Technical Report OSU-CISRC-1/13-TR01
Department of Computer Science and Engineering
The Ohio State University
Columbus, OH 43210-1277

Ftpsite: **ftp.cse.ohio-state.edu**
Login: **anonymous**
Directory: **pub/tech-report/2013**
File: **TR01.pdf**
Website: **http://www.cse.ohio-state.edu/research/techReport.shtml**

Factorization-Based Texture Segmentation

Jiangye Yuan
Computational Sciences and Engineering Division
Oak Ridge National Laboratory, Oak Ridge, TN 37831 USA
yuanj@ornl.gov

DeLiang Wang
Department of Computer Science and Engineering & Center for Cognitive Science
The Ohio State University, Columbus, OH 43210, USA
dwang@cse.ohio-state.edu

Abstract – This paper introduces a factorization-based approach that efficiently segments textured images. We use local spectral histograms as features, and construct an $M \times N$ feature matrix using M-dimensional feature vectors in an N-pixel image. Based on the observation that each feature can be approximated by a linear combination of several representative features, we factor the feature matrix into two matrices – one consisting of the representative features, and the other containing the weights of representative features at each pixel used for linear combination. The factorization method is based on singular value decomposition and nonnegative matrix factorization. The method uses local spectral histograms to discriminate region appearances in a computationally efficient way and at the same time accurately localizes region boundaries. The experiments conducted on synthetic and natural images show the promise of this simple yet powerful approach.

Index Terms – matrix factorization, texture segmentation, spectral histogram.

1 Introduction

Image segmentation is a critical task for a wide range of applications including autonomous robots, remote sensing, and medical imaging. In this paper, we focus on segmentation of textured images, which partitions an image into a number of regions with similar texture appearance. As segmentation serves as an initial step for higher level image analysis tasks, such as recognition and classification, we aim to develop segmentation algorithms with low computational complexity. In addition, we do not use object-specific or scene-specific knowledge, which are typically not available.

Texture segmentation literature addresses two main issues: 1) finding an image model that defines region homogeneity, and 2) designing a strategy for producing segments. In general, these two issues are not treated independently due to the fact that a successful segmentation methodology should couple a good texture model and an effective segmentation strategy.

A popular choice for texture segmentation uses model-based methods, which regard textures as probability distributions and label pixels to different segments by maximum a posteriori (MAP) estimation. A typical example is to model a textured image as a Markov random field (MRF) [1] [2] [3], which takes interactions between the neighboring pixels into account. MRF provides an elegant framework for segmenting textured images.

Another line of work in texture segmentation is to extract features from local image patches and then feed them to general clustering or segmentation algorithms [4]. Various features are designed to characterize texture appearance. Widely used ones are based on filtering [5] [6], which uses filterbanks to decompose an image into a set of sub-bands, and statistical modeling [7] [8], which characterizes texture as resulting from some underlying probability distributions. Here, statistical modeling analyzes local distributions to produce features, in contrast with the model-based methods that obtain pixel labeling through parameter estimation. The features can be empirical distributions or estimated model parameters.

Recent work on texture analysis shows an emerging consensus that an image should first be convolved with a bank of filters [9]. Texture descriptors constructed based on local distributions of filter responses show promising performance for texture synthesis and texture discrimination. Such descriptors can be coupled with well-established segmentation methods to segment textured images [10] [11]. This treatment, however, has two main problems. The first problem stems from the high feature dimensionality of multiple filter responses and their distribution representations. Many widely used segmentation approaches, e.g., graph partitioning [12], curve evolution [13], and mean shift [14], heavily rely on measuring the distance among local features, and thus applying them to the texture descriptors requires a high computational cost for distance calculation. More importantly, it is always a thorny issue to select a proper distance measure for a high-dimensional space. Although dimensionality reduction techniques can be utilized, whether a technique is suitable for a feature often lacks theoretical justification. Using well-defined probability models to represent distributions can result in compact features that only involve a small number of parameters. However, these models usually lack expressive power.

The second problem is attributed to the texture descriptors generated from the image windows across boundaries. Such windows generate uncharacteristic features, which causes difficulty in accurately localizing region boundaries [10]. In order to address this problem, quadrant filters and other similar strategies are often employed, which compute features from shifted local windows around a pixel [15][16]. Another popular technique is to use local windows of different sizes, also referred to as scales [9] [17]. Boundaries are then determined by analyzing information across scales. Despite their success, these methods are ad hoc to some extent (e.g., using a discrete set of shifts), or require additional computation to analyze multiscale information. In such situations, it would be desirable to find a segmentation approach that can utilize the texture descriptors to discriminate region appearances in a computationally efficient way and at the same time accurately localize region boundaries.

In this paper, we propose a factorization-based segmentation method. The feature we use in this paper is a particular form of texture descriptors based on local distribution of filter responses, called local spectral histograms [18]. The proposed method represents an image by an $M \times N$ feature matrix, which contains M-dimensional feature vectors computed from N pixels. We regard the feature at each pixel as a linear combination of representative features, which encodes a natural criterion to identify boundaries. Consequently, the feature matrix is expressed by a product of two matrices, which respectively contain representative features and their combination weights per pixel. The combination weights indicate segment ownership for each pixel. We use singular value decomposition and nonnegative matrix factorization to factor the feature matrix, which leads to accurate segmentation.

The remainder of the paper is organized as follows. In Section 2, we present the factorization based image model, which uses local spectral histogram representation. Section 3 presents our segmentation algorithm in detail. In Section 4, we show experimental results on different types of images. Finally, we conclude in Section 5.

2 Factorization Based Image Model

2.1 Local Spectral Histograms

For a window $\mathbf{w}$ in an input image, a set of filter responses is computed through convolution with a chosen bank

of filters $\{F^{(\alpha)}, \alpha = 1, 2, \ldots, K\}$. For a sub-band image $\mathbf{W}^{(\alpha)}$, the corresponding histogram is denoted as $H_{\mathrm{W}}^{(\alpha)}$.[1] The spectral histogram with respect to a chosen filterbank is then defined as:

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

where $|\cdot|$ denotes cardinality. The size of the window is referred to as an integration scale. Spectral histograms capture local spatial patterns via filtering and global impression through histograms. It has been shown in [18] that when the filters are selected properly, the spectral histogram can uniquely represent an arbitrary texture appearance up to a translation.

A local spectral histogram is computed over the square window centered at each pixel location. In order to obtain meaningful features, the integration scale has to be large enough. Thus, computing all local histograms is computationally expensive. To address this issue, we use the integral images to speed up the histogram generation process. With integral histograms computed, any local spectral histogram can be obtained by three vector arithmetic operations regardless of window size. A detailed description of the fast implementation can be found in [16].

2.2 Image Model

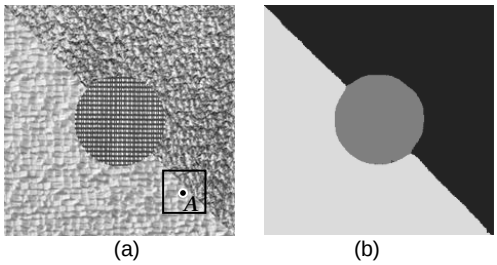
Without the loss of generality, let us consider an image as composed of homogenously textured regions as illustrated in Fig. 1(a). We assume that the spectral histograms within homogeneous regions are approximately constant. Local spectral histograms representative of each region can be computed from windows inside each region. Let us consider only the intensity filter for the time being, which gives the intensity value of each pixel as the filter response. Then the local spectral histogram is equivalent to the histogram of a local window. Under the assumption of spectral histogram constancy within the region, the local histogram of pixel A can be well approximated by the weighted sum of representative histograms of two neighboring regions, where the weights correspond to area coverage within the window and thus indicate which region pixel A belongs to. We can have the same analysis for other filter responses, as long as the scales of filters are not so large to cause significantly distorted histograms near the boundaries. Because the purpose of filtering in local spectral histogram is to capture elementary patterns, the chosen filters generally have small scales.

By extending the above analysis, the feature of each pixel can be regarded as the linear combination of all the representative features weighted by the corresponding area coverage. In the case when a window is completely within one region, the weight of the representative feature for that region is close to one, while the other weights are close to zero.

Given an image with N pixels and feature dimensionality of M, all the feature vectors can be compiled into an $M \times N$ matrix, $\mathbf{Y}$. Assuming that there are L representative features, the image model can be expressed as:

$$\mathbf{Y} = \mathbf{Z}\boldsymbol{\beta} + \boldsymbol{\varepsilon}, \quad (2)$$

where $\mathbf{Z}$ is an $M \times L$ matrix whose columns are representative features, $\boldsymbol{\beta}$ is an $L \times N$ matrix whose columns are weight vectors, and $\boldsymbol{\varepsilon}$ is model error.



(a) (b)

Fig. 1. Linear combination of representative features. (a) Textured image with size 320 × 320. The feature at pixel A can be approximated by the weighted sum of two neighboring representative features. (b) Segmentation result using least squares estimation.

This image model has been studied from a multivariate linear regression perspective in [19]. The representative feature matrix $\mathbf{Z}$ is computed from manually selected windows within each homogeneous region, and $\boldsymbol{\beta}$ is then estimated by least squares estimation:

$$\boldsymbol{\beta} = \left(\mathbf{Z}^T\mathbf{Z}\right)^{-1}\mathbf{Z}^T\mathbf{Y}. \quad (3)$$

Segmentation is obtained by examining $\hat{\boldsymbol{\beta}}$ – each pixel is assigned to the segment where the corresponding representative feature has the largest weight. For example, we compute three representative features from pixels around the center of each region in Fig. 1(a) and obtain the segmentation result shown in Fig. 1(b). We use an intensity filter and two LoG (Laplacian of Gaussian) filters with the scale values of 0.5 and 1.0 to compute local spectral histograms. The integration scale is chosen at 19 × 19. Given the ground truth, the segmentation error, which is the percentage of mislabeled pixels, is 1.1% for this segmentation.

3 Factorization Based Segmentation

For fully automatic segmentation, both $\mathbf{Z}$ and $\boldsymbol{\beta}$ are unknowns, and we aim to estimate these two matrices by factoring the matrix $\mathbf{Y}$. In this section, we present the factorization algorithm, which can produce segmentation with high accuracy and efficiency.

3.1 Low Rank Approximation

For a unique solution in (2) to exist, $\mathbf{Z}$ has to be full rank so that $\mathbf{Z}^T\mathbf{Z}$ in (3) is invertible. Thus, the rank of $\mathbf{Z}$ is the number of columns, i.e., representative features (the feature dimension is generally larger than the number of representative features). In other words, representative features have to be linearly independent in order to have a unique segmentation solution. Since each feature is a linear combination of representative features, the rank of the feature matrix $\mathbf{Y}$ should be equal to the rank of $\mathbf{Z}$. However, due to image noise, the matrix $\mathbf{Y}$ tends to be full rank. Hence, the noise-free feature matrix should be a matrix

[1] Based on previous studies [18], we use eleven equal-width bins for each filter response.

that has the rank equal to the number of representative features.

A typical solution to low rank approximation is singular value decomposition (SVD) [20], where the feature matrix is decomposed into:

$$\mathbf{Y} = \mathbf{U}\Sigma\mathbf{V}^T. \quad (4)$$

Here $\mathbf{U}$ and $\mathbf{V}$ are orthogonal matrices of size $M \times M$ and $N \times N$, respectively. The columns of $\mathbf{U}$ are the eigenvectors of matrix $\mathbf{YY}^T$, and the columns of $\mathbf{V}$ are the eigenvectors of matrix $\mathbf{Y}^T\mathbf{Y}$. $\mathbf{\Sigma}$ is an $M \times N$ rectangular diagonal matrix, where the diagonal terms, called singular values, are the square roots of the eigenvalues of the matrix $\mathbf{YY}^T$, or $\mathbf{Y}^T\mathbf{Y}$. The singular values are sorted in a nonincreasing order. The well-known Eckart–Young theorem [21] states that the best rank-r approximation to $\mathbf{Y}$, in the least-squares sense, has the same form of SVD, except that $\mathbf{\Sigma}$ is replaced with a new matrix that contains only the first r singular values (the other singular values are replaced by zero).

We need to determine the underlying rank of the feature matrix, which corresponds to the number of representative features, or segments[2]. Let $\mathbf{Y}'$ be the approximated matrix of rank-r. The approximation error can be obtained as follows:

$$\|\mathbf{Y} - \mathbf{Y}'\| = \sqrt{\sum_{i=r+1}^{M} \sigma_i^2}, \quad (5)$$

where $\|\cdot\|$ denotes the Frobenius norm, which is the square root of the sum of the squares of all matrix entries. σ_1, σ_2, …, σ_M are singular values in a nonincreasing order. Therefore, the error corresponds to the discarded singular values in the approximation. In practice, we can determine the number of segments by thresholding the error. That is, we estimate the segment number n as

$$n = \min\left\{ i : \sqrt{\sum_{i+1}^{M} \sigma_i^2} \Big/ N < \omega \right\}, \quad (6)$$

where ω is a pre-specified threshold that depends on the noise level of images.

3.2 SVD Based Solution

Assuming that the first r singular values are chosen using (6), (4) can be rewritten as:

$$\mathbf{Y}' = \mathbf{U}'\mathbf{\Sigma}'\mathbf{V}'^T \quad (7)$$

where $\mathbf{U}'$ and $\mathbf{V}'$ consist of the first r columns of $\mathbf{U}$ and $\mathbf{V}$ in the SVD of matrix $\mathbf{Y}$, respectively. $\mathbf{\Sigma}'$ is an $r \times r$ matrix with the largest r singular values on the diagonal. If we define $\mathbf{Z}_1 = \mathbf{U}'$ and $\boldsymbol{\beta}_1 = \mathbf{\Sigma}'\mathbf{V}'^T$, the two matrices $\mathbf{Z}_1$ and $\boldsymbol{\beta}_1$ are of the same size as the matrices $\mathbf{Z}$ and $\boldsymbol{\beta}$ in (2). Thus, $\mathbf{Z}_1$ and $\boldsymbol{\beta}_1$ can serve as a solution in the model in (2), which simultaneously ensures a minimum least square error due to the Eckart–Young theorem. However, the decomposition is not unique due to the fact that

$$\mathbf{Y}' = \mathbf{Z}_1\boldsymbol{\beta}_1 = \mathbf{Z}_1\mathbf{Q}\mathbf{Q}^{-1}\boldsymbol{\beta}_1 \quad (8)$$

where $\mathbf{Q}$ can be any invertible square matrix, suggesting that $(\mathbf{Z}_1\mathbf{Q})$ and $(\mathbf{Q}^{-1}\boldsymbol{\beta}_1)$ can also be possible solutions. Therefore, $\mathbf{Z}_1$ and $\boldsymbol{\beta}_1$ generally differ from the desired matrices that represent underlying representative features and combination weights. Although the decomposition cannot directly give a valid solution, it leads to a striking fact that the representative features should be in the form of $\mathbf{Z}_1\mathbf{Q}$, i.e., a linear transformation of $\mathbf{Z}_1$. Likewise, combination weights should be a linear transformation of $\boldsymbol{\beta}_1$.

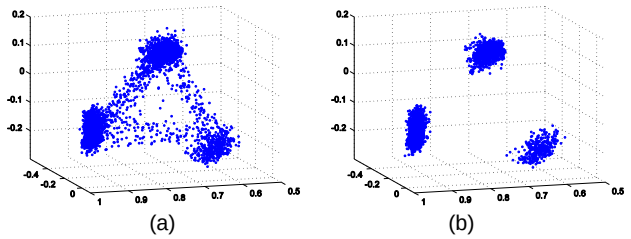


Fig. 2. Scatterplot of features in subspace. (a) Scatterplot of features projected onto the 3-d subspace. (b) Scatterplot after removing features with high edgeness

In order to obtain a segmentation result, we need to estimate $\mathbf{Q}$. Based on the fact that the desired matrix of representative features is a linear transformation of $\mathbf{Z}_1$, we know that the representative features should lie in an r-dimensional subspace spanned by the columns of $\mathbf{Z}_1$. Since $\mathbf{Z}_1$ forms an orthonormal basis, each column of $\mathbf{Q}$ is the Cartesian coordinate of each representative feature in the subspace. In the absence of noise, all other features also lie in the subspace because they are certain linear combinations of representative features. Meanwhile, there exist additional properties of the feature distribution owing to constraints on combination weights. Because the combination weights of a feature represent the coverage fraction of its local window, the weights should be nonnegative, and the sum of weights for each feature should be one. These two conditions restrict the features within a convex set with the vertices defined by representative features. In the case of r representative features, all features lie in an r-vertex convex hull, or an $(r$-1)-simplex, in an r-dimensional space.

As an illustration, we project all the features from the image in Fig. 1(a) onto the 3-dimensional space spanned by $\mathbf{Z}_1$, and show the scatterplot in Fig. 2(a). The data points are downsampled in order to better show the distribution. It is clear that the features approximately lie in a triangle. Most points are concentrated on the vertices, which correspond to the features inside each region, and the points along the edges correspond to the features near region boundaries. There are some points within the triangle, which correspond to the features computed over windows straddling all three regions.

As shown in the example above, most features aggregate densely at simplex vertices thanks to the discriminative power of local spectral histograms. Thus, we apply k-means clustering in the subspace with k equal to r, and cluster centers correspond to the representative features, which immediately give $\mathbf{Q}$. Then, $\boldsymbol{\beta}$ can be easily solved based on (8), which provides the segmentation result. As

[2]A representative feature corresponds to a segment. The connectivity of segments is not considered here, which can be achieved by postprocessing.

features are in a Cartesian space, the Euclidean distance is used as a metric.

It is well known that *k*-means clustering is sensitive to initialization. We employ two strategies to reduce the sensitivity. Because features near boundaries could make the clustering process stuck at local minima, we remove those features before clustering by computing an edge indicator. The indicator value of a feature at (x, y) is chosen as the sum of the two feature distances between the pixel locations $<(x + h, y), (x - h, y)>$ and $<(x, y + h), (x, y - h)>$, where h is chosen as half of the window size. The features with low values are expected to reside within the homogenous regions. Fig. 2(b) shows the features without those of high edgeness, which form well-grouped clusters. The second strategy is to choose initial means with maximum distance from each other [22]. Specifically, we choose the first initial mean $\mathbf{e}_1$ as the feature with the maximum length; then, the jth initial mean $\mathbf{e}_j$ is the feature with the largest distance to the set $S_{j-1} = \{ \mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_{j-1}\}$, where the distance between a feature $\mathbf{k}$ and S_{j-1}, denoted by $d(\mathbf{k}, S_{j-1})$, is defined as $d(\mathbf{k}, S_{j-1}) = \min\{d(\mathrm{k}, \mathbf{e}_1), d(\mathrm{k}, \mathbf{e}_2), \ldots, d(\mathrm{k}, \mathbf{e}_{j-1})\}$. By doing so, the initial means are near the simplex vertices. In our experiments, we observe that clustering results are very stable by incorporating these two simple strategies.

The representative feature matrix can alternatively be estimated by clustering in the original feature space. However, clustering in the subspace, derived from the above analysis, provides two major advantages. First, given the segment number, the SVD based solution guarantees minimum error, while the solution from direct clustering does not. As we showed earlier, the representative features from the subspace constructed by $\mathbf{Z}_1$ can give the best possible rank-r approximation that minimizes least square error, due to the Eckart–Young theorem. In contrast, the representative features from direct clustering are not guaranteed to lie in that subspace, and thus do not assure such a minimum error. Secondly, the subspace projection greatly reduces feature dimensionality, which speeds up the clustering significantly.

3.3 Nonnegativity Constraint

As we noted earlier, according to their interpretations, the combination weights should have two constraints, nonnegativity and full additivity (sum-to-one). However, the algorithm presented above does not enforce the combination weights to obey the constraints. When the features are very noisy, estimated combination weights can violate the constraints to a large extent, leading to incorrect segmentation. This problem is especially severe when the number of representative features is relatively large. Fig. 3(a) shows an image containing seven textures in 16 patches. The segmentation results from the unconstrained solution are shown in Fig. 3(b), where a large number of pixels are incorrectly segmented.

For a more accurate solution, we need to impose the constraints when estimating combination weights. This can be treated as constrained least squares estimation given the representative features from the SVD based solution. While a closed form solution exists for imposing full additivity [23], we find that the combination weights from the SVD-based solution are close to full additivity and the results with and without full additivity are very similar. The nonnegativity constraint can be achieved by a nonnegative least squares (NNLS) algorithm [24]. Our experiments show that this algorithm indeed improves segmentation accuracy, but the computation time is increased significantly, which remains a major limitation of the NNLS algorithm.

Fig. 3. Segmentation with nonnegativity constraint. (a) Synthetic image containing seven textures in 16 patches. (b) Segmentation result from the SVD-based solution. (c) Segmentation result with the nonnegativity constraint.

Alternating Least Squares (ALS) algorithms are proposed to efficiently provide a low rank approximation with nonnegative factored matrices [25]. ALS algorithms start from an initial matrix $\mathbf{A}$ and compute a matrix $\mathbf{B}$ using least squares estimation. After setting negative elements in $\mathbf{B}$ to zero, $\mathbf{A}$ is recomputed using least squares estimation. The operations are repeated in an alternating fashion. ALS algorithms have been applied to nonnegative matrix factorization problems and shown to be effective and fast in a number of studies [25]. ALS algorithms can be readily applied to our factorization problem, initialized with the representative features $\mathbf{Z}$ obtained from the SVD based solution. Based on the previous analysis, the initial $\mathbf{Z}$ should be close to the desired solution, hence a good initialization. Note that the initial $\mathbf{Z}$ is nonnegative because each column is a cluster center of all the features that are nonnegative histograms. As a result, ALS algorithms will converge to a solution near the initial $\mathbf{Z}$ and also enforce the combination weights to be nonnegative.

We employ a modified ALS algorithm [26], which minimizes the following function

$$f(\mathbf{Z}, \boldsymbol{\beta}) = \|\mathbf{Y} - \mathbf{Z}\boldsymbol{\beta}\|^2 + \lambda_1 \|\mathbf{Z}\|^2 + \lambda_2 \|\boldsymbol{\beta}\|^2 \qquad (9)$$

where $\mathbf{Z}$ and $\boldsymbol{\beta}$ have to be nonnegative and λ_1 and λ_2 are small regularization parameters. The regularization terms are used to penalize the factored matrices with very large norms. For all the experiments, we set both λ_1 and λ_2 to 0.1. The main steps of the algorithm are:

1. Initialize $\mathbf{Z}$ as the representative features of the SVD-based solution from Section 3.2.
2. Solve for $\boldsymbol{\beta}$ in matrix equation $(\mathbf{Z}^T\mathbf{Z} + \lambda_2\mathbf{I})\,\boldsymbol{\beta} = \mathbf{Z}^T\mathbf{Y}$.
3. Set all negative elements in $\boldsymbol{\beta}$ to 0.
4. Solve for $\mathbf{Z}$ in matrix equation $(\boldsymbol{\beta}\boldsymbol{\beta}^T + \lambda_1\mathbf{I})\,\mathbf{Z}^T = \boldsymbol{\beta}\mathbf{Y}^T$.
5. Set all negative elements in $\mathbf{Z}$ to 0.
6. Check whether stopping criteria are reached. If not, return to step 2.

Here $\mathbf{I}$ is the identity matrix. In each alternating step, $\mathbf{Z}$ and $\boldsymbol{\beta}$ are solved using least squares estimation. In addition to a maximum number of iterations (set to 100 in our experiments), we use a second stopping criterion: the difference of approximation errors between two consecutive iterations is less than 10^{-3}. Fig. 3(c) shows the segmentation result after applying the ALS algorithm. We can see that the segmentation accuracy is significantly improved.

3.4 Influence of Integration Scale

Local spectral histograms involve multiple scale parameters, including filter scales and integration scales. This is analogous to another widely used image descriptor, the structure tensor [27]. For a structure tensor, one scale corresponds to the scale of computing gradients, and the other describes the extent of local patches over which the structure tensor is constructed. Despite no theoretical relations, it is common in many practical applications to couple the two scale parameters in structure tensors by a constant. For local spectral histograms, with multiple filters, it is more complicated to couple the filter scales with the integration scales. Considering that the goal of filtering is to capture basic and small structures, we will use a set of filters with fixed scales and other parameters and focus on the effect of integration scales.

The choice of integration scales has a direct effect on segmentation results. Specifically, as the integration scale increases, the proposed method tends to produce smoother boundaries. To illustrate such an effect, we show an image containing jagged boundaries in Fig. 4(a), where a square window is used to compute a local feature. According to the coverage of the two regions within the window, the proposed method segments the corresponding pixel (the dot) into the darker region, as shown in Fig. 4(b). With the integration scale sufficiently large, we obtain a segmentation result shown in Fig. 4(c), where the boundary is close to a straight line.

Although the smoothing effect may cause loss of important details, like corners and small objects, it is interesting to note that the effect tends to reduce the total length of boundaries, and thus can serve as a form of regularization, which is often explicitly included as an objective for image segmentation in itself [28]. In our case, the smoothing effect emerges as a byproduct of the segmentation algorithm. Smoothing is apparent in the experimental analysis discussed in Section 4.

3.5 Computation Time

The feature extraction step in our algorithm, including filtering and spectral histogram computation, takes linear time with respect to the number of pixels. In our algorithm, we do not need to perform a complete SVD. After the eigenvalue decomposition of $\mathbf{Y}\mathbf{Y}^T$, which is an $M \times M$ matrix (M is the feature dimension), we only need the first several eigenvalues and the corresponding eigenvectors to construct $\mathbf{Z}_1$. $\boldsymbol{\beta}_1$ can be obtained by least square estimation. This process can be quickly completed. The k-means clustering for finding the matrix $\mathbf{Q}$ is also very fast because the features are projected onto a low dimensional subspace and the clusters are well grouped. The ALS algorithm generally stops in less than five iterations. We have implemented the whole system using Matlab. To segment a 256 × 256 image using seven filters, our algorithm takes a few seconds on an Intel Core 2 Quad 2.6 GHz processor.

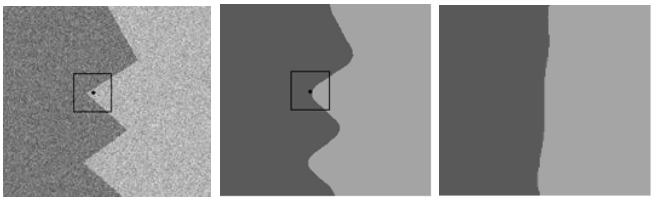


Fig. 4. Illustration of smoothing effect. (a) Synthetic image containing two regions with different Gaussian noise. (b) Segmentaion result using our method. (c) Segmentation result with a very large integration scale.

4 Experimental Results

In this section, we show the performance of our method on different types of images.

4.1 Texture Mosaics

We test our method on the Outex texture segmentation dataset [29]. The dataset contains 100 texture mosaics, which are generated using the ground truth image shown in the first column of Fig. 5. The size of each image is 512 × 512. Here we use two LoG filters with the scale values of 1.5 and 3.0 and eight Gabor filters with the orientations of 0°, 45°, 90°, and 135° and the scale values of 3.0 and 5.0. The integration scale is set to 55 × 55. By comparing the segmentation result with the ground truth, we calculate the segmentation error rates, the average of which on the dataset is 12.3%. To clarify the contribution of imposing the nonnegativity constraint, we examine the segmentation results from the SVD based solution, which gives an average error rate of 17.5%. Thus, using the ALS algorithm to enforce nonnegativity provides a clear improvement.

Fig. 5 presents four examples. In the columns containing three images, the first row shows the original images, the second row the results from the SVD based solution, and the last row the results from the complete method. As we can see in the last row, the regions with different textures are separated successfully, and the boundaries are localized well. Compared with the SVD based results, the complete method gives similar results when the SVD solution works sufficiently well, while considerably improves the results when the SVD solution fails to produce good segmentation. It is worth noting that a sizable portion of errors in the results are due to the smoothing effect, which tends to eliminate high frequency sinusoids on boundaries.

To quantitatively show the influence of the integration scale on segmentation results, we apply our algorithm to the images in Fig. 5 using four different integration scales, and the error rates are presented in Table I. As we can see, some integration scales are more favorable than others, as

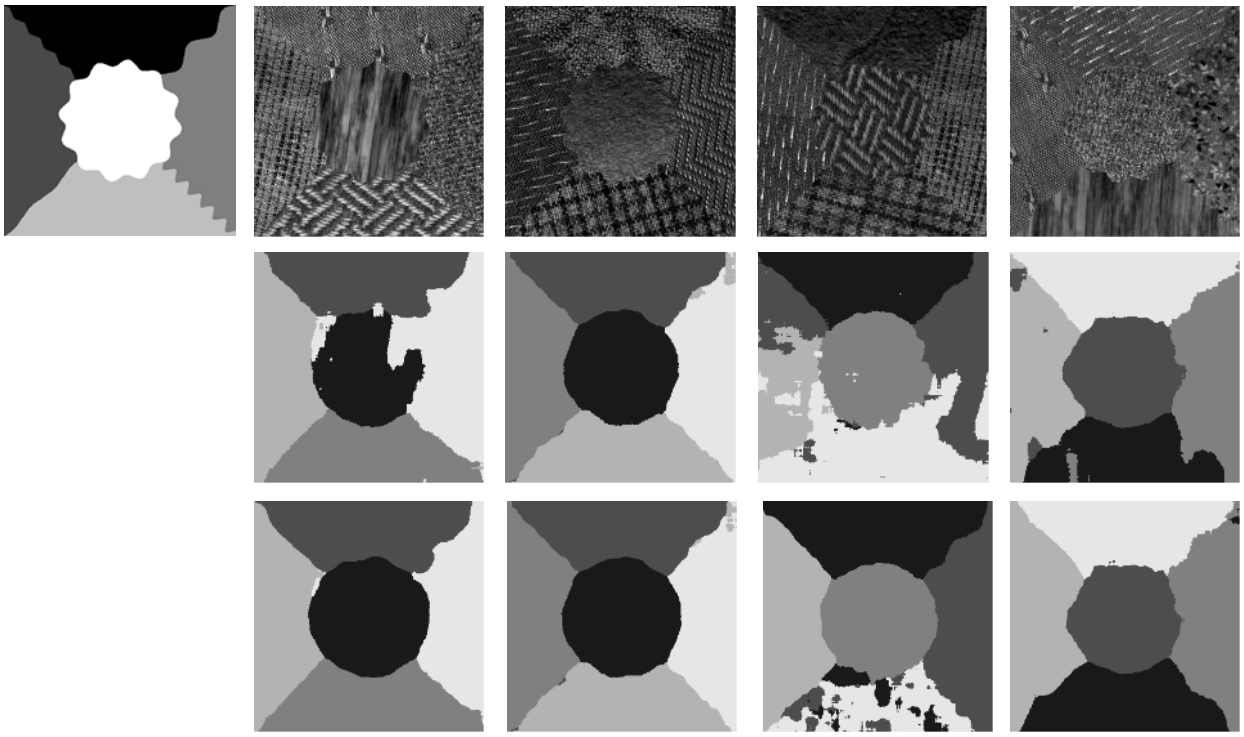


Fig. 5. Texture image segmentation. The first column is the ground truth. In the columns with three images, the first row is the original image, the second the segmentation result using the SVD based solution, the third the segmentation result using the complete method.

TABLE I
Segmentation error for the proposed method using different integration scales

Image	45 × 45	55 × 55	65 × 65	75 × 75
1	15.13%	4.26%	5.09%	5.19%
2	3.47%	3.88%	4.11%	4.19%
3	15.47%	11.77%	8.82%	8.32%
4	5.54%	4.45%	4.80%	5.01%

they achieve a good compromise between capturing meaningful local features and smoothing boundaries. We can also see that increasing integration scales does not lead to serious performance degradation.

4.2 Natural Images

We have also applied our method to the Berkeley Segmentation Dataset (BSD) [30]. To deal with color images, we include one intensity filter and two LoG filters with scale values of 0.5 and 1.0 to each color channel and compute local spectral histograms using all resulting bands. The integration scale is set to 17 × 17, and the threshold ω is set to 0.45 to determine the segment number. It has been noted that the $L^*a^*b^*$ color space is more perceptually uniform [31], which motivates the use of this color space for natural image segmentation [32] [33]. In our experiments, we convert RGB color values to the $L^*a^*b^*$ color space and apply filters to the converted color channels. Fig. 6 illustrates some representative segmentation results. It can be seen that, without involving any object-specific models or human intervention, our method generates rather meaningful results, where main regions are clearly segmented.

To put the performance of our method in perspective, we quantitatively compare with four publicly available segmentation methods: Multiscale Normalized Cut (MNC) [34], Compression-based Texture Merging (CTM) [32], Texture and Boundary Encoding-based Segmentation (TBES) [33], and Oriented Watershed Transform Ultrametric Contour Maps with globalPb as contour detector (gPb-owt-ucm) [4]. The comparison is based on 100 color images of the BSD testset. Two region-based quantitative measures are used, Probabilistic Rand Index (PRI) [35] and Variation of Information (VOI) [36]. PRI measures the probability that a pair of samples is labeled consistently in two segmentations, which has high values if two segmentations are similar. VOI measures the information difference between two segmentations, which gives low values if two segmentations are similar. Since each image in BSD has multiple ground-truth segmentations provided by different human subjects, a machine-generated segmentation is compared with each ground-truth segmentation, and the value of measures is given by the average. In addition to region-based evaluation, we also use a boundary-based measure, global F-measure (GFM) [9], which is defined as the harmonic mean of precision and recall. Precision measures the fraction of true boundaries in the detected boundaries, and recall measures the fraction of ground-truth boundaries that are detected. We apply the method used in [4] to compute the GFM measure with

Fig. 6. Natural image segmentation. In each pair, the left is the original image, and the right the segmentation result from the proposed method, where each region is indicated by its mean color.

multiple ground-truth segmentations.

We intend to consider both segmentation accuracy and computation time in comparison. The evaluation scores of four comparison methods on BSD are reported in [33] and [4]. We run the codes of comparison methods distributed by the authors on BSD to obtain computation time. The quantitative measures and the average time are presented in three plots of Fig. 7. For all three measures, our method does better than MNC. Compared with CTM, our method gives better PRI score, slightly better GFM score, and lower VOI score, which is partially because CTM optimizes information-theoretic criteria and thus favored by VOI. TBES and gPb-owt-ucm outperform our method, but at a much higher computational cost. Worth mentioning is that gPb-owt-ucm requires training images for the contour detector, while all othe methods are unsupervised. Overall, our method gives competitive results, but with the shortest running time.

Sandler and Lindenbaum propose a segmentation method based on nonnegative matrix factorization [37], which bears some resemblance to our method. However, there exist major differences. In their method, an image is divided into small tiles, and the histograms of all the tiles comprise the original matrix. Under their formulation, segmentation is performed based on tiles, and thus additional efforts are required to refine boundaries. Apart from a very different factorization algorithm, the factored matrices in their method cannot directly yield accurate segmentation, and anisotropic diffusion is performed to obtain final results. As reported in [37], their method gives a similar performance to MNC on the BSD testset at a fairly high computational cost (it takes minutes to obtain a useful factorization for a small matrix of size 32×256). Compared with their method, the proposed method gives better results with higher efficiency.

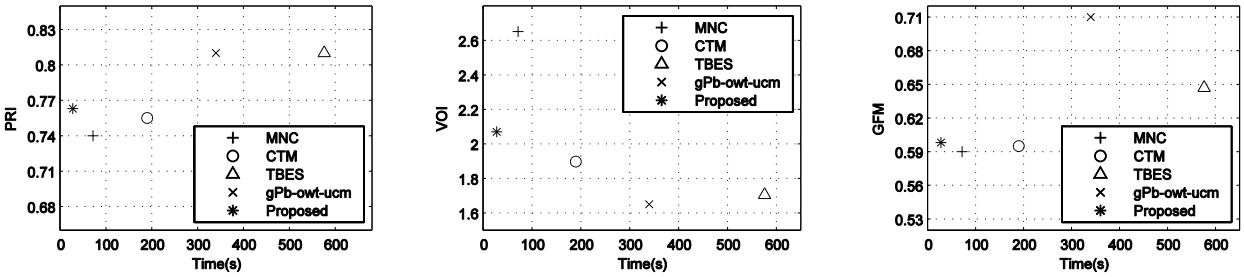


Fig. 7. Comparison on the BSD. The left is a PRI versus time plot, the middle a VOI versus time plot, and the right a GFM versus time plot. Four comparision methods are Multiscale Normalized Cut (MNC), Compression-based Texture Merging (CTM), Texture and Boundary Encoding-based Segmentation (TBES), and Oriented Watershed Transform Ultrametric Contour Maps with globalPb as contour detector (gPb-owt-ucm).

5 Concluding Remarks

In this paper, we have developed a new method for segmenting textured images. Using local spectral histograms as features, we frame the segmentation problem as a matrix factorization task. An efficient algorithm is proposed to produce factored matrices, which lead to accurate segmentation. Results of experiments on synthetic and natural images are very encouraging. The proposed method extends that in [19]. In that work, the feature subspace is not fully revealed, and the results are similar to those from the SVD based solution, but with less stability due to lack of proper initialization. More importantly, this paper explicitly relates segmentation to matrix factorization, more specifically to nonnegative matrix factorization. Although matrix factorization has been explored in many computer vision problems [38] [39], very little work has been done on its connection to image segmentation [37]. This important connection makes it possible to leverage extensively studied factorization techniques for improving segmentation results or adapting the method for specific tasks.

The proposed method is not limited to segmenting textured images. As can be seen in Section 4.2, satisfying results are obtained for non-textured natural images. However, the current version of our method is more suited for segmenting textured images because spectral histogram representations are particularly powerful for capturing texture information. For non-textured images, the proposed method works under the assumption that local histograms are similar within regions. Although such an assumption holds in many applications, similarity of local histograms can fail to reflect homogeneity in a non-textured region; e.g., histograms are sensitive to lighting changes. How to improve the performance of our method on general non-textured images will be a topic for future research.

Acknowledgement

This work was supported in part by an NGA University Research Initiatives grant (HM 1582-07-1-2027). The authors would like to thank Dr. Alper Yilmaz for valuable suggestions.

References

[1] G.R. Cross and A. Jain, Markov random field texture models, PAMI, 5(1): 25–39, 1983

[2] S. Krishnamachari and R. Chellappa, Multiresolution Gauss-Markov random field models for texture segmentation, IP, 6(2): 251–267, 1997

[3] D. Benboudjema and W. Pieczynski, Unsupervised statistical segmentation of nonstationary images using triplet Markov fields, PAMI, 29(8): 1367–1378, 2007.

[4] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. Contour detection and hierarchical image segmentation. PAMI, 33(5): 898–916, 2011.

[5] A. Jain and R. Farrokhsia, Unsupervised texture segmentation using Gabor filters, Pattern Recognition, 24: 1167–1186, 1991.

[6] T. Randen and J. Hakon, Filtering for texture classification: a comparative study, PAMI, 21(4): 291–310, 1999.

[7] J. Mao and A. Jain, Texture classification and segmentation using multiresolution simultaneous autoregressive models, Pattern Recognition, 25: 173–188, 1992.

[8] M. Tuceryan and A. Jain, Texture analysis, in The Handbook of Pattern Recognition and Computer Vision, C. H. Chen, L. F. Pau, and P. S. P. Wang, Eds., 2nd ed. River Edge, NJ: World Scientific, 207–248, 1998.

[9] D. Martin, C. Fowlkes, and J. Malik, Learning to detect natural image boundaries using local brightness, color, and texture cues, PAMI, 26(5): 530–549, 2004.

[10] J. Malik, S. Belongie, T. Leung, and J. Shi, Contour and texture analysis for image segmentation, IJCV, 43(1): 7–27, 2001.

[11] N. Paragios and R. Deriche, Geodesic active regions and level set methods for supervised texture segmentation, IJCV, 46(3): 223-247, 2002.

[12] J. Shi and J. Malik, Normalized cuts and image segmentation, PAMI, 22(8): 888–905, 2000.

[13] T. Chan and L. Vese, Active contours without edges, IP, 10(2): 266–277, 2001.

[14] D. Comaniciu and P. Meer, Mean shift: a robust approach toward feature space analysis, PAMI, 24(5): 603-619, 2002.

[15] S.C. Kim and T.J. Kang, Texture classification and segmentation using wavelet packet frame and Gaussian mixture model, Pat-

tern Recognition, 40(4): 1207–1221, 2007.
[16] X. Liu and D. L. Wang, Image and texture segmentation using local spectral histograms, IP, 15(10): 3066–3077, 2006.
[17] M. Galun, E. Sharon, R. Basri, and A. Brandt, Texture segmentation by multiscale aggregation of filter responses and shape elements, In ICCV, 2003.
[18] X. Liu and D. L. Wang, A spectral histogram model for texton modeling and texture discrimination, Vision Research, 42: 2617–2637, 2002.
[19] J. Yuan, D. L. Wang, and R. Li, Image segmentation using local spectral histograms and linear regression, Pattern Recognition Letters, 33(5): 615–622.
[20] G. Golub and C. Van Loan, Matrix Computations, 3rd ed., The Johns Hopkins University Press, 1996.
[21] C. Eckart and G. Young, The approximation of one matrix by another of lower rank, Psychometrika, 1: 211–218, 1936.
[22] J. T. Tou and R. C. Gonzalez, Pattern Recognition Principles. Reading, MA: Addison-Wesley, 1974.
[23] S. M. Kay, Fundamentals of Statistical Signal Processing: Estimation Theory, Englewood Cliffs, NJ: Prentice Hall, 1993.
[24] C. L. Lawson and R. J. Hanson, Solving Least Squares Problems, Englewood Cliffs, NJ: Prentice-Hall, 1974.
[25] M. Berry, M. Browne, A. Langville, V. Pauca, and R. Plemmons. Algorithms and applications for approximate nonnegative matrix factorization. Comput. Stat. and Data Anal., 52(1):155–173, 2007.
[26] R. Albright, J. Cox, D. Duling, A. Langville, and C. Meyer, Algorithms, initializations, and convergence for the nonnegative matrix factorization, NCSU Technical Report Math 81706, 2006.
[27] C. Harris and M. Stephens, A combined corner and edge detector, In Alvey Vision Conference, 1988.
[28] D. Mumford and J. Shah, Optimal approximations by piecewise smooth functions and associated variational problems, Comm. Pure Appl. Math., 42(5): 577–685, 1989.
[29] T. Ojala, T. Mäenpää, M. Pietikäinen, J. Viertola, J. Kyllönen, and S. Huovinen, Outex—New framework for empirical evaluation of texture analysis algorithms, In ICPR, 2002.
[30] D. Martin, C. Fowlkes, D. Tal, and J. Malik, A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics, In ICCV, 2001.
[31] A. Jain, Fundamentals of Digital Image Processing, Prentice Hall, 1989.
[32] A. Yang, J. Wright, Y. Ma, and S. Sastry, Unsupervised segmentation of natural images via lossy data compression. CVIU, 110(2):212–225, 2008.
[33] H. Mobahi, S. Rao, A. Yang, S. Sastry, and Y. Ma, Segmentation of natural images by texture and boundary compression, IJCV, 95(1): 86–98, 2011.
[34] T. Cour, F. Benezit, and J. Shi, Spectral segmentation with multiscale graph decomposition, In CVPR, 2005.
[35] W. Rand, Objective criteria for the evaluation of clustering methods, J. of the Am. Stat. Assoc., 66(336): 846–850, 1971.
[36] M. Meila, Comparing clusterings: an axiomatic view, In ICML, 2005.
[37] R. Sandler and M. Lindenbaum, Nonnegative matrix ractorization with earth mover's distance metric for image analysis, PAMI, 33(8): 1590–1602, 2011.
[38] C. Tomasi and T. Kanade. Shape and motion from image streams under orthography: A factorization method. IJCV, 9(2): 137-154, 1992.
[39] D. D. Lee and H. S. Seung. Learning the parts of objects by nonnegative matrix factorization. Nature, 401(6755): 788–791, 1999.