Factorization-Based Texture Segmentation

Jiangye Yuan, *Member, IEEE,* DeLiang Wang, *Fellow, IEEE,* and Anil M. Cheriyadat, *Member, IEEE*

***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 public segmentation datasets show the promise of this simple yet powerful approach.**

***Index Terms*—Matrix factorization, texture segmentation, spectral histogram.**

I. 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. These two issues should not be treated independently. A successful segmentation methodology generally couples a good image model with an effective segmentation strategy.

A broad family of texture segmentation approaches is to extract features from local image patches and then feed them to general clustering or segmentation algorithms. Various features are designed to characterize texture appearance. Widely used ones are based on filtering [1], [2], which uses filterbanks to decompose an image into a set of sub-bands, and statistical modeling [3], [4], which characterizes texture as resulting from some underlying probability distributions.

Recent work on texture analysis shows an emerging consensus that an image should first be convolved with a bank of filters [5]. Texture descriptors constructed based on local distributions of filter responses show promising performance for texture discrimination [6], [7]. Such descriptors can be coupled with well-established segmentation methods to segment textured images [8], [9]. 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 [10], curve evolution [11], and mean shift [12], 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 [13], [14]. Moreover, 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.

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 [9]. 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 [5], [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 II, we present the factorization based image model, which uses local spectral histogram representation. Section III presents our segmentation algorithm in detail. In Section IV,

J. Yuan and A. M. Cheriyadat are with Oak Ridge National Laboratory, Oak Ridge, TN 37831. (e-mail: {yuanj, cheriyadatam}@ornl.gov)

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

we show experimental results on different segmentation datasets. In Section V, we discuss the difference between the proposed method and other factorization-based methods in terms of methodology and performance. Finally, we conclude in Section VI.

II. Factorization based Image Model

A. 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 $H_{\mathbf{W}}^{(\alpha)}$.[1] The spectral histogram with respect to a chosen filterbank is then defined as:

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

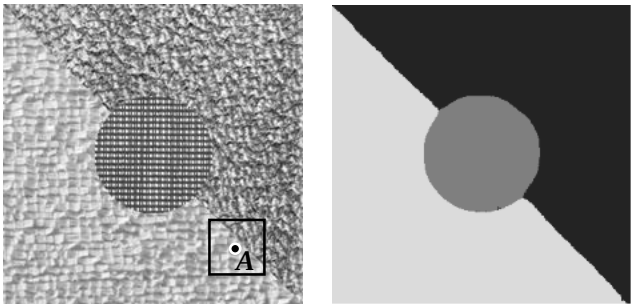
where $|\cdot|$ denotes cardinality. The size of the window is referred to as an integration scale. Spectral histograms capture local spatial patterns via filtering and global impression through histograms. It has been shown in [18] that when the filters are selected properly, the spectral histogram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

B. 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.

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



(a) (b)

Fig. 1. Linear combination of representative features. (a) 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.

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:

$$\mathbf{Y} = \mathbf{Z}\boldsymbol{\beta} + \boldsymbol{\varepsilon}, \tag{2}$$

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.

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

$$\hat{\boldsymbol{\beta}} = (\mathbf{Z}^T\mathbf{Z})^{-1}\mathbf{Z}^T\mathbf{Y}. \tag{3}$$

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). Here 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. It can be seen that the boundaries are accurately localized.

III. 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 $\mathbf{Y}$. In this section, we present the factorization algorithm, which can produce segmentation with high accuracy and efficiency.

A. 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 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:

$$\mathbf{Y} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T, \tag{4}$$

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 $\mathbf{Y}\mathbf{Y}^T$, 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 $\mathbf{Y}\mathbf{Y}^T$, 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 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:

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

where $||\cdot||$ denotes the Frobenius norm, which is the square root of the sum of all squared matrix entries. $\sigma_1, \sigma_2, ..., \sigma_M$ are singular values in a nonincreasing order. Therefore, the error corresponds to the discarded singular values in the approximation. We can determine the number of segments by thresholding the error. That is, we estimate the segment number n as

$$n = min\left\{ i : \frac{1}{N}\sqrt{\sum_{i+1}^{M} \sigma_i^2} < \omega \right\}, \tag{6}$$

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

B. SVD Based Solution

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

$$\mathbf{Y}' = \mathbf{U}'\mathbf{\Sigma}'\mathbf{V}'^T, \tag{7}$$

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

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

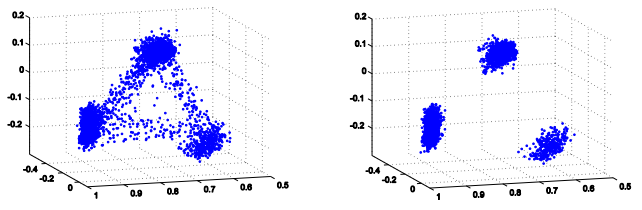
$$\mathbf{Y}' = \mathbf{Z}_1\boldsymbol{\beta}_1 = \mathbf{Z}_1\mathbf{Q}\mathbf{Q}^{-1}\boldsymbol{\beta}_1, \tag{8}$$

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. $\mathbf{Z}_1$ 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 $\boldsymbol{\beta}_1$.

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 $\mathbf{Q}$ corresponds to the Cartesian coordinate of each representative feature in the subspace. In the absence of noise, all other features also lie in this 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.

Based on the above analysis, each representative feature in the subspace (i.e., a column of $\mathbf{Q}$) should be a vertex of the underlying simplex and close to other features from the same region so that it can be representative. We use the following method to find such r points. After projecting all the features onto the subspace, we select the first point $\mathbf{e}_1$ as the feature with the maximum length; then, the jth selected point $\mathbf{e}_j$ is the feature with the largest distance to the set $S_{j-1} = \{\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_{j-1}\}$. The distance between a feature



(a) (b)

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.

(a) (b) (c)

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.

$\mathbf{k}$ and S_{j-1}, is defined as $d(\mathbf{k}, S_{j-1}) = min\{d(\mathbf{k}, \mathbf{e}_1), d(\mathbf{k}, \mathbf{e}_2), \ldots, d(\mathbf{k}, \mathbf{e}_{j-1})\}$. This procedure selects points near the simplex vertices. Then, we apply k-means clustering with the selected points as initialization, which moves each point to its cluster center. The resulting points give columns of $\mathbf{Q}$. After obtaining $\mathbf{Q}$, $\boldsymbol{\beta}$ can be easily solved based on (8), which provides the segmentation result. As feature points are in a Cartesian space, the Euclidean distance is used as a metric

Noisy features near region boundaries can generate points far outside the simplex, which results in selected points poorly approximating underlying representative features. To address this issue, we retain only features inside regions 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 side length. The features with low edgeness are expected to reside within the homogenous regions. Fig. 2(b) shows the features with low edgeness, where only points near vertices are preserved.

The representative feature matrix can alternatively be estimated through k-means clustering in the original feature space. However, the method presented above provides three major advantages. First, given the segment number, the factored matrices from SVD guarantee minimum error. As we explained earlier, the representative features from the subspace constructed by $\mathbf{Z}_1$ 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 hence do not assure minimum error. Secondly, the presented method is much more efficient because feature dimensionality is greatly reduced by subspace projection. An original feature often has over 100 dimensions, which can be reduced to the number of segments (usually less than 10). Last, k-means clustering is sensitive to initialization, while starting from points near simplex vertices provides good initialization and produces stable results.

C. 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 [22], we find that the combination weights from the SVD-based solution are close to full additivity and the segmentation results with and without full additivity are very similar. The nonnegativity constraint can be achieved by a nonnegative least squares (NNLS) algorithm [23]. 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.

Alternating Least Squares (ALS) algorithms are proposed to efficiently provide a low rank approximation with nonnegative factored matrices [24]. 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 [24]. 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. 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 [25], 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:

(a) (b) (c)

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.

1) Initialize $\mathbf{Z}$ as the representative features of the SVD-based solution from Section III-B.
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 $(\boldsymbol{\beta}\boldsymbol{\beta}^T + \lambda_2\mathbf{I})\mathbf{Z}^T = \boldsymbol{\beta}\mathbf{Y}^T$.
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 50 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.

D. 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 [26]. 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 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 produces 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 [27]. In our case, the smoothing effect emerges as a byproduct of the segmentation algorithm. In practice, we can choose a integration scale similar to a window size containing the largest non-repeating texture in an image or an image set, which can capture meaningful features without over-smoothing boundaries.

E. 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. Estimating $\mathbf{Q}$ is also very fast because the features are projected onto a low dimensional subspace. The ALS algorithm generally stops in less than ten iterations. We have implemented the whole system using Matlab. To segment a 320×320 image using seven filters, our algorithm runs in less than a second on an Intel 2.6 GHz processor.

IV. Experimental Results

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

A. Texture Mosaics

We first test our method on the Prague segmentation benchmark dataset [28]. The dataset contains 80 color texture mosaics of size 512×512. A benchmark website is developed along with the dataset, which calculates multiple accuracy indicators for segmentation results uploaded by users. The accuracy indicators include correct segmentation (CS), over-segmentation (OS), under-segmentation (US), missed error (ME), noise error (NE), omission error (O), commission error (C), class accuracy (CA), recall (CO), precision (CC), type I error (I.), type II error (II.), mean class accuracy estimate (EA), mapping score (MS), root mean square proportion estimation error (RM), comparison index (CI), global consistency error (GCE), and local consistency error (LCE). Here we use a filter bank consisting of the intensity filter, three LoG filters with the scale values of 0.8, 1.2, and 1.8, and eight Gabor filters with the orientations of 0°, 45°, 90°, and 135° and the scale values of 2.5 and 3.5. Since it has been noted that the *L*a*b** color space is more perceptually uniform [29], we convert RGB color values to the *L*a*b** color space. The intensity filter is applied to the three channels and all other filters to the gray scale image converted from the input image. Local spectral histograms are computed using all resulting bands. The integration scale is set to 60×60. A fixed ω value of 0.07 is used to determine segment numbers. Since the dataset contains some texture patterns that are composed

TABLE I
COMPARISON ON PRAGUE DATASET. UP ARROWS INDICATE BETTER RESULTS CORRESPOND TO LARGER VALUES, AND DOWN ARROWS THE OPPOSITE. THE BEST SCORES ARE MARKED IN BOLD. THE SECOND BEST SCORES ARE MARKED WITH STARS.

Method	TS	SWA	GMRF	AR3D	TFR	TFR+	RS	PMCFA	Proposed
↑ CS	59.13	27.06	31.93	37.24	46.13	51.25	46.02	**75.32**	69.02*
↓ OS	10.89	50.21	53.27	59.53	**2.37**	5.84*	13.96	11.95	17.30
↓ US	18.79	**4.53**	11.24	8.86	23.99	7.16*	30.01	9.65	11.85
↓ ME	10.45	25.76	14.97	12.54	26.70	31.64	12.01	**4.57**	6.28*
↓ NE	9.93	27.50	16.91	13.14	25.23	31.38	11.77	**4.63**	5.66*
↓ O	–	33.01	36.49	35.19	27.00	23.60	35.11	**4.51**	10.79*
↓ C	–	85.19	12.18	11.85*	26.47	22.42	29.91	**8.87**	13.75
↑ CA	–	54.84	57.91	59.46	61.32	67.45	58.75	**83.50**	77.50*
↑ CO	–	60.67	63.51	64.81	73.00	76.40	68.89	**88.16**	84.11*
↑ CC	–	88.17	89.26	**91.79**	68.91	81.12	69.30	90.73*	86.89
↓ I.	–	39.33	36.49	35.19	27.00	23.60	31.11	**11.84**	15.89*
↓ II.	–	**2.11**	3.14	3.39	8.56	4.09	8.63	1.47*	2.60
↑ EA	–	66.94	68.41	69.60	68.62	75.80	65.87	**88.10**	83.99*
↑ MS	–	53.71	57.42	58.89	59.76	65.19	55.52	**83.98**	78.25*
↓ RM	–	6.11	4.56	4.66	7.57	6.87	10.96	**3.76**	4.51*
↑ CI	–	70.32	71.80	73.15	69.73	77.21	67.35	**88.74**	84.71*
↓ GCE	–	17.27	16.03	12.13	15.52	20.35	11.23	**6.51**	10.82*
↓ LCE	–	11.49	7.31	6.69*	12.03	14.36	7.70	**3.92**	7.51

of large sub-patterns, region merging based on spatial relations has shown to be effective to improve results [33]. Here we use a simple post-processing step to form more complete segments. For each segment from our algorithm, an indicator is defined to be the ratio of the segment size to the length of the longest common boundary between the segment and one of its neighbors. A segment with a low indicator value is small and bordered by a dominant neighbor, and thus can be merged with the dominant neighbor. Segments keep merging until the lowest indicator value exceeds 40.

We compare the proposed method with algorithms that report leading performance on this dataset, including the Texel-based Segmentation algorithm (TS) [30], Segmentation by Weighted Aggregation (SWA) [17], Gaussian MRF Model (GMRF) [31], 3-D Auto Regressive Model (AR3D) [32], Texture Fragmentation and Reconstruction (TFR) [33], and its enhanced version (TFR+). Their accuracy scores are given in [30], [33]. We also include a recent method, the voting representativeness - Priority Multi-Class Flooding Algorithm (PMCFA), which reports the state-of-the-art performance[3]. In addition, we compare with the Regression-based Segmentation algorithm (RS) [19], which builds on the same image model.

Table I shows accuracy scores of the nine algorithms. Out of 18 indicators PMCFA achieves the best score for 14 indicators and the second best for two indicators. The proposed method achieves the second best score for 12 indicators. None of the remaining methods have more than three best or second best scores. PMCFA appears to combine a method from [34] with a hierarchical scheme. Given the lack of a detailed description, it is difficult to analyze the complexity of the complete algorithm. We find that the method in [34] reports a running time of 12 seconds on a 214×320 image. In contrast, the proposed method takes 2 seconds on a 512×512 image from the benchmark dataset, so it is likely that our method is significantly faster. Six examples of the results are displayed in Fig. 5. RS results are also presented for a visual comparison. The results from the proposed method are significantly better than RS results, as reflected by the quantitative measures, and very close to the ground truth.

[3]The results are provided at http://mosaic.utia.cas.cz/icpr2014/?5. No publication corresponding to the method can be found at the time of writing.

B. Natural Images

To assess the performance of our method on natural images, we test on the Berkeley Segmentation Dataset (BSD500) [35]. We apply the intensity filter to the *L*a*b** channels and three LoG filters with scale values of 0.5, 0.8, and 1.2 to the gray scale image. The integration scale is set to 21×21, and ω is set to 0.1. Fig. 6 illustrates some examples of our segmentation results. It can be seen that, without involving any object-specific models, our method generates rather meaningful results, where main regions are clearly segmented and boundaries are well localized.

We compare with two leading methods, Texture and Boundary Encoding-based Segmentation (TBES) [36] and oriented watershed transform ultrametric contour maps with globalPb as contour detector (gPb-owt-ucm) [37]. We follow the benchmarking procedure in [37], which compute two region-based quantitative measures, Probabilistic Rand Index (PRI) [38] and Variation of Information (VOI) [39], and a boundary-based measure, global F-measure (GFM). Better results correspond to higher scores for PRI and GFM and lower scores for VOI. Each algorithm is applied to the 200 images in the BSD500 testset.

Table II summarizes the quantitative measures of comparison methods as well as their average running time. Compared with TBES, our method gives slightly lower PRI and GFM scores, and a higher VOI score, which is partially because TBES optimizes information-theoretic criteria and thus favored by VOI. Our method is outscored by gPb-owt-ucm,

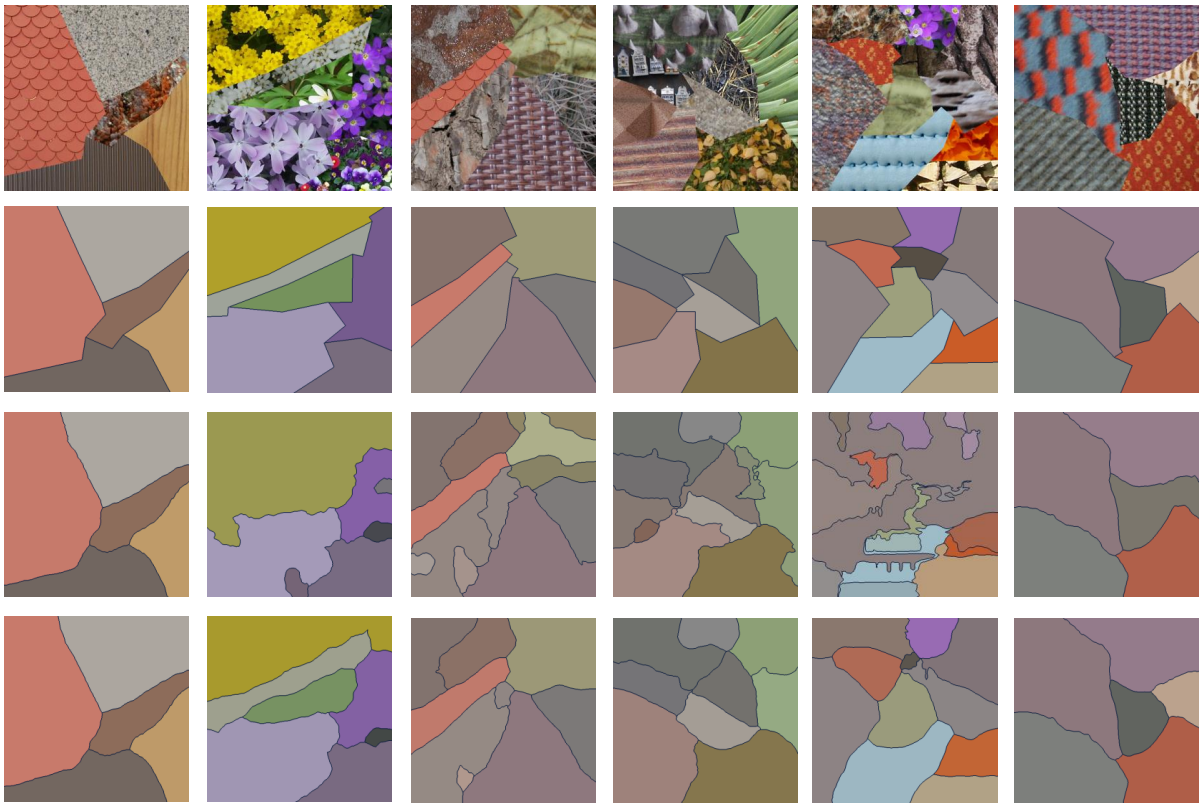


Fig. 5. Example results on Prague dataset. The first row shows original images, the second the ground truth, the third RS results, and the fourth segmentation results using the proposed method.

which unlike the other two unsupervised methods requires training images for the contour detector. Our method runs significantly faster than both methods. Given the fact that BSD500 is a benchmark dataset for general image segmentation while our method is designed for textured images, the performance of our method is rather appealing, especially considering its efficiency and simplicity.

V. Related work

Although matrix factorization has been extensively explored in many computer vision problems [40], [41], very little work has been done on its connection to image segmentation. Among a few methods that apply factorization to segmentation, we discuss two most related ones and how they are compared with the proposed method.

Sandler and Lindenbaum propose a segmentation method based on nonnegative matrix factorization [42]. 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 [42], their method gives a GFM score of 0.55 on the BSD and has 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.

A more recent method [43] factorizes local histograms at each pixel location, which bears more resemblance to our method. However, there exist major differences. Since the decomposition rule of histograms on boundaries is not utilized, to mitigate the boundary localization problem the method uses a small local window, which causes the difficulty in dealing with large texture. Also, the method determines the number of segment in an empirical way. Regarding performance, the method is inferior to comparison methods on the Prague dataset (the paper uses evaluation measures different from the standard scores provided by the dataset). In addition, the method takes 22 seconds to segment an image in the dataset, while the proposed takes 2 seconds.

TABLE II
Comparison on BSD500

Method	PRI	VOI	GFM	Time(s)
TBES	0.80	1.86	0.64	600
gPb-owt-ucm	0.83	1.69	0.73	340
Proposed	0.79	2.10	0.62	2.5

VI. Concluding Remarks

We have presented a simple and intuitive, yet powerful 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

Fig. 6. Example results on Berkeley segmentation dataset. 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.

to produce factored matrices that give meaningful segmentation. The experiments demonstrate that the proposed method performs consistently well on different datasets.

The proposed method extends the RS method in [19]. The feature subspace analysis and the factorization method result in significant improvements over the RS method. More importantly, this paper explicitly relates segmentation to matrix factorization, more specifically to nonnegative matrix factorization. This important connection makes it possible to leverage extensively studied factorization techniques for improving segmentation results or adapting the method for specific tasks.

Acknowledgment

This work was supported in part by U.S. Department of Energy/National Nuclear Security Administration under Grant DOE-NNSA/NA-22. This manuscript has been authored by employees of UT-Battelle, LLC, under contract DE-AC05-00OR22725 with the U.S. Department of Energy. Accordingly, the United States Government retains and the publisher, by accepting the article for publication, acknowledges that the United States Government retains a non-exclusive, paid-up, irrevocable, world-wide license to publish or reproduce the published form of this manuscript, or allow others to do so, for United States Government purposes.

References

[1] A. Jain and R. Farrokhsia, "Unsupervised texture segmentation using gabor filters," *Pattern Recognition*, vol. 24, no. 5, pp. 1167–1186, 1991.

[2] H. Ji, X. Yang, H. Ling, and Y. Xu, "Wavelet domain multifractal analysis for static and dynamic texture classification," *IEEE Transactions on Image Processing*, vol. 22, no. 1, pp. 286–299, 2013.

[3] J. Mao and A. Jain, "Texture classification and segmentation using multiresolution simultaneous autoregressive models," *Pattern Recognition*, vol. 25, pp. 173–188, 1992.

[4] 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. River Edge, NJ: World Scientific, 1998, pp. 207–248.

[5] D. Martin, C. Fowlkes, and J. Malik, "Learning to detect natural image boundaries using local brightness, color and texture cues," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 5, pp. 530–549, 2004.

[6] M. Crosier and L. D. Griffin, "Using basic image features for texture classification," *International Journal of Computer Vision*, vol. 88, no. 3, pp. 447–460, 2010.

[7] L. Liu and P. W. Fieguth, "Texture classification from random features," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 3, pp. 574–586, 2012.

[8] N. Paragios and R. Deriche, "Geodesic active regions and level set methods for supervised texture segmentation," *International Journal of Computer Vision*, vol. 46, no. 3, pp. 223–247, 2002.

[9] J. Malik, S. Belongie, T. Leung, and J. Shi., "Contour and texture analysis for image segmentation," *International Journal of Computer Vision*, vol. 43, no. 1, pp. 7–27, 2001.

[10] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 888–905, 2000.

[11] T. Chan and L. Vese, "Active contours without edges," *IEEE Transactions on Image Processing*, vol. 10, no. 2, pp. 266–277, 2001.

[12] D. Comaniciu, "Mean shift: a robust approach toward feature space analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 5, pp. 603–619, 2002.

[13] S. Alpert, M. Galun, A. Brandt, and R. Basri, "Image segmentation by probabilistic bottom-up aggregation and cue integration," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 34, no. 2, pp. 315–327, 2012.

[14] Z. Yu, A. Li, and O. Au, "Bag of textons for image segmentation via soft clustering and convex shift," in *Proceedings of the International Conference on Computer Vision*, 2012.

[15] S. Kim and T. Kang, "Texture classification and segmentation using wavelet packet frame and Gaussian mixture model," *Pattern Recognition*, vol. 40, no. 4, pp. 1207–1221, 2007.

[16] X. Liu and D. L. Wang, "Image and texture segmentation using local spectral histograms," *IEEE Transactions on Image Processing*, vol. 15, no. 10, pp. 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 *Proceedings of the International Conference on Computer Vision*, 2003.

[18] X. Liu and D. L. Wang, "A spectral histogram model for texton modeling and texture discrimination," *Vision Research*, vol. 42, pp. 2617–2637, 2002.

[19] J. Yuan, D. L. Wang, and R. Li, "Image segmentation using local spectral histograms and linear regression," *Pattern Recognition Letters*, vol. 33, no. 5, pp. 615–622, 2012.

[20] G. Golub and C. V. 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*, vol. 1, pp. 211–218, 1936.

[22] S. M. Kay, *Fundamentals of Statistical Signal Processing: Estimation Theory*. Prentice Hall, 1993.

[23] C. L. Lawson and R. J. Hanson, *Solving Least Squares Problems*. Society for Industrial and Applied Mathematics, 1974.

[24] M. W. Berry, M. Browne, A. N. Langville, V. P. Pauca, and R. J. Plemmons, "Algorithms and applications for approximate nonnegative matrix factorization," in *Computational Statistics and Data Analysis*, 2006, pp. 155–173.

[25] 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.

[26] C. Harris and M. Stephens, "A combined corner and edge detector," in *Alvey Vision Conference*, 1988.

[27] D. Mumford and J. Shah, "Optimal approximations by piecewise smooth functions and associated variational problems," *Communications on Pure and Applied Mathematics*, vol. 42, no. 5, pp. 577–685, 1989.

[28] M. Haindl and S. Mikes, "Texture segmentation benchmark," in *Proceedings of the International Conference on Pattern Recognition*, 2008.

[29] A. Jain, *Fundamentals of Digital Image Processing*. Prentice Hall, 1989.

[30] S. Todorovic and N. Ahuja, "Texel-based texture segmentation," in *Proceedings of the International Conference on Computer Vision*, 2009.

[31] M. Haindl and S. Mikeš, "Model-based texture segmentation," in *Image Analysis and Recognition*. Springer, 2004, pp. 306–313.

[32] M. Haindl and S. Mikes, "Unsupervised texture segmentation using multispectral modelling approach," in *Proceedings of the International Conference on Pattern Recognition*, 2006.

[33] G. Scarpa, R. Gaetano, M. Haindl, and J. Zerubia, "Hierarchical multiple markov chain model for unsupervised texture segmentation," *IEEE Transactions on Image Processing*, vol. 18, no. 8, pp. 1830–1843, 2009.

[34] C. Panagiotakis, I. Grinias, and G. Tziritas, "Natural image segmentation based on tree equipartition, bayesian flooding and region merging," *Image Processing, IEEE Transactions on*, vol. 20, no. 8, pp. 2276–2287, 2011.

[35] 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 *Proceedings of the International Conference on Computer Vision*, 2001.

[36] H. Mobahi, S. Rao, A. Yang, S. Sastry, and Y. Ma, "Segmentation of natural images by texture and boundary compression," *International Journal of Computer Vision*, vol. 95, no. 1, pp. 86–98, 2011.

[37] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik, "Contour detection and hierarchical image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 5, pp. 898–916, 2011.

[38] W. Rand, "Objective criteria for the evaluation of clustering methods," *Journal of the American Statistical Association*, vol. 66, no. 336, pp. 846–850, 1971.

[39] M. Meila, "Comparing clusterings: an axiomatic view," in *Proceedings of the International Conference on Machine Learning*, 2005.

[40] C. Tomasi and T. Kanade, "Shape and motion from image streams under orthography: a factorization method," *International Journal of Computer Vision*, vol. 8, no. 2, pp. 137–154, 1992.

[41] D. D. Lee and H. S. Seung, "Learning the parts of objects by non-negative matrix factorization," *Nature*, vol. 401, no. 6755, pp. 788–791, 1999.

[42] R. Sandler and M. Lindenbaum, "Nonnegative matrix ractorization with earth mover's distance metric for image analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 8, pp. 1590–1602, 2011.

[43] M. McCann, D. Mixon, M. Fickus, C. Castro, J. Ozolek, and J. Kovacevic, "Images as occlusions of textures: A framework for segmentation," 2013.