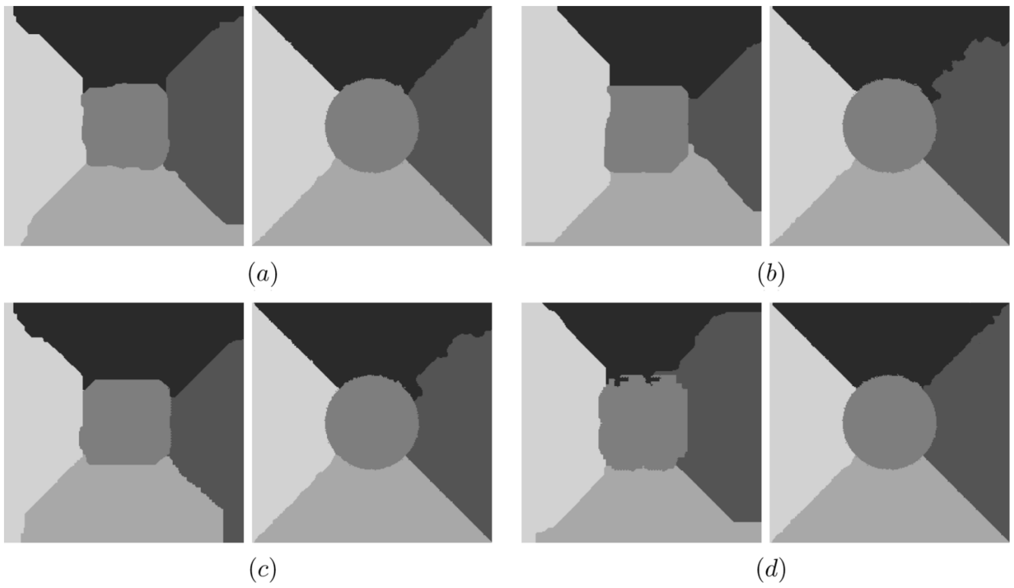


Fig. 20. Segmentation results using different filters on the image shown in Fig. 7(a). In each panel, the result without and with boundary localization is shown on the left and right, respectively. (a) Intensity filter. (b) Two gradient filters. (c) Two LoG filters. (d) Three Gabor filters.

TABLE III
SEGMENTATION ACCURACY (%) FOR THE PROPOSED METHOD IN FIG. 7(a) USING DIFFERENT COMBINATIONS OF THE EIGHT FILTERS

Filter(s)	Accuracy (%)	
	without localization	with localization
Intensity filter	95.2	99.2
Two gradient filters	82.4	97.8
Intensity + two gradient filters	94.4	98.9
Two LoG filters	91.4	98.0
Three Gabor filters	88.9	99.3
Two LoG + three Gabor filter	93.2	98.7
Intensity + two LoG filter	94.3	99.2
Two gradient + three Gabor filters	90.9	98.7

a specific choice of filters is not critical because of the integration in the spectral histogram computation and the boundary localization. To demonstrate this, again we apply our method on the image shown in Fig. 7(a) with integration scale of 35×35 but with different filters. Table III summarizes the results and selected segmentation results are shown in Fig. 20. As the results show, there is an "optimal" set of filters for this image without boundary localization. With boundary localization, the difference in performance by using different sets of filters is significantly reduced. Similar to the situation with the integration scale, our method allows us to achieve accurate segmentation result with different sets of filters. This shows that the choice of filters is not critical in our method.

VII. CONCLUSION

In this paper, we have presented a new segmentation method for images consisting of texture, as well as nontexture regions using local spectral histograms. By decomposing the algorithm into three stages, we derive probability models and couple feature extraction with segmentation through iterative updating. These lead to more accurate region boundaries, which are further localized through a localization stage. Comparisons with other methods show that our method gives more accurate segmentation.

Here, we assume filters and the integration scale for segmentation are given. While a fixed setting works sufficiently well for different kinds of images used in this paper and our method appears not very sensitive to particular choices of integration scale and filters, adaptive filters and integration scales for different regions could further improve segmentation results. Future research needs to address the issue of automatically choosing filters and integration scales to achieve optimal segmentation of given images.

ACKNOWLEDGMENT

The authors would like to thank three anonymous reviewers whose comments have improved the presentation of this paper and Dr. D. Clausi at the University of Waterloo for making the original and ground truth images in Fig. 19 available to them.

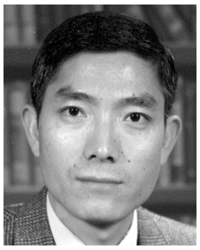
REFERENCES

[1] R. Azencott, J. P. Wang, and L. Younes, "Texture classification using windowed fourier filters," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 19, no. 2, pp. 148–153, Feb. 1997.
[2] J. R. Bergen and E. H. Adelson, "Early vision and texture perception," *Nature*, vol. 333, pp. 363–367, 1988.
[3] J. Bigun and J. M. du Buf, "N-folded symmetries by complex moments in Gabor space and their application to unsupervised image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 16, no. 1, pp. 80–87, Jan. 2004.
[4] J. Canny, "A computational approach to edge detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-8, no. 6, pp. 679–698, Jun. 1986.

[5] T. F. Chan and L. A. Vese, "A level set algorithm for minimizing the Mumford-Shah functional in image processing," in *Proc. IEEE Workshop on Variational and Level Set Methods in Computer Vision*, 2001, pp. 161–168.
[6] D. A. Clausi and H. Deng, "Design-based texture feature fusion using Gabor filters and co-occurrence probabilities," *IEEE Trans. Image Process.*, vol. 14, no. 7, pp. 925–936, Jul. 2005.
[7] N. Fusco, "An overview of the Mumford-Shah problem," presented at the Mathematical Models of Visual Perception 2004.
[8] D. J. Heeger and J. R. Bergen, "Pyramid-based texture analysis/synthesis," in *Proc. SIGGRAPH*, 1995, pp. 229–238.
[9] T. Hofmann, J. Puzicha, and J. M. Buhmann, "Unsupervised texture segmentation in a deterministic annealing framework," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 20, no. 8, pp. 803–818, Aug. 1998.
[10] X. Liu, Computational Investigation of Feature Extraction and Image Organization The Ohio State Univ., Columbus, 1999, Ph.D. dissertation.
[11] X. Liu and L. Cheng, "Independent spectral representations of images for recognition," *J. Opt. Soc. Amer. A*, vol. 20, no. 7, pp. 1271–1282, 2003.
[12] X. Liu and D. L. Wang, "A spectral histogram model for texton modeling and texture discrimination," *Vis. Res.*, vol. 42, no. 23, pp. 2617–2634, 2002.
[13] ——, "Texture classification using spectral histograms," *IEEE Trans. Image Process.*, vol. 12, no. 6, pp. 661–670, Jun. 2003.
[14] D. Marr, *Vision*. New York: W. H. Freeman, 1982.
[15] J. M. Morel and S. Solimini, *Variational Methods for Image Segmentation*. Boston, MA: Birkhauser, 1995.
[16] D. Mumford and J. Shah, "Optimal approximations of piecewise smooth functions and associated variational problems," *Commun. Pure Appl. Math.*, vol. XLII, no. 4, pp. 577–685, 1989.
[17] T. Randen and J. H. Husoy, "Filtering for texture classification: A comparative study," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 21, no. 4, pp. 291–310, Apr. 1999.
[18] B. Schiele and J. L. Crowley, "Recognition without correspondence using multidimensional receptive field histograms," *Int. J. Comput. Vis.*, vol. 36, pp. 31–50, 2000.
[19] H. Schneiderman and T. Kanade, "Object detection using the statistics of parts," *Int. J. Comput. Vis.*, vol. 56, pp. 151–177, 2004.
[20] J. Shi and J. Malik, "Normalized cut and image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 8, pp. 888–905, Aug. 2000.
[21] S. Ullman, *High-level Vision*. Cambridge, MA: MIT Press, 1996.
[22] P. Viola and M. J. Jones, "Robust real-time face detection," *Int. J. Comput. Vis.*, vol. 57, no. 2, pp. 137–154, 2004.
[23] G. Wall, F. Iqbal, J. Isaacs, X. Liu, and S. Foo, "Real-time texture classification using FPGAs," in *Proc. Applied Imagery and Pattern Recognition Workshop*, 2004, pp. 130–135.
[24] D. L. Wang and D. Terman, "Image segmentation based on oscillatory correlation," *Neural Comput.*, vol. 9, pp. 805–836, 1997.
[25] C. Waring and X. Liu, "Face detection using spectral histograms and SVM," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 35, no. 3, pp. 467–476, Mar. 2005.
[26] S. C. Zhu and A. Yuille, "Region competition: unifying snakes, region growing, and Bayes/MDL for multiband image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 18, no. 8, pp. 884–900, Aug. 1996.
[27] S. C. Zhu, X. Liu, and Y. Wu, "Exploring Julesz ensembles by efficient Markov chain monte carlo," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 6, pp. 554–569, Jun. 2000.
[28] S. C. Zhu, Y. N. Wu, and D. Mumford, "Minimax entropy principle and its application to texture modeling," *Neural Comput.*, vol. 9, pp. 1627–1660, 1997.

Xiuwen Liu (S'97–A'99–M'02) received the B.Eng. degree in computer science in 1989 from Tsinghua University, Beijing, China, the M.S. degrees in geodetic science and surveying and computer and information science in 1995 and 1996, respectively, and the Ph.D. degree in computer and information science in 1999 from The Ohio State University, Columbus.

From August 1989 to February 1993, he was with the Department of Computer Science and Technology, Tsinghua University. Since 2000, he has been with the Department of Computer Science, Florida State University, Tallahassee. His current research interests include low-dimensional representations of images, statistical shape and surface modeling, optimization and inference algorithms on manifolds, computational modeling of visual perception and inference, real-time vision algorithms and implementations, and general areas of computer vision and pattern recognition.



DeLiang Wang (M'90–SM'01–F'04) received the B.S. degree in 1983 and the M.S. degree in 1986 from Peking (Beijing) University, Beijing, China, and the Ph.D. degree from the University of Southern California, Los Angeles, CA, in 1991, all in computer science.

From July 1986 to December 1987, he was with the Institute of Computing Technology, Academia Sinica, Beijing, China. Since 1991, he has been with the Department of Computer Science and Engineering and the Center for Cognitive Science, The Ohio State University, Columbus, where he is currently a Professor. From October 1998 to September 1999, he was a Visiting Scholar in the Department of Psychology, Harvard University, Cambridge, MA. His research interests include machine perception and neurodynamics.

Dr. Wang is the President of the International Neural Network Society (for 2006) and a recipient of the 1996 U.S. Office of Naval Research Young Investigator Award.