Pattern Recognition Letters 33 (2012) 615–622

Contents lists available at SciVerse ScienceDirect

Pattern Recognition Letters

journal homepage: www.elsevier.com/locate/patrec

Image segmentation using local spectral histograms and linear regression

Jiangye Yuan[a,*], DeLiang Wang[b], Rongxing Li[a]

[a] Mapping and GIS Laboratory, Department of Civil and Environmental Engineering and Geodetic Science, Columbus, OH 43210, United States
[b] Department of Computer Science and Engineering, Center for Cognitive Science, The Ohio State University, Columbus, OH 43210, United States

ARTICLE INFO

Article history:
Received 2 June 2011
Available online 17 December 2011
Communicated by S. Wang

Keywords:
Texture segmentation
Spectral histogram
Linear regression

ABSTRACT

We present a novel method for segmenting images with texture and nontexture regions. Local spectral histograms are feature vectors consisting of histograms of chosen filter responses, which capture both texture and nontexture information. Based on the observation that the local spectral histogram of a pixel location can be approximated through a linear combination of the representative features weighted by the area coverage of each feature, we formulate the segmentation problem as a multivariate linear regression, where the solution is obtained by least squares estimation. Moreover, we propose an algorithm to automatically identify representative features corresponding to different homogeneous regions, and show that the number of representative features can be determined by examining the effective rank of a feature matrix. We present segmentation results on different types of images, and our comparison with other methods shows that the proposed method gives more accurate results.

1. Introduction

The goal of image segmentation is to partition an image into a number of regions so that each region should be as homogeneous as possible and neighboring ones should be as different as possible. It is a critical task for a wide range of image analysis problems. Although substantial progress has been made in this area (Shi and Malik, 2000; Chan and Vese, 2001; Comaniciu and Meer, 2002; Felzenszwalb and Huttenlocher, 2004; Sharon et al., 2006; Li et al., 2008; Wang et al., 2009), image segmentation remains an unsolved problem in computer vision.

An image segmentation method should work on different kinds of imagery including texture and nontexture images. It is widely recognized that a visual texture is very difficult to characterize. A large number of methods have been proposed to deal with texture images. These methods address two issues: underlying texture model that defines region homogeneity and a framework or strategy for producing segmentation (Paragios and Deriche, 2002; Deng and Clausi, 2004; Kim and Kang, 2007; Lehmann, 2011). In general the two issues are not treated independently. A successful segmentation methodology should couple an accurate texture model and an effective segmentation strategy.

Numerous texture models have been proposed. The most successful ones focus on the following two aspects. One is on filtering, which typically uses filterbanks to decompose an image into a set of sub-bands. Filtering methods have received experimental supports on human texture perception and have shown impressive performance for texture segmentation and classification (Jain and Farrokhsia, 1991; Dunn and Higgins, 1991; Randen and Hakon, 1999; Clausi and Jernigan, 2000). The other is on statistical modeling, which characterizes texture regions as resulting from some underlying stochastic processes. For example, autoregressive models (Mao and Jain, 1992) and Markov random fields (Cross and Jain, 1983; Cesmeli and Wang, 2001; Benboudjema and Pieczynski, 2007) emphasize global appearance and are robust to noise.

Building on the above themes, a local spectral histogram has been proposed as a texture model, which consists of marginal distributions of chosen filter responses in an image window (Liu and Wang, 2002). Local spectral histograms provide a generic statistic model for texture as well as nontexture regions. Using local spectral histograms as features, the segmentation problem can be approached by measuring the distance among the features (Liu and Wang, 2006). However, since the local spectral histograms computed over the windows straddling boundaries do not give distinctive features, such methods have difficulty in accurately localizing region boundaries. As Malik et al. (2001) point out, this problem widely exists in the approaches based on measuring texture descriptors over local windows. To address this problem, quadrant filters or similar strategies are often employed, which compute features from shifted local windows around a pixel and make a best choice by examining certain criteria (Kim and Kang, 2007; Liu and Wang, 2006). Another popular technique is to use local windows of different sizes, also referred to as scales

* Corresponding author. Tel.: +1 614 214 0686.
E-mail address: yuan.75@osu.edu (J. Yuan).

0167-8655/$ - see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.patrec.2011.12.003

(Liang and Tjahjadi, 2006; Martin et al., 2004). Boundaries are determined by analyzing information across different scales. Although reasonable results are achieved, these methods inevitably involve more computation and hand chosen parameters.

In this paper, we propose a new segmentation method that leverages local spectral histograms to discriminate region appearances and at the same time accurately localizes boundaries. Using local spectral histograms as features, we regard a pixel location as a linear combination of representative features, which encodes a natural criterion to identify boundaries. The relationship between features of all the pixels and representative features is modeled by a multivariate linear regression, and the segmentation problem can be directly solved by least square estimation. The rest of the paper is organized as follows. The local spectral histogram representation is introduced in Section 2. Section 3 presents our segmentation algorithm in detail. In Section 4, we show experimental results and comparisons. Section 5 concludes the paper.

2. Local spectral histograms

Motivated by perceptual observations, the spectral histogram model has been proposed to characterize texture appearance (Liu and Wang, 2002). For a window $\boldsymbol{W}$ in an input image, a set of filter responses is computed through convolving with a chosen bank of filters $\{F^{(\alpha)}, \alpha = 1, 2, \ldots, K\}$. For a sub-band image $\boldsymbol{W}^{(\alpha)}$, a bin of its histogram can be written as

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

Here z_1 and z_2 specify the range of the bin. $\vec{v}$ represents a pixel location, and δ denotes the Dirac delta function. In this paper, we use 11 equal-width bins for each filter response. The spectral histogram with respect to the chosen filters is then defined as (Liu and Wang, 2002)

$$H_W = \frac{1}{|\boldsymbol{W}|}\left(H_W^{(1)}, H_W^{(2)}, \ldots, H_W^{(K)}\right) \tag{2}$$

where $|\,|$ denotes cardinality. The spectral histogram is a normalized feature statistic, which can compare image windows of different sizes. For each pixel location, the local spectral histogram is computed over the window centered at the pixel. The size of the window is called integration scale. When the filters are selected properly, the spectral histogram is sufficient to capture texture appearance (Liu and Wang, 2002).

In this paper, we use seven filters: the intensity filter, two LoG (Laplacian of Gaussian) filters with the scale values of 0.2 and 1.0, and four Gabor filter with the orientations of 0°, 45°, 90°, and 135° and the scale value of 3. The parameters of filters are not adjusted for individual images, but for a type of images a set of filters is chosen based on general image attributes.

In order to extract meaningful texture features, the integration scale is set to be relatively large, which makes computing local spectral histograms computationally expensive. A fast implementation method is therefore introduced in (Liu and Wang, 2002). For an input image, an integral histogram image is defined as follows: at location (x, y) the integral histogram is calculated using the pixel values above and to the left of (x, y) (see Fig. 1(a)). The integral histogram image can be efficiently computed in one pass over the image. Given the integral histogram image, the histograms of arbitrary rectangular regions can be obtained with four references. As illustrated in Fig. 1(b), we can compute the histogram of region R using the following four references: $L_4 + L_1 - L_2 - L_3$. Hence, once the integral histogram image is computed, we only need three vector arithmetic operations to obtain any local spectral histogram regardless of window size. A detailed description of the fast implementation can be found in (Liu and Wang, 2002).

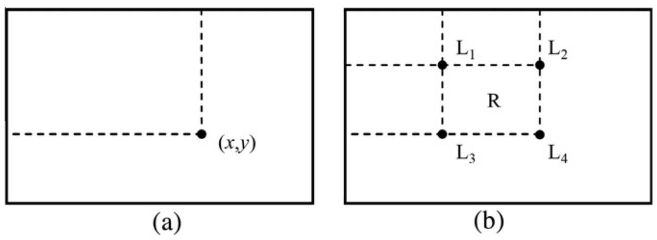

Fig. 1. Illustration of fast implementation for computing local spectral histograms. (a) The integral histogram value at location (x, y) is the histogram of the image window above and to the left of (x, y). (b) The histogram of region R can be computed using four references: $L_4 + L_1 - L_2 - L_3$.

3. Segmentation algorithm

As discussed in Section 1, although local spectral histograms provide an effective feature, segmentation methods using feature distance to measure region homogeneity tend to produce inaccurate boundaries caused by features extracted in image windows that cross multiple regions. Here, we describe a new segmentation algorithm based on linear regression, which can produce segmentation with high accuracy and great efficiency.

3.1. Segmentation using linear regression

Fig. 2(a) illustrates the difficulty of extracting features over a large window. Here, an image contains five homogenous regions. Given a pixel location A, the corresponding local spectral histogram is computed using a square window. Since this window straddles two different regions, the extracted feature is not discriminative. As a result, it is difficult to correctly classify the corresponding pixel by measuring feature similarity.

Let us define a window $\boldsymbol{W}$ consisting of disjoint connected subregions $\{\boldsymbol{W}_1, \boldsymbol{W}_2, \ldots, \boldsymbol{W}_s\}$, and $H_{W_i}^{(\alpha)}$ is the histogram computed from region $\boldsymbol{W}_i$ and filter α. Since $H_W^{(\alpha)} = H_{W_1}^{(\alpha)} + H_{W_2}^{(\alpha)} + \cdots + H_{W_s}^{(\alpha)}$, we can rewrite the spectral histogram H_W as

$$\begin{aligned}
H_W &= \frac{1}{|\boldsymbol{W}|}\left(H_W^{(1)}, H_W^{(2)}, \ldots, H_W^{(K)}\right) \\
&= \frac{1}{|\boldsymbol{W}|}\left(\sum_{i=1}^{s} H_{W_i}^{(1)}, \sum_{i=1}^{s} H_{W_i}^{(2)}, \ldots, \sum_{i=1}^{s} H_{W_i}^{(K)}\right) \\
&= \frac{|\boldsymbol{W}_1|}{|\boldsymbol{W}|}\left(\frac{1}{|\boldsymbol{W}_1|}\left(H_{W_1}^{(1)}, H_{W_1}^{(2)}, \ldots, H_{W_1}^{(K)}\right)\right) \\
&\quad + \frac{|\boldsymbol{W}_2|}{|\boldsymbol{W}|}\left(\frac{1}{|\boldsymbol{W}_2|}\left(H_{W_2}^{(1)}, H_{W_2}^{(2)}, \ldots, H_{W_2}^{(K)}\right)\right) + \cdots \\
&\quad + \frac{|\boldsymbol{W}_s|}{|\boldsymbol{W}|}\left(\frac{1}{|\boldsymbol{W}_s|}\left(H_{W_s}^{(1)}, H_{W_s}^{(2)}, \ldots, H_{W_s}^{(K)}\right)\right).
\end{aligned}$$

With the definition in (2), we have

$$H_W = \frac{|\boldsymbol{W}_1|}{|\boldsymbol{W}|}H_{W_1} + \frac{|\boldsymbol{W}_2|}{|\boldsymbol{W}|}H_{W_2} + \cdots + \frac{|\boldsymbol{W}_s|}{|\boldsymbol{W}|}H_{W_s}. \tag{3}$$

Therefore, a spectral histogram of an image window can be linearly decomposed into spectral histograms of its subregions, where weights are proportional to region areas.

Because spectral histograms can characterize image appearance, we assume that spectral histograms within a homogeneous region are approximately constant. This assumption along with Eq. (3) implies that the spectral histogram of a local window can be approximated by a weighted sum of the spectral histograms

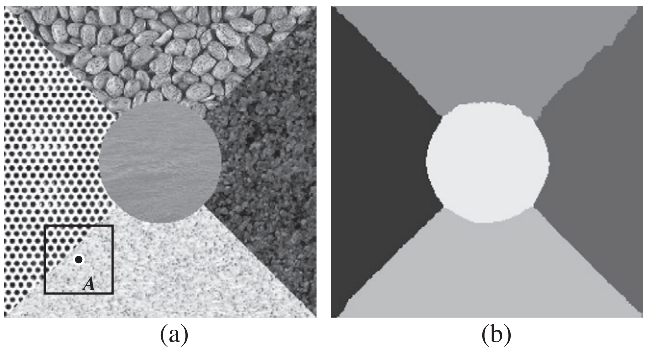

Fig. 2. Texture image segmentation via linear regression. (a) Texture image with size 256 × 256. The local spectral histogram at location A is computed within the square window. (b) Segmentation result using linear regression. Each region is represented by a distinct gray value.

of regions overlapping that window, and the corresponding pixel can be classified into the region whose spectral histogram weighs the most. For instance, in Fig. 2(a), the local spectral histogram at location A can be decomposed into the spectral histograms of two neighboring regions, and A can be segmented according to the weights.

The above analysis is not valid in some scenarios: (1) subregions within an image window are too small to obtain meaningful histograms; and (2) filter scales are so large as to cause distorted histograms in near-boundary subregions, making them different from the constant spectral histograms of the regions that include the subregions. However, both scenarios should have a minimal impact for the following reasons. For the first scenario, if a subregion within a window is small, it contributes little to the spectral histogram and thus can be neglected in feature decomposition. While it is possible that a window consists of many small subregions, such scenarios do not occur often in an image. For the second scenario, because the purpose of filtering in a spectral histogram is to capture a local spatial pattern, the chosen filters should not have large scales.

For each homogeneous region in an image, we define a representative feature, which should be equal to the constant spectral histogram of the region. By extending the above analysis, we regard the local spectral histogram of a pixel location as a linear combination of all representative features weighted by the area coverage of each feature. Consequently, we can use a multivariate linear regression model to associate each feature to the representative features. Given an image with N pixels, feature dimensionality of M, and L representative features, the model can be expressed as

$$\boldsymbol{Y} = \boldsymbol{Z}\boldsymbol{\beta} + \boldsymbol{\varepsilon} \tag{4}$$

where $\boldsymbol{Y}$ is an $M \times N$ matrix whose columns are feature vectors of all the pixel locations, $\boldsymbol{Z}$ is an $M \times L$ matrix whose columns are representative features, and $\boldsymbol{\beta}$ is an $L \times N$ matrix whose columns are weight vectors. $\boldsymbol{\varepsilon}$ is an error term.

Since the feature matrix $\boldsymbol{Y}$ and the representative feature set $\boldsymbol{Z}$ are known, the segmentation problem boils down to estimating $\boldsymbol{\beta}$ that best models the relationship between the feature matrix and the representative features. The least squares estimate of $\boldsymbol{\beta}$ is given by Johnson and Wichern (2007)

$$\hat{\boldsymbol{\beta}} = (\boldsymbol{Z}^T\boldsymbol{Z})^{-1}\boldsymbol{Z}^T\boldsymbol{Y}. \tag{5}$$

One can manually select windows at the centers of homogeneous regions to calculate representative features, and a segmentation result is provided by examining $\hat{\boldsymbol{\beta}}$ – each pixel is assigned to the class where the corresponding representative feature has the largest weight. Thus, the segmentation is completed through simple matrix operations. Fig. 2(b) shows the segmentation result of the image in Fig. 2(a), where different segments are indicated using distinct gray values. By counting the wrongly segmented pixels, the segmentation error rate in this case is only 1.1%.

Certain constraints should be imposed on least squares solutions. For example, the sum of the weight vector of each pixel should be equal to one, and the weight vector should be nonnegative. There are solutions for least squares estimates satisfying these constraints (Kay, 1993). However, for the sake of simplicity and computational efficiency, we take the unconstrained solution in Eq. (5), which gives satisfactory results in our experiments.

3.2. Automatic representative feature selection

In the segmentation algorithm presented above, representative features are assumed to be manually given. We present an algorithm to automatically select representative features corresponding to homogeneous regions. The goal is to find such features that each of them is close to a subset of all the features and they are as different as possible from each other. A straightforward method is to perform k-means clustering on all the features and take cluster centers as representative features. However, the features near boundaries can form small clusters, which make the clustering result unstable. To address this problem, we compute image gradients based on local spectral histograms and exclude the features associated with large gradients from the clustering process. Because the remaining features are expected to be inside homogenous regions, they should yield a stable clustering result.

At a pixel location (x,y), we compute the feature distance between pixel locations $(x + d,y)$ and $(x - d,y)$ and that between $(x,y + d)$ and $(x,y - d)$, where d is half of side length of $\boldsymbol{W}$. The gradient is the sum of the two distances (Liu and Wang, 2002). We employ χ^2-statistics to measure the distance, which is commonly used with histograms,

$$\chi^2(H_{W_1}, H_{W_2}) = \frac{1}{|\boldsymbol{W}|}\sum_{\alpha=1}^{K}\sum_{z}\frac{\left(H_{W_1}^{(\alpha)}(z) - H_{W_2}^{(\alpha)}(z)\right)^2}{H_{W_1}^{(\alpha)}(z) + H_{W_2}^{(\alpha)}(z)}. \tag{6}$$

This distance measure is also used in k-means clustering. It should be noted that the particular form of distance measure is not critical for spectral histograms. Other distance measures can also achieve comparable results.

Fig. 3(a) shows the gradient of the image in Fig. 2(a), where the gray value is proportional to the gradient. One can see that the pixels near boundaries have large gradients. By applying k-means clustering to the features with small gradients, we obtain cluster centers as the representative features. Fig. 3(b) shows the segmentation result

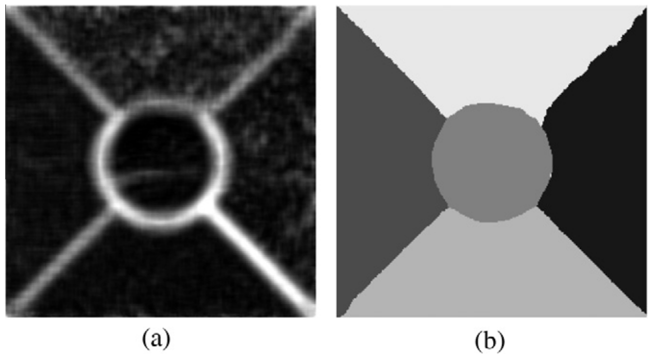

Fig. 3. Segmentation using automatically selected features. (a) Texture gradient of the image shown in Fig. 2(a). (b) Segmentation result with the representative features selected automatically.

using the resulting representative features, which is similar to the result in Fig. 2(b). Generally speaking, because automatically selected features pool over populations of features, they can be more representative and produce better segmentation results than manually selected features.

3.3. Segment number determination

A thorny issue in k-means clustering is how to determine the cluster number, i.e. the segment number in our case. Note that our method puts pixels with similar features into one segment without considering the connectivity of each segment, which, however, can be achieved via postprocessing. Here, we present a method to determine the segment number.

For a unique solution in Eq. (4) to exist, it requires that $\boldsymbol{Z}$ have the full column rank. In other words, representative features have to be linearly independent in order to have a unique segmentation solution. Since each feature is considered to be a linear combination of representative features, the rank of the feature matrix $\boldsymbol{Y}$ should equal the rank of $\boldsymbol{Z}$, which is exactly the number of representative features, or the segment number. Although in real cases the rank of $\boldsymbol{Y}$ tends to be larger than the segment number due to image noise, we can estimate the segment number by determining the effective rank of $\boldsymbol{Y}$.

We employ singular value decomposition (SVD) to determine the effective rank of a matrix (Konstaintinides and Yao, 1988). The feature matrix can be decomposed into the following form

$$\boldsymbol{Y} = U\Sigma\boldsymbol{V}^T = \begin{pmatrix} u_{11} & \cdots & u_{1M} \\ \vdots & \ddots & \vdots \\ u_{M1} & \cdots & u_{MM} \end{pmatrix} \begin{pmatrix} \sigma_{11} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \sigma_{MM} \end{pmatrix} \begin{pmatrix} v_{11} & \cdots & v_{1N} \\ \vdots & \ddots & \vdots \\ v_{N1} & \cdots & v_{NN} \end{pmatrix}^T. \quad (7)$$

Here $\boldsymbol{U}$ and $\boldsymbol{V}$ are orthogonal matrices, where the columns of $\boldsymbol{U}$ are the eigenvectors of the matrix $\boldsymbol{YY}^T$, and the columns of $\boldsymbol{V}$ are the eigenvectors of the matrix $\boldsymbol{Y}^T\boldsymbol{Y}$. The diagonal terms of the matrix $\boldsymbol{\Sigma}$ are called singular values, which are the square roots of the eigenvalues of the matrix $\boldsymbol{YY}^T$, or $\boldsymbol{Y}^T\boldsymbol{Y}$. The effective rank can be determined as follows. First rank the singular values, and then discard those below a certain threshold. The number of remaining singular values is the effective rank.

We have computed the singular values for the image in Fig. 2(a). There are 33 singular values. The first 5 singular values are 219.0, 91.7, 63.3, 62.3, and 37.9. The remaining singular values decrease gradually from 16.6 to 0. Hence, the first 5 values are noticeably larger than the rest, agreeing with the number of regions in the original image.

A few comments should be made regarding this method. Based on the fact that $\boldsymbol{YY}^T$ and $\boldsymbol{Y}^T\boldsymbol{Y}$ have the same nonzero eigenvalues, we choose $\boldsymbol{YY}^T$ to compute the singular values, which is a much smaller matrix than $\boldsymbol{Y}^T\boldsymbol{Y}$. When the singular values are computed, we need to set a threshold to determine the effective rank. We should note that in some cases the singular values are not well separated, and the threshold need to be set with respect to the type of images.

3.4. Fast selection of representative features

Apart from estimating the segment number, the SVD of matrix $\boldsymbol{Y}$ provides a way to reveal the underlying structure of the matrix, based on which we propose a more efficient algorithm for extracting representative features.

Assuming the effective rank of $\boldsymbol{Y}$ is r, a new matrix can be obtained:

$$\boldsymbol{Y}_p = \boldsymbol{U}_p\boldsymbol{\Sigma}_p\boldsymbol{V}_p^T \quad (8)$$

where $\boldsymbol{U}_p$ and $\boldsymbol{V}_p$ consist of the first r columns of $\boldsymbol{U}$ and $\boldsymbol{V}$ in the SVD of matrix $\boldsymbol{Y}$, respectively. $\boldsymbol{\Sigma}_p$ is a $r \times r$ matrix with the largest r singular values on the diagonal. $\boldsymbol{Y}_p$ is the optimal r-rank approximation of matrix $\boldsymbol{Y}$ in the least-squares sense (Golub and Van Loan, 1996). Let $\boldsymbol{Z}_1 = \boldsymbol{U}_p$ and $\boldsymbol{\beta}_1 = \boldsymbol{\Sigma}_p\boldsymbol{V}_p^T$. $\boldsymbol{Z}_1$ and $\boldsymbol{\beta}_1$ can therefore serve as a least squares solution for the linear regression model in Eq. (4). However, they are generally not the real solutions, due to the fact that

$$\boldsymbol{Y}_p = \boldsymbol{Z_1\beta_1} = \boldsymbol{Z_1QQ^{-1}}\ \boldsymbol{\beta_1} \quad (9)$$

where $\boldsymbol{Q}$ can be any invertible square matrix.

Although the above analysis can not directly give representative features, it leads to an important finding: the representative features should be a linear transformation of $\boldsymbol{U}_p$, whose columns are orthogonal vectors. In other words, the representative features lie in an r-dimensional subspace spanned by the columns of $\boldsymbol{U}_p$. Based on the fact that the features inside homogeneous regions are close to the representative features, we project the features with small gradients onto the subspace and apply k-means clustering. Because the projection significantly reduces the feature dimension, the clustering speed is boosted. The representative features in the original space can be obtained by multiplying the r-dimensional clustering centers with the transpose of $\boldsymbol{U}_p$. Our experiments show that the fast algorithm gives segmentation results very similar to those from the original algorithm.

3.5. Steps of the algorithm

The steps of our image segmentation algorithm are summarized below:

1. Compute local spectral histograms at each pixel location to construct the feature matrix $\boldsymbol{Y}$.
2. Compute the SVD, $\boldsymbol{Y} = \boldsymbol{U\Sigma V}^T$. Determine the segment number r by thresholding the singular values.
3. Reduce the feature dimension by multiplying $\boldsymbol{Y}$ by $\boldsymbol{U}_p$, which is the first r columns of $\boldsymbol{U}$.
4. Compute the texture gradients based on the features.
5. Perform k-means clustering on the features with small gradients. The cluster centers are taken as the representative features.
6. Solve $\boldsymbol{\beta}$ using (5) to obtain the segmentation result.

3.6. Smoothing effect

The proposed method has a smoothing effect on region boundaries. Consider a two-region image containing a zigzag boundary. The proposed method will classify pixels according to the coverage of two regions within their local windows. If the integration scale is small, the resulting boundary would be close to the true boundary; with the integration scale sufficiently large, we would obtain a segmentation result with a straight boundary. Although the smoothing effect may cause deviation from the true boundary, it is interesting to note that the effect tends to reduce the total curvature of boundaries. In other words, the smoothing effect amounts to a form of regularization often formulated as an objective for image segmentation (Caselles et al., 1997; Mumford and Shah, 1989). This smoothing effect is apparent in our experiments with real images in Section 4.

Smoothness is controlled by integration scales. Ideally, one should choose an integration scale that exhibits some boundary smoothing but does not lead to over-smoothing. Different techniques could be considered, e.g., decomposition stability (Fukunaga, 1990), inter-cluster*versus* intra-cluster variability (Kauffman and Rousseeuw, 1990), or multiscale analysis. For instance, at each location in multiscale analysis one can compute spectral histograms at multiple integration scales. By analyzing feature variations and relationships across scales, an optimal local integration scale or a combination of multiple scales could be identified. Such analysis will be addressed in future work.

4. Experimental results and comparisons

4.1. Experimental results

We have applied our algorithm to different types of images. Fig. 4 shows the segmentation results for a set of texture images of size 340×340, including an image consisting of regions with different sizes, an image containing very similar textures, and an image with irregular-shaped regions. We use all seven filters to calculate local spectral histograms. For the experiments in this section, the singular value threshold and the integration scale are determined by analyzing a subset of a class of images. Here the integration scale is set to 25×25, and the singular value threshold 20. The left column shows the original images, the middle column ground truth segmentation, and the right column our segmentation results. As we can see, the regions with different textures are separated successfully, and the boundaries are localized well due to feature decomposition. The inaccuracy of the boundaries is mostly caused by similar texture appearances. Although texture appearance inside some regions varies noticeably, the segmentation results are not much affected. We attribute the robustness to both the texture features used and effective least square solutions.

To show the influence of the integration scale on segmentation results, we apply our algorithm to the image in Fig. 4(a) using four different integration scales, and the results are presented in Fig. 5. It can be observed that different integration scales affect segmentation results, and some values are more favorable than others. On a whole, the result is not very sensitive to a particular value, and similar results are obtained over a range of values. The results also demonstrate that the boundaries become increasingly smoother as the integration scale increases, in agreement with the discussion in Section 3.6.

We have also applied our method to a large number of natural images, including the Berkeley segmentation dataset (Kauffman and Rousseeuw, 1990). In general, orientation is not a strong cue for segmenting natural images. Hence, we only use the intensity filters and two LoG filters with different scales. Fig. 6 presents our segmentation results on a set of 9 images. It can be seen that both texture and nontexture regions are well segmented in general. For example, in the first (zebra) image the resulting segment by and large agrees with the true shape of the zebra, and in the per-

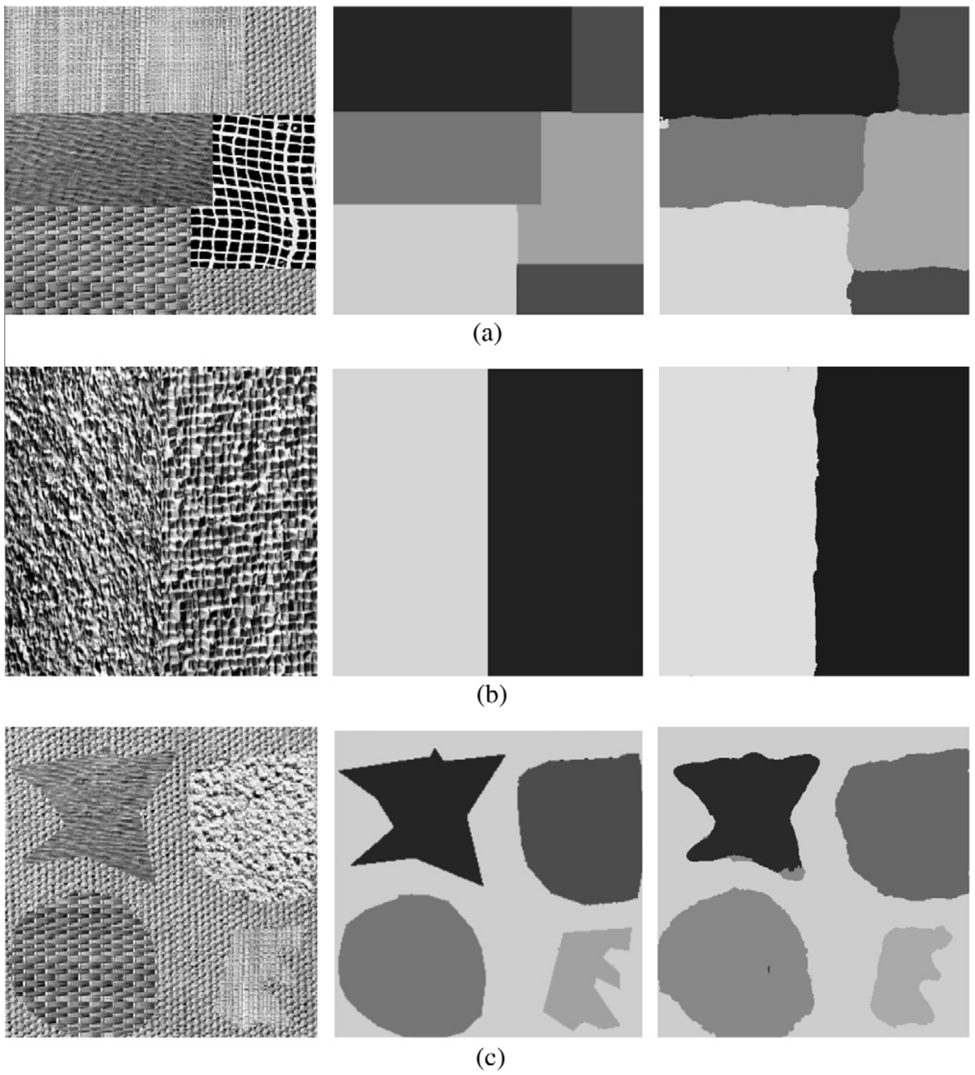

Fig. 4. Texture image segmentation. In each row, the left is the original image, the middle is the ground true, and the right is the segmentation result using our method. The integration scale used here is 25×25.

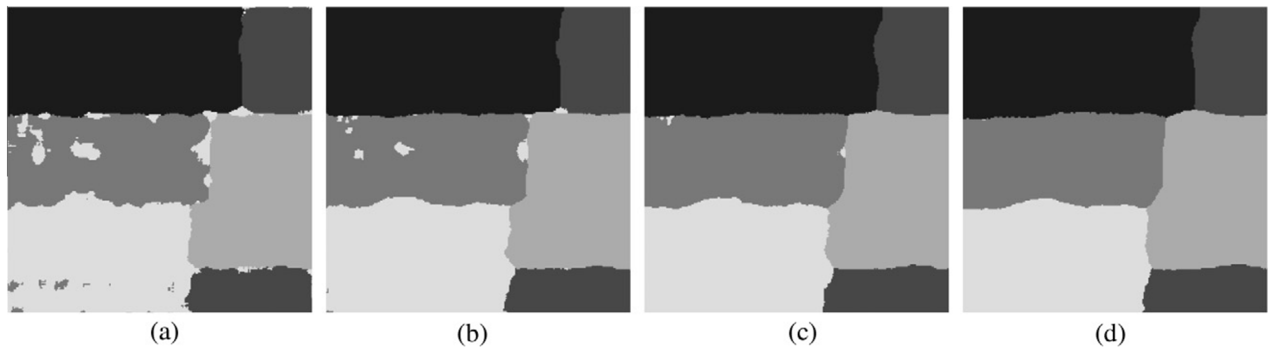
Fig. 5. Segmentation results using different integration scales. (a) 11 × 11. (b) 17 × 17. (c) 23 × 23. (d) 29 × 29.

son image the main components, including the eyes, the skin, and the shirt, are successfully segmented. These two images are frequently used in image segmentation literature. Without involving any object-specific models or human intervention, our results are comparable with the best results available (Michailovich et al., 2007; Kokkinos et al., 2009).

4.2. Systematic evaluation and comparisons

We compare the proposed method with another spectral histogram based segmentation algorithm proposed by Liu and Wang (2006). They build probability models based on local spectral histograms and iteratively update segmentation using the model, and the final result is obtained by a separate boundary localization algorithm. We refer to this method as the LW method. Fig. 7 shows the comparison results on two natural images. To show the accuracy of segmentation results, the boundaries of the salient objects are embedded in the original images. Both methods successfully segment the objects with well localized boundaries. A closer comparison, however, indicates that our results have much smoother boundaries, which should be more desirable to practical uses.

To provide a systematic assessment, we apply our algorithm to the test images in the Berkeley segmentation benchmark (Martin et al., 2001). For all the images, we use the intensity filter and

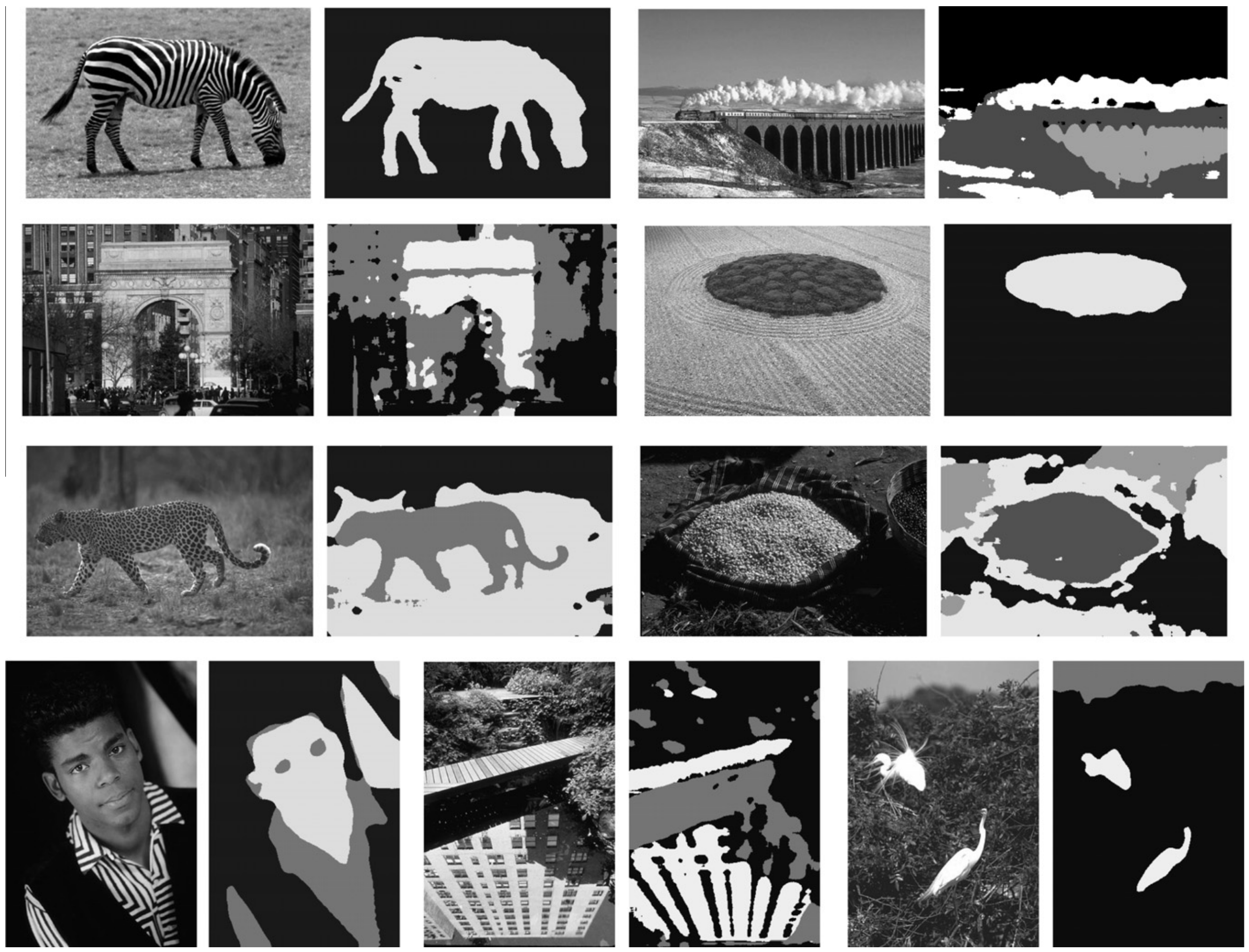
Fig. 6. Natural image segmentation. In each of the 9 pairs, the left is the original image, and the right is the segmentation result using our method. The integration scale used here is 15 × 15.

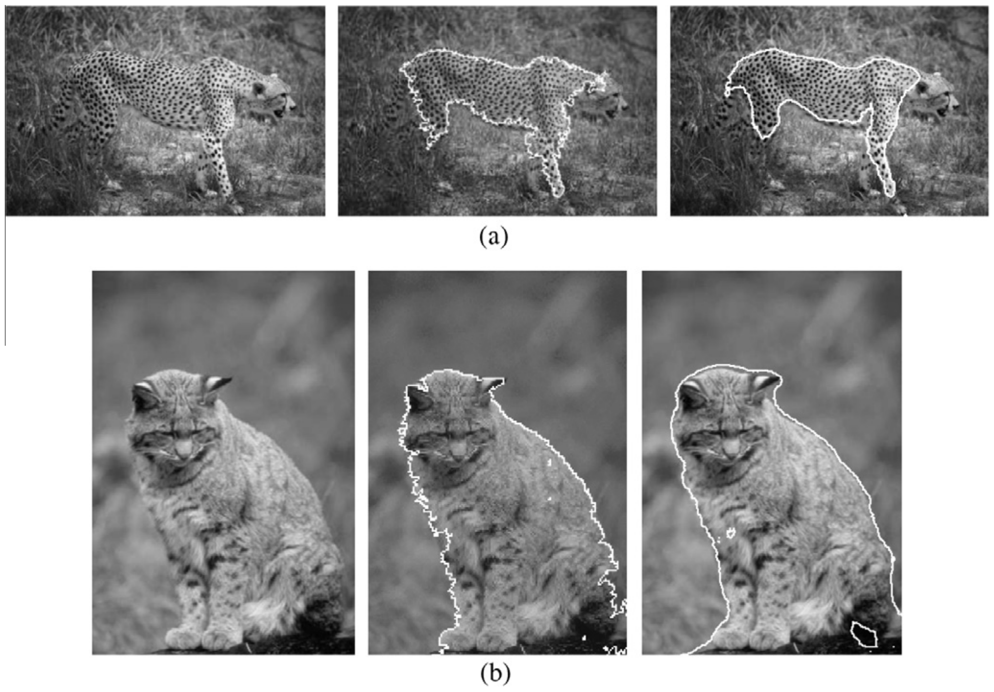

Fig. 7. Comparison between the LW method and proposed method on two natural images. In each panel, the left is the original image, the middle is the segmentation results from the LW method, and the right is the segmentation results from our method. The region boundaries are marked in white.

two LoG filters, and the integration scale is set to 15×15. We use the bidirectional consistency errors (BCE) (Martin, 2002) to quantitatively evaluate segmentation results. A local error at a pixel p_i between two segmentations S_a and S_b is defined as

$$E(S_a, S_b, p_i) = \frac{|R(S_a, p_i) \setminus R(S_b, p_i)|}{|R(S_a, p_i)|} \tag{10}$$

where $R(S,p_i)$ is the set of pixels that shares the same segment with pixel p_i in segmentation S, and \stands for set difference. Note that $E(S_a,S_b,p_i)$ is not equal to $E(S_b,S_a,p_i)$. Given a segmentation S_0 and a set of human segmentations $\{S_1,S_2,\ldots,S_H\}$, the BCE takes the following form

$$\mathrm{BCE}(S_0) = \frac{1}{n}\sum_{i=1}^{n} \min_h \{\max[E(S_0, S_h, p_i), E(S_h, S_0, p_i)]\} \tag{11}$$

which compares S_0 with all human segmentations and averages over n image pixels. Kokkinos et al. (2009) use the BCE to evaluate the performances of their method and the normalized cut algorithm (Shi and Malik, 2000) on the benchmark. They produce the segmentation results with different segment numbers and calculate the average and median values of BCE for each case. We follow the same evaluation scheme and summarize the results in Table 1. We can see that the proposed method outperforms the other two in most cases. The "optimal" column show the numbers calculated using the best result of each image. The comparison shows that our method gives the lowest BCE.

Table 1
Average/median values of BCE on 100 test images from different segmentation algorithms.

Author	Optimal	2 Segments	3 Segments	4 Segments
Proposed	.37/.38	.42/.41	.46/.48	.53/.55
Kokkinos et al.	.38/.39	.46/.49	.49/.51	.51/.52
N. Cuts	.41/.43	.49/.51	.52/.53	.55/.53

We have further applied precision-recall measurements (Martin et al., 2004) to the segmentation results of the benchmark images in order to assess the accuracy of segment boundaries. Precision is the proportion of correctly detected boundaries, and recall is the proportion of the boundaries in human segmentation that are detected. The harmonic mean of precision and recall, known as the *F*-measure, provides a single metric. Kokkinos et al. reported the *F*-measures of their results and the normalized cut results with multiple predefined segment numbers (Kokkinos et al., 2009), where the best scores are 0.5 and 0.42, respectively. For our method, the segment number is automatically determined by setting the singular value threshold to 60. The average *F*-measure of our results reaches 0.55, higher than their scores.

4.3. Computation time

The main steps in our algorithm, including filtering, spectral histogram computation, and least squares estimation, take linear time with respect to the number of pixels. In our case, we do not need complete SVD, only the eigenvalue decomposition of $\boldsymbol{YY}^T$ which is an $M \times M$ matrix (M is the feature dimension), and then choose the first several eigenvalues and the corresponding eigenvectors to construct $\boldsymbol{Z}_1$. For the features with 7 filters and 11-bin histograms, $\boldsymbol{YY}^T$ is a 77×77 matrix, and its eigenvalue decomposition takes negligible time. The computation time can vary for *k*-means clustering in representative feature selection, but the process is generally fast because the features are well grouped and can be projected onto a low dimensional subspace. We have implemented the whole system using Matlab. On 481×321 benchmark images, our algorithm takes several seconds to half a minute on a Pentium 1.7 GHz PC. With representative features provided beforehand, our algorithm generally takes a fraction of a second. Using the codes provided by the authors, the running times of the normalized cut algorithm and the LW algorithm on images of the same size are about 1 min and 5 min, respectively. While Kokkinos et al. do not report computation time in Kokkinos et al. (2009), their algorithm should have relatively high time complexity because it involves curve evolution using level set methods, which is known to be time-consuming. Compared to those algorithms, our algorithm is more efficient.

5. Conclusion

In this paper, we have developed a novel segmentation method for images consisting of texture and nontexture regions. We use local spectral histograms as features. Based on the observation that each feature can be approximated by linearly combining several representative features, we formulate the segmentation problem as a multivariate linear regression, where the solution is given by least squares estimation. We have also proposed algorithms to automatically select representative features and determine their numbers. Experiments show that our method gives more accurate segmentation than other methods.

Although in the current version we assume that filterbanks and integration scales are given for a type of images, fully autonomous segmentation requires techniques for filter and integration scale selection that are task-dependent. Adaptive filters and integration scales may be adopted, which should be suited to the spatial structure of an image. We are currently investigating these issues along with a multiscale treatment of the smoothing effect.

References

Benboudjema, D., Pieczynski, W., 2007. Unsupervised statistical segmentation of nonstationary images using triplet Markov fields. IEEE T. Pattern Anal. Machine Intell. 29, 1367–1378.

Caselles, V., Kimmel, R., Sapiro, G., 1997. Geodesic active contours. Internat. J. Comput.Vis. 22, 61–79.

Cesmeli, E., Wang, D.L., 2001. Texture segmentation using Gaussian–Markov random fields and neural oscillator networks. IEEE Trans. Neural Networks 12, 394–404.

Chan, T., Vese, L., 2001. Active contours without edges. IEEE T. Image Process. 10, 266–277.

Clausi, D.A., Jernigan, M.E., 2000. Designing Gabor filters for optimal texture separability. Pattern Recognit. 33, 1835–1849.

Comaniciu, D., Meer, P., 2002. Mean shift: A robust approach toward feature space analysis. IEEE T. Pattern Anal. Mach. Intell. 24, 603–619.

Cross, G.R., Jain, A.K., 1983. Markov random field texture models. IEEE T. Pattern Anal. Machine Intell. 5, 25–393.

Deng, H., Clausi, D.A., 2004. Unsupervised image segmentation using a simple MRF model with a new implementation scheme. Pattern Recognit. 37, 2323–2335.

Dunn, D., Higgins, W.E., 1991. Optimal Gabor filters for texture segmentation. IEEE T. Image Process. 4, 947–964.

Felzenszwalb, P., Huttenlocher, D., 2004. Efficient graph-based image segmentation. Internat. J. Comput.Vis. 59, 167–181.

Fukunaga, K., 1990. Introduction to Statistical Pattern Recognition, 2nd ed. Academic Press, New York.

Golub, G., Van Loan, C., 1996. Matrix Computations, 3rd ed. The Johns Hopkins University Press, Baltimore, MD.

Jain, A.K., Farrokhsia, R., 1991. Unsupervised texture segmentation using Gabor filters. Pattern Recognit. 24, 1167–1186.

Johnson, R.A., Wichern, D.W., 2007. Applied Multivariate Statistical Analysis, 6th ed. Pearson Prentice Hall, Upper Saddle River, NJ.

Kauffman, L., Rousseeuw, P., 1990. Finding Groups in Data: An Introduction to Cluster Analysis. John Wiley & Sons.

Kay, S.M., 1993. Fundamentals of Statistical Signal Processing: Estimation Theory. Pearson Prentice Hall, Upper Saddle River, NJ.

Kim, S.C., Kang, T.J., 2007. Texture classification and segmentation using wavelet packet frame and Gaussian mixture model. Pattern Recognit. 40, 1207–1221.

Kokkinos, L., Evangelopoulos, G., Maragos, P., 2009. Texture analysis and segmentation using modulation features, generative models, and weighted curve evolution. IEEE T. Pattern Anal. Machine Intell. 31, 142–157.

Konstaintinides, K., Yao, K., 1988. Statistical analysis of effective singular values in matrix rank determination. IEEE Trans. Acoust. Speech Signal Process. 36, 757–763.

Lehmann, F., 2011. Turbo segmentation of textured images. IEEE T. Pattern Anal. Machine Intell. 33, 16–29.

Liang, K.-H., Tjahjadi, T., 2006. Adaptive scale fixing for multiscale texture segmentation. IEEE T. Image Process. 15, 249–256.

Li, C., Kao, C., Gore, J.C., Ding, Z., 2008. Minimization of region-scalable fitting energy for image segmentation. IEEE T. Image Process. 17, 1940–1949.

Liu, X., Wang, D.L., 2002. A spectral histogram model for texton modeling and texture discrimination. Vis. Res. 42, 2617–2637.

Liu, X., Wang, D.L., 2006. Image and texture segmentation using local spectral histograms. IEEE T. Image Process. 15, 3066–3077.

Malik, J., Belongie, S., Leung, T., Shi, J., 2001. Contour and texture analysis for image segmentation. Internat. J. Comput. Vis. 43, 7–27.

Mao, J., Jain, A.K., 1992. Texture classification and segmentation using multiresolution simultaneous autoregressive models. Pattern Recognit. 25, 173–188.

Martin, D., 2002. An empirical approach to grouping and segmentation. University of California Berkeley, Ph.D. dissertation.

Martin, D., Fowlkes, C., Tal, D., Malik, J., 2001. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics, In: Proc. of Eighth Internat. Confer. on Computer Vision.

Martin, D., Fowlkes, C., Malik, J., 2004. Learning to detect natural image boundaries using local brightness, color, and texture cues. IEEE T. Pattern Anal. Machine Intell. 26, 530–549.

Michailovich, O., Rathi, Y., Tannenbaum, A., 2007. Image segmentation using active contours driven by the Bhattacharyya gradient flow. IEEE T. Image Process. 16, 2787–2801.

Mumford, D., Shah, J., 1989. Optimal approximations by piecewise smooth functions and associated variational problems. Commun. Pure Appl. Math. 21, 577–685.

Paragios, N., Deriche, R., 2002. Geodesic active regions and level set methods for supervised texture segmentation. Internat. J. Comput. Vis. 46, 223–247.

Randen, T., Hakon, J., 1999. Filtering for texture classification: A comparative study. IEEE T. Pattern Anal. Machine Intell. 21, 291–310.

Sharon, E., Galun, M., Sharon, D., Basri, R., Brandt, A., 2006. Hierarchy and adaptivity in segmenting visual scenes. Nature 442, 810–813.

Shi, J., Malik, J., 2000. Normalized cuts and image segmentation. IEEE T. Pattern Anal. Mach. Intell. 22, 888–905.

Wang, J., Ju, L., Wang, X., 2009. An edge-weighted centridal Voronoi tessellation model for image segmentation. IEEE T. Image Process. 18, 1844–1858.