Information Sciences 370–371 (2016) 428–445

Contents lists available at ScienceDirect

Information Sciences

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

Random projections and Single BoW for fast and Robust texture segmentation

Li Liu[a,*], Liansheng Wang[b,**], Lingjun Zhao[c], Paul Fieguth[d]

[a] College of Information System and Management, National University of Defense Technology, 109 Deya Road, Changsha, Hunan, 410073, China
[b] Department of Computer Science, Xiamen University, Xiamen, Fujian, 361005, China
[c] College of Information System and Management, National University of Defense Technology, 109 Deya Road, Changsha, Hunan, 410073, China
[d] Department of Systems Design Engineering, University of Waterloo, Waterloo, Ontario, N2L 3G1, Canada

ARTICLE INFO

Article history:
Received 1 May 2016
Revised 19 July 2016
Accepted 6 August 2016
Available online 8 August 2016

Keywords:
Texture segmentation
Texture classification
Random projection
Bag of words
Sparse representation

ABSTRACT

This paper develops a theoretically simple, efficient yet robust approach to supervised texture segmentation based on local radial difference features and a Bag-of-Words (BoW) model. We make the following contributions: (1) we propose an approach to optimally learn new compact single BoW histogram models from the entire training set, with the single histogram providing benefits of efficiency in both memory and computation costs; (2) we show that BoW histograms computed from local simple radial difference features can provide an accurate pixel-wise segmentation of a textured image; and (3) we investigate whether sparse reconstruction, very successful in texture classification, assists in texture segmentation, with our study demonstrating the surprising conclusion that sparse reconstruction methods actually do not improve segmentation performance.

Extensive experiments on composite natural texture images demonstrate the superiority of the proposed approach over multiple state of the art texture segmentation methods. Our experimental evaluation demonstrates a significant superiority over recent popular sparse reconstruction segmentation methods in terms of computational efficiency, while outperforming or comparable in terms of segmentation accuracy.

1. Introduction

Texture is a fundamental property of images and is a key visual cue, with consequent significant research activity. In general, texture research may be divided into five canonical problem areas [40]:

1. Segmentation;
2. Classification;
3. Synthesis;
4. Compression; and
5. Shape from texture.

* Corresponding author. Fax: +86 731 84574531.
** Corresponding author. Tel.: +86 592 2580033.
E-mail addresses: liuli_nudt@nudt.edu.cn (L. Liu), lswang@xmu.edu.cn (L. Wang), nudtzlj@163.com (L. Zhao), pfieguth@uwaterloo.ca (P. Fieguth).

http://dx.doi.org/10.1016/j.ins.2016.08.014
0020-0255/© 2016 Elsevier Inc. All rights reserved.

The focus of this paper is on the first problem, that of supervised texture segmentation. As a fundamental problem in computer vision and image analysis, texture segmentation has a wide range of applications, including content based image retrieval, medical diagnosis, analysis of satellite or aerial images, industrial inspection, and document segmentation [34,35,40,45]. Texture segmentation has been attempted in numerous ways, normally involving two major steps: texture feature extraction and a segmentation algorithm [35]. Despite years of extensive study, surveys of which may be found in [34,35,40,45], the problem of texture segmentation remains challenging.

Texture segmentation is closely related to, but different from, texture classification [40]: the goal of texture segmentation is to obtain a texture boundary map, not necessarily estimating the class membership. Recent research on texture classification [18,22,23,42,46,51] focuses on estimating the class membership of an entire texture image photographed under unknown viewing and illumination conditions, given a number of texture classes and a set of training samples. The bag of visual words (BoW) paradigm [18,22,23,51], which represents texture images statistically as histograms over a dictionary of local features, has proved effective for image level texture classification. Although significant progress [18,22,23,42,46,51] has been made recently for image level texture classification, the question whether these approaches are suitable for texture segmentation demands further investigation.

Recently the theories of compressed sensing and sparse representation [7,8,13] have given new life to a variety of problems in computer vision, pattern recognition and image analysis, including image denoising [2], image classification [47–49], face recognition [15,44,50], and texture classification [25,26,31,33,39]. The goal of many such methods has been to exploit the underlying sparsity in the problem, either explicitly or implicitly, in order to improve the robustness, speed or accuracy with which classification might be performed. Inspired by the theories of compressed sensing and sparse representation, there are two main threads of research investigation: the sparse coding with over-complete dictionaries from a generative point of view [25,26,29,31,33,39], and the *K*Means clustering for textons from a discriminative point of view [22,23].

The former approach applies sparse coding ideas inspired by the work from the neuroscience community [29], in which class specific dictionaries are trained using labeled data and then each testing signal is assigned to the class for which the best reconstruction is obtained [25,26,31,39,44]. The performance of sparse coding relies on the quality of the learned over-complete dictionary, however there is no guarantee that the subspace built on a learned dictionary is optimal for classification, as it targets the representational power (best sparse representation for the training images) but not necessarily class separability. Furthermore there are concerns of the time-complexity of dictionary construction when there are a large number of classes.

In contrast to sparse generative models based on image patches, other approaches which implicitly leverage the sparsity nature of textures have been presented [22,23] for image level texture classification. In [22], Liu and Fieguth made an important innovation by introducing the use of random projections (RP), a universal, information-preserving dimensionality-reduction technique, to project the patch vector space to a compressed patch space without a loss of salient information, with further development in [23] for rotation invariance.

Despite the successes of these methods in texture segmentation and classification, the following three fundamental questions remain unanswered:

1. Is it of great importance to enforce sparsity constraints when extracting features for texture segmentation, despite the time costs?
2. Do the RP features presented by Liu et al. [22], which yielded excellent performance in classification, offer improvements to the pixel-wise texture segmentation task?
3. The BoW model has proved effective in region or image level classification, based on sparse or dense descriptors [18,27,43,46,51], representing each texture class by a set of histograms of textons. However the number of pixel-wise BoW histograms for texture segmentation would be quite large; is there a way to select texture class models for efficient and effective pixel-wise segmentations?

In this paper we extend our earlier work on image-level texture classification [22,23] to now represent a texture class by a *single* histogram of textons, and investigate the effectiveness of this representation for texture segmentation.

Our contributions in this paper are threefold:

1. We propose an approach to optimally learn new compact single BoW histogram models from a training set, with the single histogram providing benefits of efficiency in both memory and computation.
2. We show that BoW histograms computed from local simple radial-difference features can provide an accurate pixel-wise segmentation of a textured image.
3. Finally, we investigate whether sparse reconstruction, very successful in texture classification, assists in texture segmentation, with our study demonstrating the surprising conclusion that sparse reconstruction methods actually do not further improve segmentation performance.

Extensive experiments on composite natural texture images demonstrate the superiority of the proposed approach over multiple state-of-the-art texture segmentation methods. Our experimental evaluation demonstrates a significant superiority over recent popular sparse reconstruction segmentation methods in terms of computational efficiency, while outperforming or comparable in terms of segmentation accuracy.

The remainder of this paper is organized as follows. In Section 2 we give a brief review of background and related work. Section 3 develops a basic segmentation algorithm, followed by a detailed presentation of our proposed segmentation framework. In Section 4 we provide extensive experimental evaluation of the proposed approach against multiple state of the art alternatives on natural texture images.

2. Background and related work

2.1. Local texture features

Many methods have been proposed to describe texture in a quantitative way, leading to “a galaxy of texture features” [45], broadly lying in four categories: statistical methods [35,40], model-based methods [19,35], filtering methods [34,35,40], and structural methods [41]. As texture classification and segmentation are highly related problems, we are motivated to follow up on recent work in the classification literature, in which the BoW approach has emerged to become the dominant paradigm for texture classification [18,22,42,43]. In the BoW approach, texture images are represented statistically as histograms over a discrete dictionary of local features. Texture images are described locally by vectors of local texture descriptors, and a dictionary is defined as a partition of the local feature space, typically based on clustering. This distribution is represented by the frequency histogram of cluster centers (*i.e.* textons or dictionary atoms). Local highly discriminative yet robust texture features play a key role and many kinds of local texture descriptors have been proposed, both sparse [18,51] and dense [22,23,27,42,43], the latter extracting local features at each pixel. Sparse descriptors such as SIFT [51], RIFT [18] and SPIN [18] produce a sparse output and might miss important texture primitives. They are generally not used for texture segmentation. It is therefore the dense descriptors which have been widely studied for segmentation [25,26,31,39,44]. Among the most popular dense descriptors are the LM filter bank [20], MR8 [43], Gabor wavelets, LBP [27], Patch [42], RP [22] and SRP [23].

2.2. Dictionary learning

Dictionary learning aims to learn from training samples a good dictionary capable of concise representation. In the context of texture segmentation, the research in dictionary learning has followed two main directions of discriminative and generative learning.

The *discriminative* family of dictionary learning is based on vector quantization / *K*Means clustering. Leung and Malik [20] were amongst the first to apply the *K*Means approach for dictionary learning. Their algorithm optimizes a dictionary given a set of filter responses by first grouping patterns such that their distance to a given atom is minimal, and then by updating the atom such that the overall distance in the group of patterns is minimal. The implicit assumption here is that each feature point can be represented by a *single* atom.

In contrast, the *generative* approach models an image generated by linearly combining a set of dictionary atoms (*i.e.,* sparse coding). The underlying assumption here is that the signals being studied, such as natural images, admit sparse representations, an idea which has been used for learning features for image classification and has recently led to state-of-the-art results in classification tasks [25,26,31,39,44].

A common approach when using dictionaries for classification is to train class-specific dictionaries using labeled data and then to assign each testing signal to the class for which the best reconstruction is obtained [25,26,31,39,44]. A local texture feature $\underline{\boldsymbol{x}} \in \mathbb{R}^{n\times 1}$ is approximately represented as a linear combination of a small number of items from an overcomplete dictionary $\mathbf{D} \in \mathbb{R}^{n\times K}$ having K atoms, but where the representation $\underline{\boldsymbol{x}} = \mathbf{D}\underline{\boldsymbol{\alpha}}$ is sparse, such that $\|\underline{\boldsymbol{\alpha}}\|_0 \le \tau \ll K$ where $\|\cdot\|_0$ denotes the l_0 norm counting the number of nonzero entries in a vector. Learning an over-complete dictionary with a fixed number K of atoms, that is adapted to M patches of size n from texture images, is addressed by solving the following minimization problem:

$$\min_{\mathbf{D},\underline{\boldsymbol{\alpha}}} \sum_{i=1}^{M} \|\underline{\boldsymbol{x}}_i - \mathbf{D}\underline{\boldsymbol{\alpha}}_i\|_2^2, \quad s.t. \quad \|\underline{\boldsymbol{\alpha}}_i\|_0 \le \tau,\ \forall i,\ \|\underline{\boldsymbol{d}}_j\| \le 1,\ \forall j \tag{1}$$

where τ is a sparsity constraint factor, and where the sum measures the reconstruction error. The computation of $\mathbf{D}$ and $\{\underline{\boldsymbol{\alpha}}_i\}$ typically requires solving either an NP hard problem or an alternative problem that still involves a costly iterative optimization [2]. A given test texture is decomposed into overlapping patches (one patch per pixel), and one has to compute the sparse coefficient $\underline{\boldsymbol{\alpha}}$ of each patch vector $\underline{\boldsymbol{x}}$ in terms of dictionary $\mathbf{D}$ via sparse coding. The texture classification task itself is then performed based on the reconstruction error

$$\mathcal{R}(\underline{\boldsymbol{x}}, \mathbf{D}) = \|\underline{\boldsymbol{x}} - \mathbf{D}\underline{\boldsymbol{\alpha}}\|_2^2. \tag{2}$$

The attractiveness of the sparse approach is its robustness to noise and simplicity in both implementation and interpretation, but with significant drawbacks in terms of computational complexity.

2.3. Random projection

Random projection has become a widely-used method for dimensionality reduction, and has been shown to have promising theoretical properties: it is a general data reduction technique, such that the choice of random projection matrix does not depend upon the data in any way, and random projections have been shown to have special promise for high dimensionality data clustering.

The key idea of random projection arises from the JL lemma [10,16], which states that a point set in $\mathbb{R}^{n\times 1}$ with n typically large can be linearly projected into a lower-dimensional Euclidean space $\mathbb{R}^m$ using a random orthonormal matrix while approximately preserving the relative distances between any two of these points. Subsequent research [1,3] simplified the proof by showing that such a projection can be generated using an $m \times n$ matrix $\mathbf{\Phi}$, whose entries are randomly drawn from certain probability distributions [1], specifically including the Gaussian distribution [1,3]. The Johnson–Lindenstrauss lemma (JL lemma) has found numerous applications that include compressed sensing [44], dimensionality reduction in databases [1], and learning mixtures of Gaussians [10].

The information-preserving and dimensionality-reduction power of RP is firmly demonstrated by the theory of compressed sensing (CS) [8,13], which has grown out of the surprising realization that for sparse and compressible signals, a small number of nonadaptive linear measurements in the form of random projections can capture most of the salient information in the high-dimensional signal and allow for accurately reconstruction. Furthermore, Baraniuk et al. [3] give a simple technique for verifying the Restricted Isometry Property (RIP) for random matrices that underlies CS, meaning that random Gaussian projections approximately preserve pairwise distances in the data set.

3. Methodology

We wish to address the supervised texture segmentation problem. Given a test image $\mathbf{I}^{\text{obs}}$ consisting of C texture classes, with each class having a training image, we wish to classify each pixel in the test image as belonging to one of the texture classes. The segmentation task aims at finding a label field configuration $\mathbf{C}$ by correctly determining the class to which each pixel in $\mathbf{I}^{\text{obs}}$ belongs. The obtained label field configuration $\mathbf{C}$ partitions image $\mathbf{I}^{\text{obs}}$ into disjoint subregions.

Our texture segmentation architecture is shown in Fig. 2, which is related to the image level texture classification framework used in our recent work [22,23]. We describe below the components of our framework.

3.1. Local texture features

Based on the work by Liu et al. [22,23], who demonstrated the advantages of local random features for texture classification obtained under unknown viewpoint and illumination, a natural choice here is therefore to exploit the potential of random features for texture segmentation.

The local sorted random projection (SRP) features were first proposed by Liu et al. [23] for rotation-invariant texture classification, where the SRP radial-difference (SRP RadDiff) descriptor was shown to be the most discriminative and robust. However the texture segmentation problem, the focus of this paper, requires texture features to have high discrimination power but does not require rotation invariance, therefore we do not apply the sorting strategy of [23] and just use RP RadDiff, having the following strengths:

1. It is conceptually simple and computationally efficient, in contrast to SIFT, RIFT and SPIN descriptors [18,51].
2. The signed difference space is more compact and even sparser than the intensity space and allows the use of random projection.
3. Pairwise pixel interactions carry important structural information, and both short-range and long-range interactions are relevant.

It is noteworthy that local simple radial differences have been explored for face recognition and very good performance has been achieved [12,24].

Formally, given a pixel x_0 in the image, the radial difference vector $\underline{\boldsymbol{x}}$ for a patch of size $(2R+1)\times(2R+1)$ pixels centered at x_0 is defined as follows:

$$\underline{\boldsymbol{x}} = [(\underline{\Delta}^{\text{Rad}}_{1,8})^T, \cdots, (\underline{\Delta}^{\text{Rad}}_{R,8R})^T]^T \tag{3}$$

$$\underline{\Delta}^{\text{Rad}}_{r,p} = \underline{\boldsymbol{x}}_{r,p} - \underline{\boldsymbol{x}}_{r-1,p} \tag{4}$$

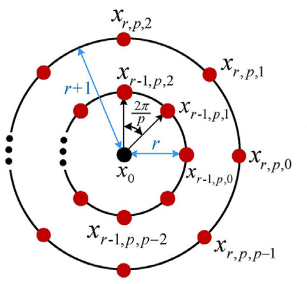
where $\underline{\boldsymbol{x}}_{r,p}$ represents the neighboring vector consisting of p neighboring pixels

$$\underline{\boldsymbol{x}}_{r,p} = [x_{r,p,0}, \ldots, x_{r,p,p-1}]^T$$

that were evenly distributed in angle on a circle of radius r centered on x_0, as shown in Fig. 1. Relative to the origin x_0 at (0, 0), the coordinates of the neighbor $x_{r,\,p,\,u}$ are given by $-r\sin(2\pi u/p), r\cos(2\pi u/p)$. The intensity values of neighbors not lying on a pixel location were estimated by interpolation.

When the RP technique is applied, given a feature vector $\underline{\boldsymbol{x}}$ of radial differences in a circular neighborhood, the regular radial difference feature vector $\underline{\boldsymbol{x}}$ is transformed as a random one via

$$\underline{\boldsymbol{y}} = \mathbf{\Phi}\underline{\boldsymbol{x}} \tag{5}$$



$$\underline{x}_{r,p} = \begin{pmatrix} x_{r,p,0} \\ \vdots \\ x_{r,p,p-1} \end{pmatrix}, \underline{x}_{r-1,p} = \begin{pmatrix} x_{r-1,p,0} \\ \vdots \\ x_{r-1,p,p-1} \end{pmatrix}$$

$$\underline{\Delta}_{r,p}^{\text{Rad}} = \underline{x}_{r,p} - \underline{x}_{r-1,p}$$

Fig. 1. A central pixel x_0, with p circularly and evenly spaced neighbors on circles of radius r and $r-1$.

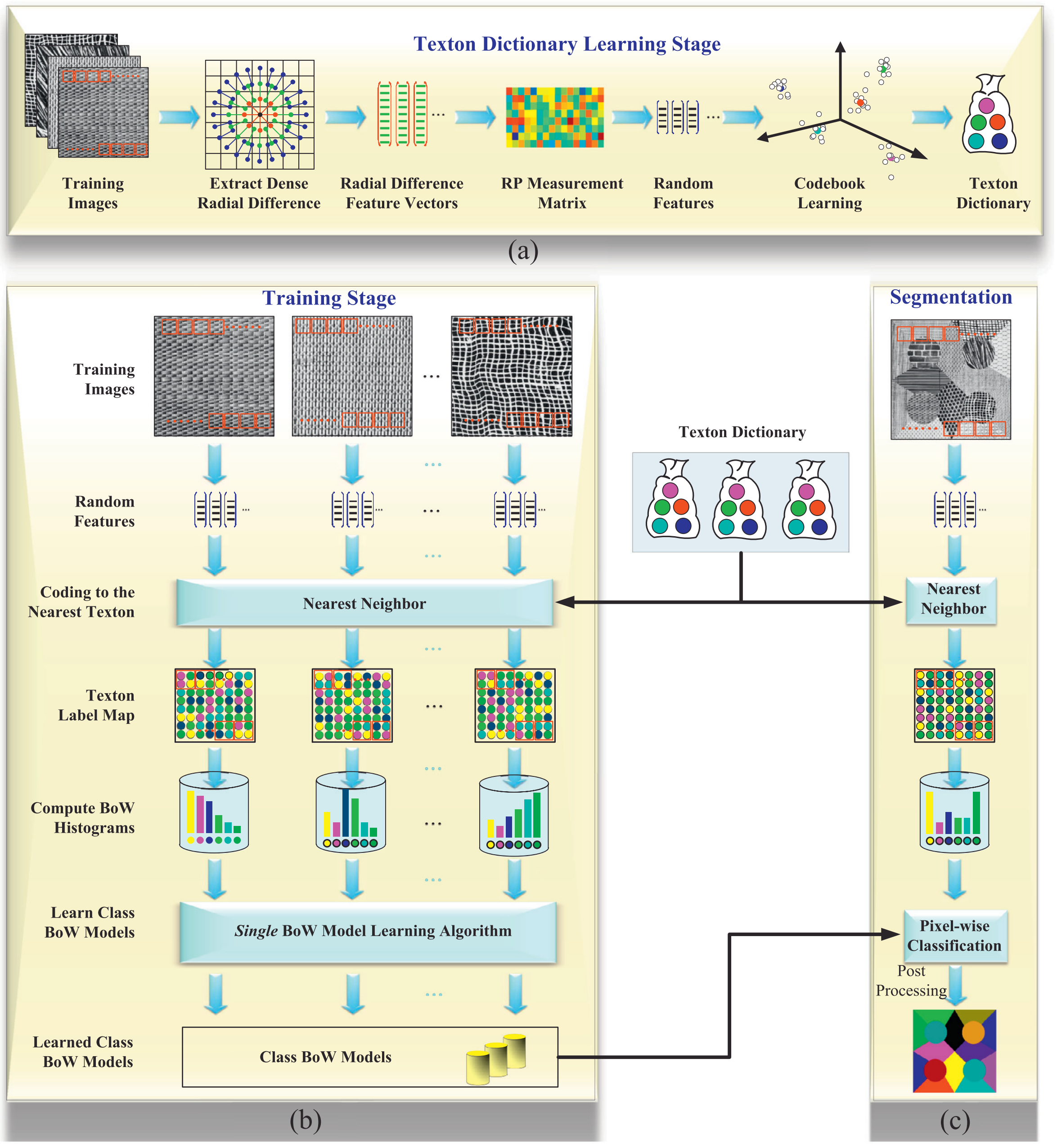


Fig. 2. Overview of the proposed texture segmentation architecture, building on texture classification work in [22].

by a random projection matrix $\mathbf{\Phi}$ whose entries are independently sampled from a zero-mean, unit-variance normal distribution. The random measurement matrix $\mathbf{\Phi}$ serves the function of nonadaptive, information-preserving, universal dimensionality reduction and projects the high dimensional radial difference vector $\underline{\boldsymbol{x}}$ to a much lower random feature vector $\underline{\boldsymbol{y}}$ without losing salient information. Readers are referred to [22,23] for details of the development of the basic RP and SRP classifiers.

3.2. Classification by Kmeans

We first develop an efficient baseline classifier. Assume that we have a training image $\mathbf{I}_c$ for each texture class, $c \in \{1, \ldots, C\}$. A set $\mathcal{S}_c = \{\underline{\boldsymbol{x}}_{ci}\}_i$, $\underline{\boldsymbol{x}}_{ci} \in \mathbb{R}^{n-1}$ of radial difference features is extracted from training image $\mathbf{I}_c$, where n denotes the size (in pixels) of a local patch used for extracting a radial difference feature. A simple strategy is to learn C dictionaries

$$\mathbf{D}_c = [\underline{\boldsymbol{d}}_{c1}, \underline{\boldsymbol{d}}_{c2}, \ldots, \underline{\boldsymbol{d}}_{cK_c}] \in \mathbb{R}^{n \times K_c}, \tag{6}$$

one for each class learned from set $\mathcal{S}_c$, where $\underline{\boldsymbol{d}}_{cj} \in \mathbb{R}^{n\times 1}$ is a dictionary atom, and K_c is the number of atoms in dictionary $\mathbf{D}_c$.

One simple and fast dictionary learning technique is the *K*Means clustering algorithm. The implicit assumption here is that each vector $\underline{\boldsymbol{x}}_{ci} \in \mathcal{S}_c$ can be approximated by a single atom $\underline{\boldsymbol{d}}_{cj}$ which has the minimal distance to $\underline{\boldsymbol{x}}_{ci}$. Then one can introduce

$$\begin{aligned} \underline{\boldsymbol{d}}^{\star}_{ci}(\underline{\boldsymbol{x}}_{ci}, \mathbf{D}_c) = & \quad \underset{\underline{\boldsymbol{d}}_{cj} \in \mathbf{D}_c}{\arg\min} \|\underline{\boldsymbol{x}}_{ci} - \underline{\boldsymbol{d}}_{cj}\|_2^2 \\ \mathcal{R}^{\star}(\underline{\boldsymbol{x}}_{ci}, \mathbf{D}_c) = & \quad \|\underline{\boldsymbol{x}}_{ci} - \underline{\boldsymbol{d}}^{\star}_{ci}(\underline{\boldsymbol{x}}_{ci}, \mathbf{D}_c)\|_2^2 \end{aligned} \tag{7}$$

Ideally, we want to learn each $\mathbf{D}_c$ from $\mathcal{S}_c$ to model the texture image $\mathbf{I}_c$ well. This can be found by choosing $\mathbf{D}_c$ such that it minimizes the residual error

$$\begin{aligned} \mathbf{D}_c &= \arg\min_{\mathbf{D}_c} \mathcal{R}^{\star}(\mathcal{S}_c, \mathbf{D}_c) \\ \mathcal{R}^{\star}(\mathcal{S}_c, \mathbf{D}_c) &= \sum_{i=1}^{|\mathcal{S}_c|} \mathcal{R}^{\star}(\underline{\boldsymbol{x}}_{ci}, \mathbf{D}_c) \end{aligned} \tag{8}$$

where $|\mathcal{S}_c|$ is the number of training vectors in each class. This clustering problem is commonly solved with the *K*Means algorithm.

At the pixel-wise classification stage, given the C dictionaries $\{\mathbf{D}_c\}_{c=1}^{C}$ learned from the previous dictionary learning stage, one approximates each patch vector $\underline{\boldsymbol{x}}$ with a dictionary atom and the C different dictionaries provide C different residual errors, which can then be used for deciding the class membership of a pixel. We then classify each pixel based on these residual errors by assigning it to the texture class that minimizes the residual error:

$$\hat{c}(\underline{\boldsymbol{x}}) = \underset{c=1,\ldots,C}{\arg\min} \mathcal{R}^{\star}(\underline{\boldsymbol{x}}, \mathbf{D}_c). \tag{9}$$

The final segmentation is obtained by applying a post smoothing algorithm (Section 3.4) on the output label map.

3.3. Proposed classifier: classification by a single BoW

Given a learned texton dictionary $\{\mathbf{D}_c\}_{c=1}^{C}$, it is possible to associate each pixel in the *training* images with the closest texton in the dictionary. In this way, we can compute a texton label map, $\mathbf{T}_c$, for each training texture image $\mathbf{I}_c$. Similarly, one can also compute the texton label map $\mathbf{T}^{\text{obs}}$ of the test image $\mathbf{I}^{\text{obs}}$. However in the baseline classifier described in Section 3.2, the spatial information contained in the texton label maps $\{\mathbf{T}_c\}_{c=1}^{C}$ of all the training images was not utilized. In order to leverage this information, we consider evaluating the dissimilarity of the training and testing BoW histograms. For this purpose, we propose the texture segmentation framework introduced in Fig. 2.

Let $\mathbf{T}_c(x, y)$ be the texton label of a pixel (x, y); for each such pixel a texton label patch of size $w \times w$ is extracted to compute a BoW histogram vector $\underline{\boldsymbol{h}}_{xy}$ of length N_{his} via

$$\underline{\boldsymbol{h}}_{xy}(k) = \sum_{a,b=-(w-1)/2}^{(w-1)/2} \delta(\mathbf{T}_c(x+a, y+b) - k), \tag{10}$$

where k is limited to the size of the universal texton dictionary

$$0 \le k \le \sum_{c}^{C} K_c = N_{\text{his}}. \tag{11}$$

Thus the model $\underline{\boldsymbol{h}}_{xy}$ is the normalized frequency histogram of pixel texton labelings, *i.e.*, an N_{his}-vector of texton probabilities.

For simplicity of notation, $\underline{\boldsymbol{h}}_{xy}$ will be denoted as $\underline{\boldsymbol{h}}_i$, where i is the lexicographic index of pixel (x, y). Each texture class c is represented by some number of models $\mathcal{H}_c = \{\underline{\boldsymbol{h}}_{cj}\}$ corresponding to the training image of that class.

Given the C set of BoW histogram models $\{\mathcal{H}_c\}_{c=1}^{C}$ computed from the training textures, the pixel-wise texture classification is the problem of estimating the true class label of each pixel in the image to be segmented: the local descriptors of the test image are generated and the pixels labeled with texton labels from the learned texton dictionary, then the normalized BoW histogram set $\mathcal{H}^{\text{obs}}$ is computed densely, and the class membership of each pixel in the test image is decided by comparing its BoW histogram feature with the training models using a suitable classifier. Multiple classification schemes have been explored for this task, most commonly nearest-neighbor matching [22,42,43] and kernel-based SVMs [46,51].

The key drawback of this BoW approach is its high dimensionality. Typically, due to the way that the BoW vectors are formed, even a moderately sized texton dictionary can lead to thousands of dimensions. For example, for a 16-class texture segmentation problem, the BoW feature has a dimensionality of 1600 if 100 textons per texture class are trained. Moreover, the sizes of the test model set $\mathcal{H}^{\text{obs}}$ or a training model set $\mathcal{H}_c$ is equal to the number of pixels in the testing or training images, respectively, which is usually tens of thousands to millions. In order to estimate the class membership of a BoW model $\underline{\boldsymbol{h}}^{\text{new}} \in \mathcal{H}^{\text{obs}}$, one needs to compute the pairwise distance between any $\underline{\boldsymbol{h}}^{\text{new}} \in \mathcal{H}^{\text{obs}}$ and any $\underline{\boldsymbol{h}}_i \in \bigcup_{c=1}^{C} \mathcal{H}_c$. This high dimensionality of the BoW feature and the large number of training and testing BoW models can be a severe obstacle for classification.

Therefore, we clearly wish to consider reducing the number of training models required to characterize each texture class. As discussed previously, the number of training models $\mathcal{H}_c$ for each texture class was the same as the number of pixels in training image $\mathbf{I}_c$. Ideally, we wish to reduce the number of models to that appropriate for each class, independent of the number of pixels in the training images.

To be sure, there are methods of model reduction in machine learning [14], to select a subset of the models while maximizing some criteria of classification and generalization, where the number of different models needed to characterize a texture is a function of how much the texture changes in appearance with imaging conditions, *i.e.* it is a function of the material properties of the texture. For example, textures may be modeled by a Markov Random Field (MRF) [9], which models a texture as a realization of a local and stationary random process. Each pixel of a textured image is characterized by a small set of spatially neighboring pixels (locality), and this characterization is the same for all pixels (stationarity). We imagine that a viewer is given a textured image, but is only allowed to observe it through a small movable window. As the window is moved the viewer can observe different parts of the image; the image is stationary if, under a proper window size, the observable portion always appears similar, and the image is local if each pixel is predictable from a small set of neighboring pixels and is conditionally independent of the rest of the image. In this case, the intrinsic number of effective BoW models could be significantly less than the number of pixels.

Based on these locality and stationarity assumptions, we want to select models for the express purpose of pixel-wise classification. Researchers have proposed to use *K*Medoids [43], *K*Means [14] or greedy search [43] methods to select significant models from the training set. However all three methods are highly computationally complex, for *K*Medoids and *K*Means due to the extremely high dimension of the BoW feature, and for greedy because of the exhaustive search which is undertaken.

Our goal, instead, is to develop a simple, efficient and general algorithm to texture segmentation, with few tunable input parameters. To this end, we propose to use compact *single* BoW models estimated optimally from the entire training BoW model set of each texture class. The key advantages of the single BoW histograms are that they are much more efficient both in terms of memory and computational resources.

The key question then is how to compute such single-histogram models. Let $\underline{\boldsymbol{h}}_{c,i} \in \mathcal{H}_c$ be one of the exemplar histograms from texture class c and $\underline{\boldsymbol{h}}_c^{\text{opt}}$ the single histogram model that we intend to seek. The *optimal* class histogram is the one which minimizes the overall distance to all of the $|\mathcal{H}_c|$ exemplar histograms $\underline{\boldsymbol{h}}_{c,i}$, as this minimizes intraclass variability, and, for best discrimination, one would also like to maximize the inter-class variability.

The optimal solution $\underline{\boldsymbol{h}}_c^{\text{opt}}$ depends on the histogram distance function $d(\underline{\boldsymbol{h}}_i, \underline{\boldsymbol{h}}_j)$ used during classification. In this paper we analyze and compare the three most common alternatives [37]: the Kullback–Leibler (KL) divergence, the Jensen–Shannon (JS) divergence, and the Chi-Square χ^2 distance. KL and JS are two important examples of information-theoretically motivated divergences.

The KL divergence is not a true metric since it is not symmetric and does not obey the triangle inequality, it is non-negative but can be unbounded. In contrast, the JS divergence between two normalized histograms is a measure that is symmetric, bounded, and numerically stable when comparing two empirical distributions. The χ^2 distance has been shown to be suitable for measuring the distance between two histogram features [22,42].

Given a class c we want to seek the model $\hat{\underline{\boldsymbol{h}}}$ which minimizes the cost

$$\mathcal{R}(\mathcal{H}_c, \underline{\boldsymbol{h}}) = \sum_{i=1}^{|\mathcal{H}_c|} d(\underline{\boldsymbol{h}}_i, \underline{\boldsymbol{h}}) \quad s.t. \quad \|\underline{\boldsymbol{h}}\| = 1,\ \underline{\boldsymbol{h}}(k) \geq 0,\ \forall k. \tag{12}$$

for some distance function d_{KL}, d_{JS}, or d_{χ^2}. In the KL case, the global minimum of (12) is found as [11]

$$\underline{\boldsymbol{h}}_c^{\text{opt}} = \frac{1}{|\mathcal{H}_c|}\sum_{i=1}^{|\mathcal{H}_c|}\underline{\boldsymbol{h}}_i \tag{13}$$

Motivated by the KL result, we use (13) to find the estimated model for the JS and χ^2 cases also.

With a distance metric and single BoW histogram defined, we are in a position to perform pixel wise segmentation of a given test texture image, as illustrated in Fig. 2. After the single BoW learning stage, an optimal BoW histogram $\underline{\boldsymbol{h}}_c^{\text{opt}}$ is learned from the training sample for each texture class c, resulting in a set of histograms $\mathcal{H}^{\text{opt}} = \{\underline{\boldsymbol{h}}_c^{\text{opt}}\}_{c=1}^{C}$. Each pixel in the test image is labeled with a texton from the learned texton dictionary. Next, a $v \times v$ pixel window, centered on the pixel i being classified, is used to compute a normalized frequency histogram of texton labelings to define a N_{his}-feature vector $\underline{\boldsymbol{h}}_i^{\text{new}}$ for the pixel. A nearest neighbor classifier is then used to assign the pixel to the class whose optimal model is most similar to $\underline{\boldsymbol{h}}_i^{\text{new}}$, where the distance between two normalized frequency histograms is measured using one of the KL, JS, χ^2 dissimilarity measures.

Near the image boundaries we apply a symmetric boundary condition. In terms of the $w \times w$ window size for the computation of the BoW histogram, a large window size will be computationally more expensive and may not able to localize texture boundaries well. The selection of w will be discussed in the experiments.

3.4. Post processing

Because adjacent pixels are very likely to belong to the same texture class, a strong prior clumping model can be asserted, and the classification performance can be improved with post smoothing. Clearly the manner in which a smoothing technique affects the error rate depends on the size and shape of the uniformly textured regions: more aggressive smoothing offers better classification within uniformly textured regions, with a cost of greater classification errors along the borders between regions.

Randen and Husøy [34] concluded that a separable Gaussian low-pass filter is the better choice, and it has also been used in [25,28,39]. However its non-uniformity makes it fairly expensive to evaluate. In our case we aim at filtering segmentation maps consisting of texture class indices, not residual error maps like those in [34,39]. Since the numerical ordering of indices normally has no physical meaning, using Gaussian smoothing may not be appropriate.

We propose to use a simple smoothing method, called a local mode filter (LMF) [30], based on the histogram of the indices in a local window centered at a pixel, and to also compare to a graph cut (GC) α-expansion algorithm [4,5,17].

3.4.1. Local mode filter

Given a class label map $\mathbf{C}^{\text{obs}}$ of a textured image, the LMF filter [30] proceeds by smoothing on a pixel-by-pixel basis, based on a window of $v \times v$ centered on each pixel. The feature response at location (i, j) is the probability of pixel labels in a window of size v centered at location (i, j):

$$\underline{\boldsymbol{f}}_{ij}(k) = \sum_{a,b=-(v-1)/2}^{(v-1)/2} \delta(\mathbf{C}^{\text{obs}}(i+a, j+b) - k), \quad k = 1, \dots, C. \tag{14}$$

with a symmetric condition at the image boundaries, and where C is the number of texture classes. The histogram $\underline{\boldsymbol{f}}_{ij}$ is essentially the occurrence frequency histogram of pixel class.

The class of pixel (i, j) is replaced to be the one that occurs the most frequently in a square neighborhood centered at location (i, j):

$$\hat{c}(i, j) = \arg\max_{k\in\{1,\dots,C\}} \underline{\boldsymbol{f}}_{ij}(k). \tag{15}$$

3.4.2. Graph cut

A GC α-expansion algorithm [4,5,17] based on a classical Potts model with an 8-neighborhood system is used as post processing, which closely follows the procedure described in [25]. Since each expansion move step consists essentially of the resolution of a max-flow problem on a suitable graph, a computationally efficient max-flow algorithm is needed. Therefore, we use the efficient algorithm introduced by Boykov in [4]. The constant regularization cost between two adjacent patches is denoted by parameter λ_{GC}.

4. Experiments

We wish to demonstrate the performance of the proposed approach for supervised texture segmentation with comprehensive experiments on a number of test images of varying complexity, summarized in Table 1. As the quality criterion, we use the classification error, based on the number of pixels misclassified. All texton dictionary learning and classifier training are performed on 256 × 256 sub-images of the texture images that are not in common with the test images.

Table 1
Summary of texture mosaics used in our experiments.

Experiment # 1					
Test number	Image size	Texture classes	Texture database	Texture classes in the mosaic for segmentation	Shown in
1	512 × 512	4	Brodatz	D1, D11, D15, D16	Fig. 3(a1)
2	512 × 512	4	Brodatz	D21, D22, D24, D25	Fig. 3(a2)
3	512 × 512	4	Brodatz	D17, D20, D26, D31	Fig. 3(a3)
4	512 × 512	4	Brodatz	D34, D36, D37, D49	Fig. 3(a4)
5	512 × 512	4	Brodatz	D38, D50, D51, D52	Fig. 3(a5)
6	512 × 512	4	Brodatz	D6, D53, D55, D56	Fig. 3(a6)
7	512 × 512	4	Brodatz	D64, D65, D68, D70	Fig. 3(a7)
8	512 × 512	4	Brodatz	D76, D77, D78, D80	Fig. 3(a8)
9	512 × 512	4	Brodatz	D79, D81, D82, D83	Fig. 3(a9)
10	512 × 512	4	Brodatz	D95, D101, D103, D105	Fig. 3(a10)
11	512 × 512	16	Brodatz	D6, D15, D16, D21, D22, D34, D49, D52, D53, D55, D77, D78, D95, D103, D104, D105	Fig. 3(b1)
12	512 × 512	16	Brodatz	D1, D11, D17, D20, D24, D26, D31, D36, D38, D50, D51, D56, D64, D65, D68, D76	Fig. 3(b2)
13	512 × 512	16	Brodatz	D1, D9, D37, D56, D70, D76, D79, D80, D81, D82, D83, D85, D92, D94, D96, D101, D102, D106	Fig. 3(b3)
14	512 × 512	16	Brodatz	D6, D21, D34, D36, D49, D50, D51, D52, D53, D55, D56, D64, D65, D68, D76, D77	Fig. 3(b4)
15	512 × 512	16	Brodatz	D15, D22, D37, D49, D52, D53, D55, D68, D77, D82, D83, D85, D94, D96, D103, D101, D104, D106	Fig. 3(b5)
16	512 × 512	16	Brodatz	D3, D4, D5, D9, D19, D29, D54, D57, D84, D85, D87, D92, D94, D104, D106, D112	Fig. 3(c1)
17	256 × 512	8	Brodatz	D10, D18, D23, D28, D33, D46, D74, D102	Fig. 3(d1)
18	256 × 512	8	Brodatz	D8, D47, D66, D71, D72, D73, D75, D109	Fig. 3(d2)
19	256 × 512	8	Brodatz	D12, D32, D86, D93, D96, D98, D110, D111	Fig. 3(d3)
Experiment # 2					
Mosaic number	Image size	Texture classes	Texture database	Texture classes in the mosaic for segmentation	Used by
1	256 × 512	2	Brodatz	D4, D84	[21,25,28,34,39]
2	256 × 512	2	Brodatz	D12, D17	[21,25,28,34,39]
3	256 × 512	2	Brodatz	D5, D92	[21,25,28,34,39]
4	256 × 256	5	Brodatz	D77, D84, D55, D53, D24	[21,25,28,34,39]
5	256 × 256	5	VisTex	Fabric.0000, Fabric.0017, Flowers.0002, Leaves.0006, Leaves.0013	[21,25,28,34,39]
6	256 × 256	5	VisTex	Fabric.0009, Fabric.0016, Fabric. 0019, Flowers.0005, Food.0005	[21,25,28,34,39]
7	256 × 256	5	VisTex	Fabric.0007, Fabric.0009, Leaves.0003, Misc.0002, Sand.0000	[21,25,28,34,39]
8	256 × 256	5	Meastex	Asphalt.0000, Concrete.0001, Grass.0002, Misc.0002, Rock.0005	[21,25,28,34,39]
9	256 × 640	10	Brodatz	D4, D9, D19, D21, D24, D28, D29, D36, D37, D38	[21,25,28,34,39]
10	256 × 640	10	VisTex	Fabric.0009, Fabric.0016, Fabric. 0019, Flowers.0005, Food.0005, Leaves.0003, Misc.0000, Misc.0002, Sand.0000, Stone.0004	[21,25,28,34,39]
11	512 × 512	16	Brodatz	D3, D4, D5, D6, D9, D21, D24, D29, D32, D33, D54, D55, D57, D68, D77, and D84	[21,25,28,34,39]
12	512 × 512	2	Vistex	Fabric.0007, Fabric.0009, Fabric. 0013, Fabric.0014, Fabric.0016, Flowers.0005, Food.0005, Grass.0001, Leaves.0003, Leaves.0008, Leaves.0012, Metal.0000, Metal.0002, Misc.0002, Sand.0000, and Stone.0004	[21,25,28,34,39]
13	512 × 512	4	Brodatz	D84, D54, D112, D22	[32]
14	512 × 512	4	Brodatz	D81, D85, D82, D80	[32]
15	512 × 512	4	Brodatz	D6, D19, D55, D84	[32]

The presentation of the experimental results is divided into two groups with corresponding objectives:

1. Section 4.2 provides thorough testing on Experiment #1 in Table 1, to examine the selection of the parameters in the proposed approach, summarized in Table 2. Our specific experimental goal is to compare the proposed approach with the state of the art sparse reconstruction based approach in [25].
2. Section 4.3 compares the proposed approach against the state of the art on the texture mosaics of Experiment #2.

4.1. Image data

Experiment #1: Over the last thirty years the evaluation of texture segmentation algorithms has been dominated by the Brodatz database [6], which has typically been used to provide single images of approximately 100 texture classes. We have used the reference images with boundaries of varying complexity, shown in Fig. 3.

Table 2
Summary of parameters involved in the proposed method.

Parameter name	Symbol	Default value	Description
Global image normalization	N/A	Apply normalization	Whether to apply global image normalization.
Patch size	$\sqrt{n}$	11	Size of a local patch used for local feature extraction.
RP dimension	m	40	Number of RP Measurements for different local patch sizes.
Texture classes no.	L	N/A	Number of texture classes in the test image to be segmented.
Number of textons per class	K	100	Number of textons learned from each texture class.
BoW feature dimension	N_{his}	$K \times L$	Number of bins in the BoW histogram feature.
Window size for BoW	w	35	The sliding window size used for BoW histogram computation.
LMF filter size	v	45	The window size used for Local Mode Filter (LMF) smoothing.

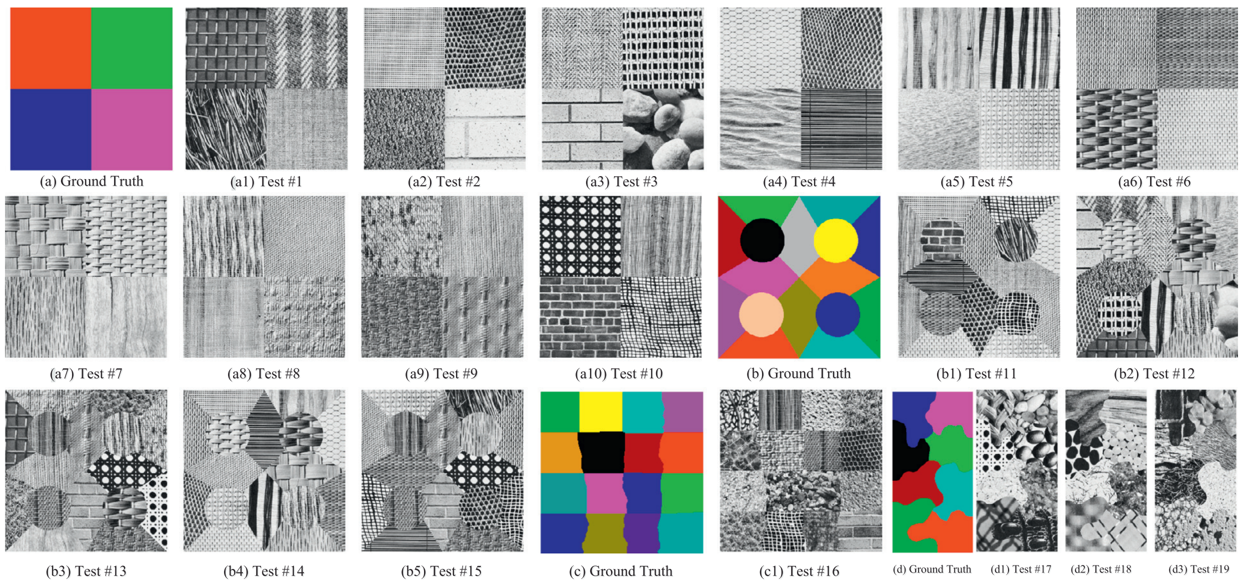


Fig. 3. Test texture mosaics from Experiment #1 in Table 1, and corresponding ground truth images for: (a) Test 1–10, (b) Test 11–15, (c) Test 16, and (d) Test 17–19. The texture classes used to generate the test mosaics are given in Table 1. The horizontal boundaries in (c) are straight, with vertical boundaries generated by a random walk. All textures are selected from the Brodatz database. The texture classes within each texture mosaic are listed in Table 1.

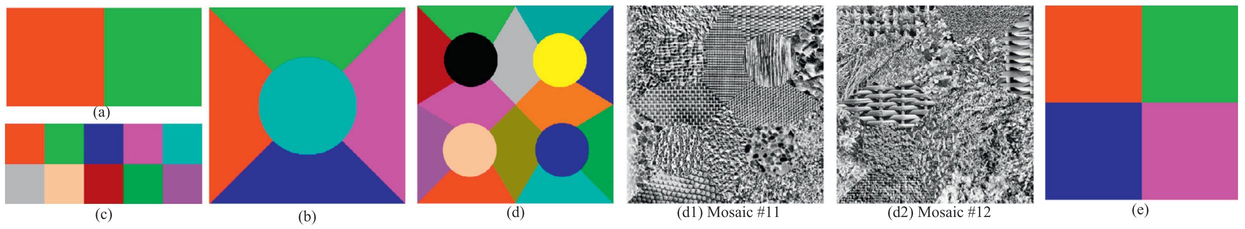


Fig. 4. Ground truth images for Mosaics (a) 1–3, (b) 4–8, (c) 9–10, (d) 11–12, and (e) 13–15. Due to space limitations only two mosaics (d1, d2) are shown. The texture classes used to generate the test mosaics are given in Table 1.

We have 19 texture mosaics (Tests 1–19), detailed in Table 1 and illustrated in Fig. 3, which are generated from a set of 82 textures from the Brodatz database. Some textures in the Brodatz database essentially belong to the same class but at different scales (D1 and D6, D25 and D26), while others are so inhomogeneous that a human observer would arguably be unable to group their samples correctly (*e.g.*, D43, D44, D45, D91 and D97). Based on these considerations, we selected 82 texture classes from the Brodatz album by visual inspection because the remaining 29 textures may not be suitable for texture segmentation.

For each texture present in a test mosaic there is a 256 × 256 training image that is extracted from a different area in the source image so that an unbiased error estimate is obtained. No additional processing of the source images is applied.

Experiment #2: We employed the twelve texture mosaics used in the supervised segmentation problems of the comparative study of Randen and Husøy [34] and in [21,25,28,39], and the three texture mosaics from the recent work by Qin

Table 3
Comparison of segmentation error rates (%) obtained by different dissimilarity measures: KL, JS and χ^2. The RadDiff descriptor and the single BoW classifier are used. All involved parameters are set to the defaults in Table 2. The LMF filter is used for post processing.

Test no.	#1	#2	#3	#4	#5	#6	#7	#8	#9	#10	#11	#12	#13	#14	#15	#16	#17	#18	#19
KL	2.44	2.00	6.25	3.00	3.08	2.41	16.44	2.73	1.56	1.72	8.30	13.29	6.88	14.05	8.42	12.87	23.92	20.06	16.23
JS	0.77	**0.36**	1.91	1.07	2.47	0.67	**3.08**	**0.62**	1.36	**0.61**	3.15	7.60	3.84	5.38	3.44	6.89	7.18	11.19	6.29
χ^2	**0.76**	**0.36**	**1.89**	**0.93**	**2.41**	**0.65**	**3.08**	0.65	**1.35**	**0.61**	**3.04**	**7.56**	**3.82**	**5.22**	**3.41**	**6.84**	**7.05**	**11.08**	**6.24**

and Clausi [32], shown in Fig. 4. In their comparative study, Randen and Husøy reviewed dozens of filtering approaches to texture segmentation and also included two classical nonfiltering approaches: co-occurrence and autoregressive features.

We will also compare the proposed approach with the recent pure sparse reconstruction based work of [25,39]. The parameters that we used for the sparse reconstruction method are those recommended by the work of [25]: a patch of size $n = 121$ (11×11), dictionaries of size $K = 256$, a sparsity factor $\tau = 4$, and the KSVD dictionary learning algorithm with 20 iterations. We use the publicly available Matlab codes of KSVD. The sparse reconstruction algorithm used here is Orthogonal Matching Pursuit (OMP) [25], which is already a computationally simpler algorithm among all of the reconstruction algorithms presented in the literature.

This dataset involves images from three different sources: Brodatz [6], MIT Vision Texture [52], and the MeasTex [34] databases. In order to get comparable results we followed the experimental setup of Randen and Husøy as closely as possible.

Implementation details: To make the comparisons as meaningful as possible, we follow the same training and testing methodology as in [21,25,28,34,39]. Each of the patches of the training images were used as a training set; each image (training or testing) is normalized to be zero mean and unit standard deviation, and the extracted local feature vector is normalized to have unit norm. Because of this normalization the gray level mean and deviation are roughly equal between the training and testing images. However, since the training and testing portions were extracted from different locations in the large source image, there are a few notable, even visible, differences in the gray level properties between the training and testing portions of some texture classes in certain mosaics.

4.2. Implementation evaluation

4.2.1. Evaluation of different dissimilarity measures

Table 3 shows the classification results on the dataset of Experiment #1. We can see that KL divergence always performs more poorly than either of the χ^2 and JS measures, similar to the findings given in [37]. The χ^2 and JS behave more stably than the KL divergence, as expected. The classical χ^2 statistic and the JS divergence yield nearly identical results on all test mosaics, with the χ^2 distance performing slightly better. Therefore the χ^2 distance will be used in the remainder of this paper.

The reason for KL weakness stems from empty histogram bins. The model histograms are estimated from the training images, which contain a certain percentage of pixels which do not occur in the ideal or perfect class models, possibly due to quantization effects of either clustering or histogram binning, a result of an inadequate amount of training data, or due to outliers or noise. As a consequence, the learned (training or testing) BoW histogram models have some zero bins, a standard pitfall in histogram based density estimation. These zero bins have a much greater influence on the KL measure than on JS or χ^2.

4.2.2. Evaluation of BoW and LMF window sizes

Figs. 5 and 6 plot the pixel wise classification error as a function of window size; for visualization purposes, we plot the results for all 19 test mosaics in different panels since the classification accuracies vary significantly for different test mosaics. The behavior is clearer in the averaged result, in Fig. 5(d) and Fig. 6(d), where some intermediate optimum is visible, where larger v values result in oversmoothing and larger w in more mislabeled pixels along texture boundaries.

We also design a set of experiments to investigate whether the optimal window size can be decided via cross validation (CV) on both datasets in Experiment #1 and #2. Table 4 shows the classification results: a fixed window size of $w = 35$ yields results as good as CV, and nearly as good as the result produced by the optimum choice of w in each test. Therefore CV is not pursued, and a fixed window size of $w = 35$ will be used for the Kmeans+BoW approach.

4.2.3. Evaluation of different approaches

The performance of the proposed method is assessed in Table 5: the proposed KMeans + single BoW approach performs significantly better than KMeans on its own, however the random projection approach, so effective in texture recognition, offers no significant improvement.

We have undertaken extensive experiments, reported in Table 6, in which the best classification error rates are highlighted. Each part (a,b,c) of the table compares random, dense and sparse coding approaches, and the dense approach slightly outperforms the random projection (RP) and sparse (KSVD) ones, the weakness of the latter already observed in [36,38], which noted that a sparse reconstruction based method is not necessarily required for image classification.

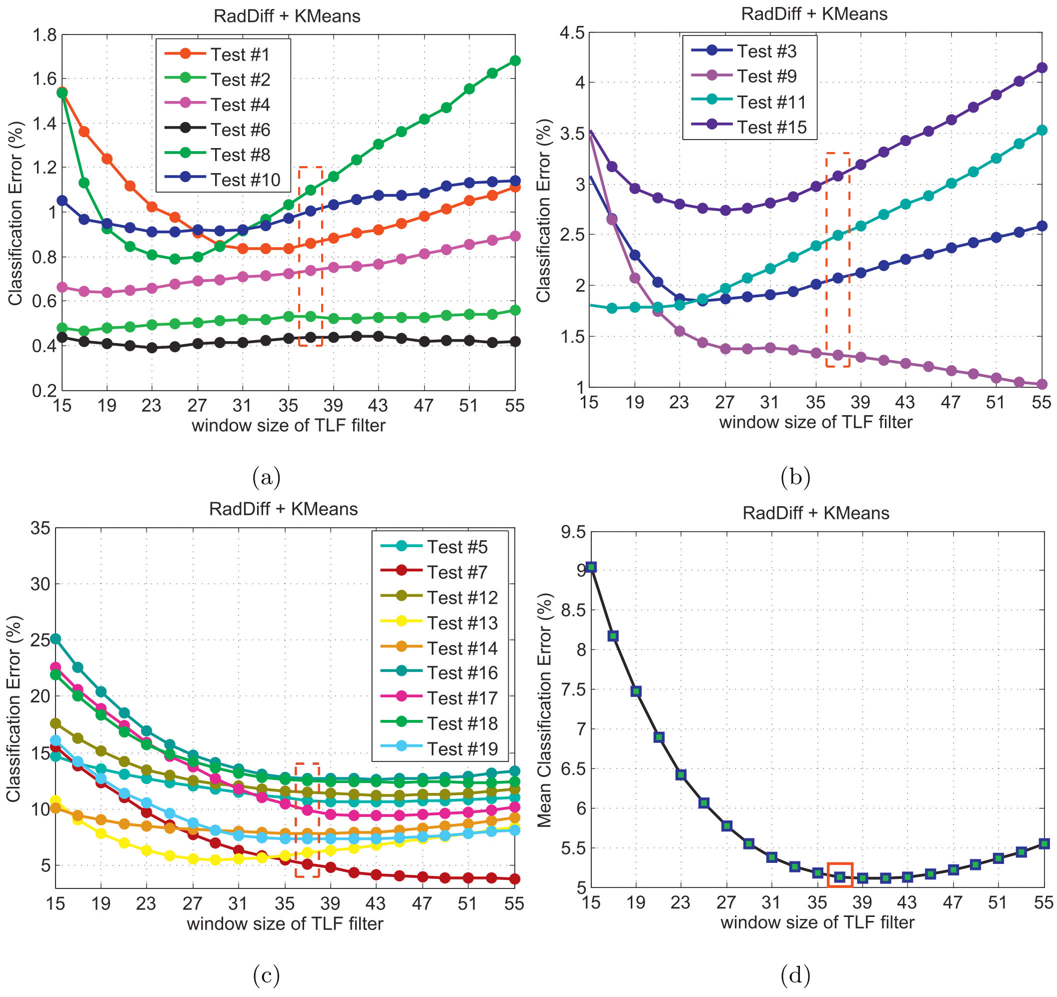


Fig. 5. Segmentation error rate of the Basic Classifier (RadDiff + KMeans) as a function of the LMF window size v.

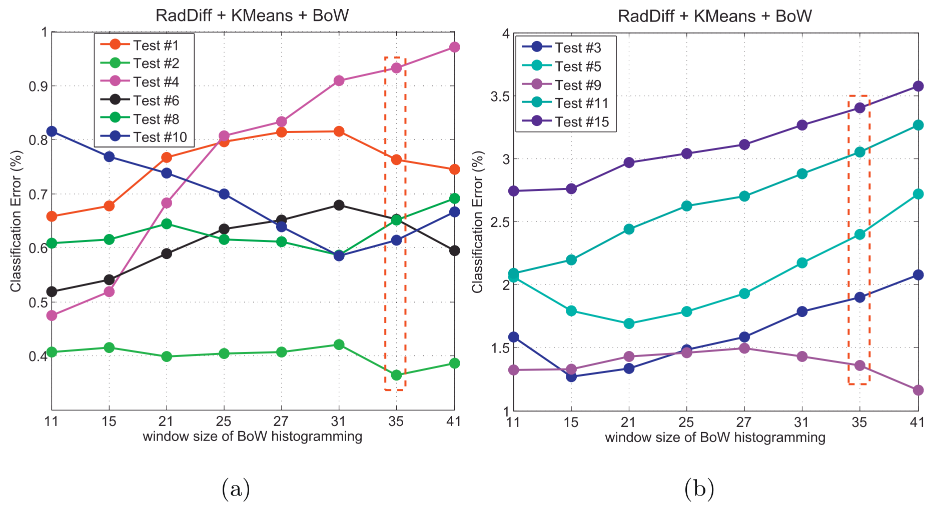
Table 4
Comparison of the segmentation error rates (%) for three different training strategies regarding the BoW window size w: fixed $w = 35$, cross validation over w, or the best (minimum) result over w. A fixed value of w produced results as good as cross validation, and nearly as good as the optimum.

Test #	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	Average
Fixed $w = 35$	0.76	0.36	1.90	0.93	2.40	0.65	3.05	0.65	1.36	0.61	3.05	7.57	3.82	5.22	3.40	6.76	7.06	11.14	6.22	3.52
CV over w	0.67	0.42	1.58	0.48	1.79	0.52	3.10	0.61	1.16	0.81	2.09	8.12	3.90	7.13	2.74	6.79	7.48	10.72	6.72	3.52
Min over w	0.66	0.36	1.27	0.47	1.69	0.52	2.19	0.59	1.16	0.58	2.09	7.57	3.50	5.22	2.74	6.74	6.84	10.74	5.92	3.20

Table 6(a) and (b) compare patch-based versus radial-difference-based features, with the radial-difference approach, on average, outperforming. The results shown in Table 6(b) are also plotted in Fig. 7(a). Similar to Fig. 7(a), Fig. 7(b) plots the results on the test mosaics in Experiment # 2. We do expect the RadDiff descriptor to be the more powerful, as it incorporates the more discriminative power of the cooccurring differences than that of the intensities in an image patch. Furthermore the RadDiff descriptor is inherently gray-scale invariant, a very useful property when the gray-scale properties of the unknown test sample differ from the training data, which is the case in most of the 19 test mosaics used in this study.

4.2.4. Evaluation of postprocessing methods

Our pixel wise classification results have been post-processed by smoothing to obtain the final segmentation, which closely follows the procedure described in the work by Mairal et al. [25]. We have implemented two alternatives, either using a simple LMF filtering, or applying a GC α-expansion algorithm of Kolmogorov and Zabih [4,5,17], based on a classical Potts model with an 8-neighborhood system. A constant regularization cost between two adjacent pixels is denoted as λ_{GC}.



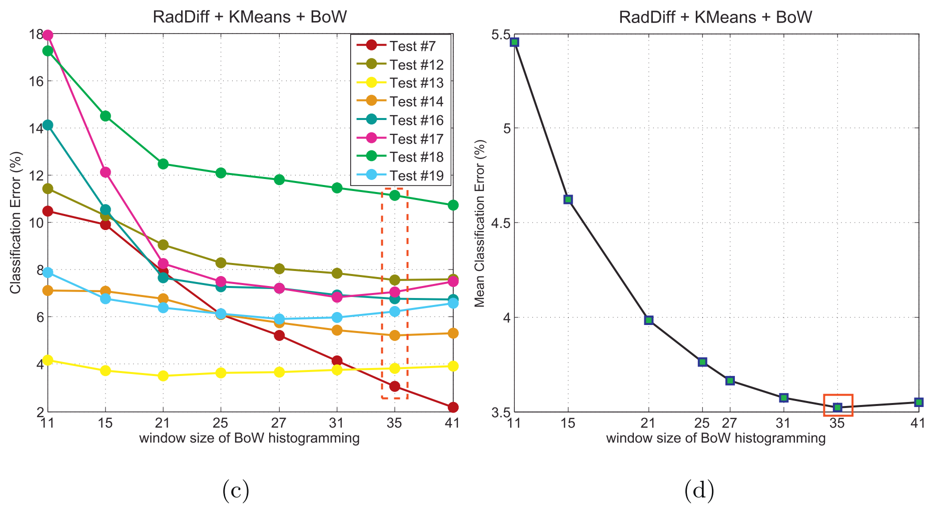


Fig. 6. Segmentation error rate of the Proposed Classifier (RadDiff + KMeans + BoW) as a function of the BoW histogram computation window size *w*.

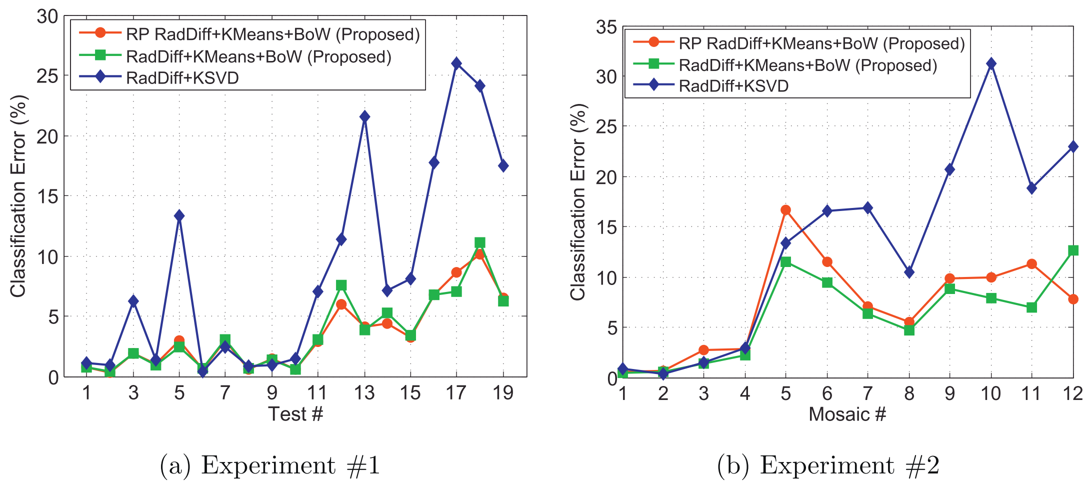


Fig. 7. Segmentation error rates (%) for the tests and mosaics in Experiments #1 and #2. The proposed Kmeans+BoW methods generally outperform sparse KSVD.

Table 5
Comparing the basic KMeans and proposed KMeans-BoW classifiers. Also comparing regular and random-projection features. The proposed KMeans-BoW approach offers a significant improvement, however randomized features do not.

Experiment # 1					Experiment # 2				
Descriptor	RadDiff		RP RadDiff		Descriptor	RadDiff		RP RadDiff	
Classifier	*K*Means	*K*Means + BoW	*K*Means	*K*Means + BoW	Classifier	*K*Means	*K*Means + BoW	*K*Means	*K*Means + BoW
Window size	$\nu = 37$	$\nu = 45$	$\nu = 37$	$\nu = 45$	Window size	$\nu = 43$	$\nu = 45$	$\nu = 43$	$\nu = 45$
Test # 1	0.86	0.76	1.20	0.78	Mosaic # 1	0.29	0.50	0.34	0.48
Test # 2	0.53	0.36	0.67	**0.35**	Mosaic # 2	0.39	0.60	0.37	0.58
Test # 3	2.07	1.90	2.33	1.90	Mosaic # 3	1.59	1.37	1.77	2.77
Test # 4	0.74	0.93	0.89	1.03	Mosaic # 4	2.36	2.23	3.37	2.89
Test # 5	10.77	2.40	7.52	2.99	Mosaic # 5	11.68	11.55	9.19	16.63
Test # 6	0.44	0.65	0.40	0.60	Mosaic # 6	10.88	9.43	12.24	11.53
Test # 7	5.14	3.05	8.25	2.94	Mosaic # 7	7.08	6.40	11.94	7.05
Test # 8	1.10	0.65	1.35	0.56	Mosaic # 8	6.04	4.73	8.35	5.57
Test # 9	1.31	1.36	1.30	1.48	Mosaic # 9	13.80	8.88	18.13	9.91
Test # 10	1.00	0.61	0.84	0.57	Mosaic # 10	17.88	7.88	24.07	9.97
Test # 11	2.49	3.05	3.23	2.86	Mosaic # 11	15.92	6.94	21.24	7.82
Test # 12	11.46	7.57	10.82	5.99	Mosaic # 12	19.65	12.61	28.46	13.88
Test # 13	6.11	3.82	8.11	4.12	Mosaic # 13	1.74	0.85	2.39	0.90
Test # 14	7.85	5.22	7.84	4.35	Mosaic # 14	1.68	1.44	2.15	2.04
Test # 15	3.08	3.40	3.91	3.22	Mosaic # 15	0.71	0.56	1.05	0.72
Test # 16	12.75	6.76	18.35	6.75					
Test # 17	9.91	7.06	10.48	8.61					
Test # 18	12.49	11.14	12.39	10.13					
Test # 19	7.39	6.22	8.93	6.53					
Average	5.13	3.52	5.73	3.46	Average	7.5	5.06	9.67	6.18

Table 6
Comparison of segmentation error rates (%) for three different feature extractions: random, dense, and sparse. The three tables test patch-based approaches and radial-difference methods with LMF and GC filtering. The inferiority of the sparse reconstruction approach (KSVD) is clear. Similar performance is seen with and without random projection (RP). Overall, the best results are found by the radial differences (RadDiff) with the proposed SingleBoW approach, using GC filtering.

Test	#1	#2	#3	#4	#5	#6	#7	#8	#9	#10	#11	#12	#13	#14	#15	#16	#17	#18	#19
(a) LMF filtering on patches, no image normalization																			
RP Patch + SingleBoW	0.82	**0.38**	3.31	**0.67**	17.08	0.58	1.79	1.71	1.28	**0.46**	3.21	**7.50**	7.32	**4.31**	**3.20**	13.42	10.24	**14.06**	7.76
Patch + SingleBoW	**0.68**	0.39	**2.29**	0.99	13.42	0.61	**1.59**	**1.19**	0.96	0.61	3.28	7.83	**5.98**	4.51	3.55	**13.38**	**9.63**	15.67	**6.61**
Patch + KSVD	0.97	0.55	2.59	0.69	**5.41**	**0.37**	26.64	1.23	**0.72**	0.69	**2.30**	10.26	8.44	9.35	3.35	15.63	12.38	18.07	10.08
(b) LMF filtering on RadDiff, image normalization applied																			
RP RadDiff + SingleBoW	0.78	**0.35**	**1.90**	1.03	2.99	0.60	2.94	**0.56**	1.48	**0.57**	**2.86**	**5.99**	4.12	**4.35**	**3.22**	**6.75**	8.61	**10.13**	6.53
RadDiff + SingleBoW	**0.76**	0.36	**1.90**	**0.93**	**2.40**	0.65	3.05	0.65	**1.36**	0.61	3.05	7.57	**3.82**	5.22	3.40	6.76	**7.06**	11.14	**6.22**
RadDiff + KSVD	1.20	1.14	6.24	1.48	13.64	**0.44**	**2.39**	1.20	**0.98**	1.46	7.57	11.59	22.68	8.18	8.67	18.22	25.18	23.70	15.95
(c) GC smoothing on RadDiff, image normalization applied																			
RP RadDiff + SingleBoW	0.58	0.24	2.06	**0.94**	2.34	**0.40**	**1.22**	0.19	0.95	**0.41**	**2.65**	**5.41**	4.89	3.94	**3.07**	**4.25**	7.38	**8.51**	7.07
λ_{GC}	(50)	(50)	(50)	(50)	(50)	(50)	(50)	(50)	(50)	(50)	(10)	(10)	(10)	(10)	(10)	(30)	(15)	(15)	(15)
RadDiff + SingleBoW	0.65	**0.15**	1.77	0.96	**2.16**	0.62	1.46	0.49	0.95	0.45	2.83	6.86	**4.22**	4.66	3.12	4.52	**6.93**	9.90	**6.59**
λ_{GC}	(50)	(50)	(50)	(50)	(50)	(50)	(50)	(50)	(50)	(50)	(10)	(10)	(10)	(10)	(10)	(30)	(15)	(15)	(15)
RadDiff + KSVD	**0.41**	0.40	**1.59**	1.19	25.40	0.40	1.77	**0.17**	**0.50**	0.50	8.56	5.74	21.28	**3.89**	6.36	10.37	20.11	22.81	12.09
λ_{GC}	(50)	(50)	(50)	(50)	(50)	(50)	(50)	(50)	(50)	(50)	(5)	(5)	(5)	(5)	(5)	(3)	(3)	(3)	(3)

Table 6(b) and (c) compares the results for the two post-processing approaches, with GC outperforming LMF. Nevertheless the window size parameter ν in LMF was fixed, in contrast the selection of GC parameter λ_{GC} was a function of the textures in the test image. On the whole, the conclusion is that the proposed radial-difference with a Kmeans/SingleBoW approach with GC filtering outperforms the other variations tested.

4.2.5. Evaluation of running time

The implementation of our texture segmentation framework consists of the following major stages: local simple feature computation, *K*Means clustering, and single BoW model training. All components of our system are implemented in MATLAB and run on a computer with a 2.7 GHz Intel Core2 Duo CPU and 2 GB of RAM. Table 7 reports running times obtained for the representative set of tests. We can see clearly from the tables that the proposed approach is very fast relative to sparse reconstruction; the computational simplicity of the proposed approach lies in the simple and efficient single-BoW strategy.

Table 7
Comparisons of computation times, in seconds; all methods implemented in MATLAB. Parameter settings for KSVD are kept as in [25]; the number of iterations in each dictionary learning method is set to 20, as in [2,25]. All results are obtained with a two-core, 2G memory and 2.7 GHz GPU desktop. The random projection approach offers slightly faster processing because of dimensionality reduction, however all of the proposed radial-difference (RD) approaches are much faster than sparse KSVD.

			Run time in Matlab (seconds) (with both training and testing included.)				
	Size	Classes	RPRD + KMeans	RPRD + KMeans + BoW	RD + KMeans	RD + KMeans + BoW	RD + KSVD
Test # 1	512 × 512	4	21.7	82.7	38.9	99.6	4315.1
Test # 11	512 × 512	16	73.8	292.3	138.8	366.9	17363.1
Test # 16	512 × 512	16	73.7	334.7	138.9	398.4	17388.9
Test # 17	256 × 512	8	34.6	104.8	66.8	138.7	7791.9
Mosaic # 1	256 × 512	2	10.9	37.3	19.3	45.7	1967.4
Mosaic # 4	256 × 256	5	22.0	49.7	42.3	72.2	4600.4
Mosaic # 9	256 × 640	10	46.4	154.4	87.2	202.4	9850.9
Mosaic # 11	256 × 512	16	77.4	338.9	142.4	417.9	17338.2

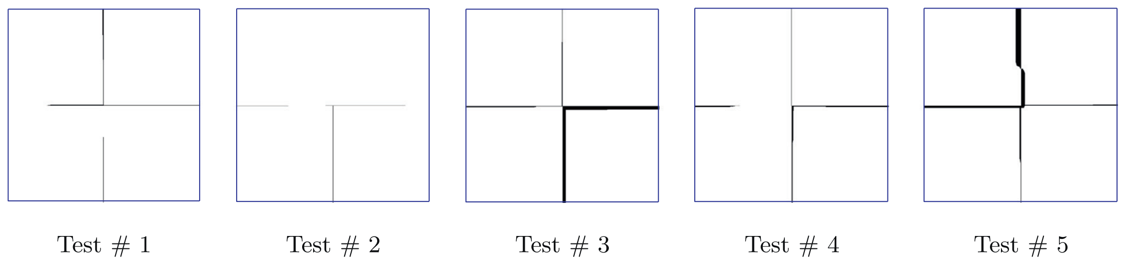


Fig. 8. Visualization of the final segmentation error maps for the proposed segmentation, with GC post processing.

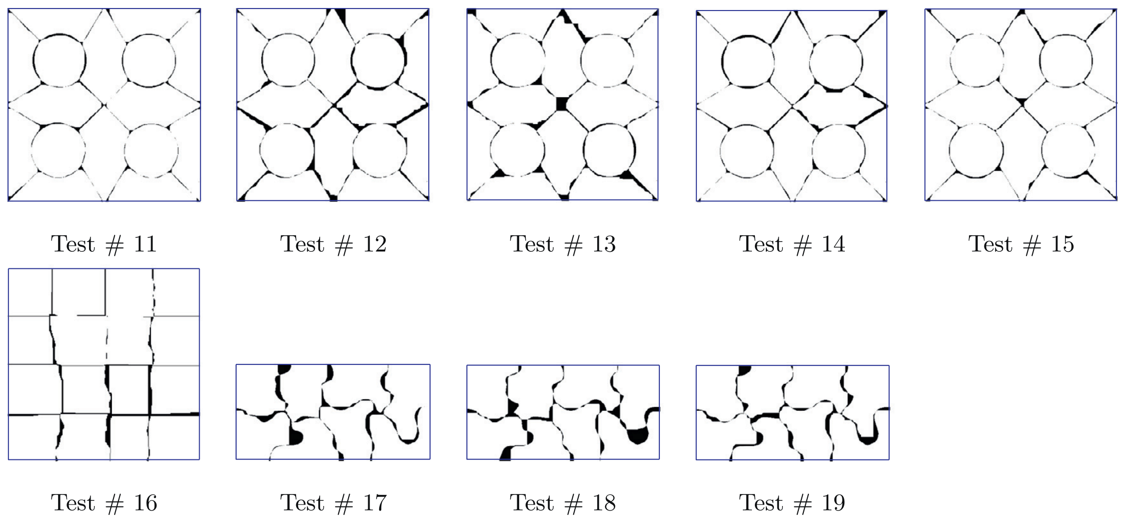


Fig. 9. Segmentation error maps for the proposed segmentation with GC post processing.

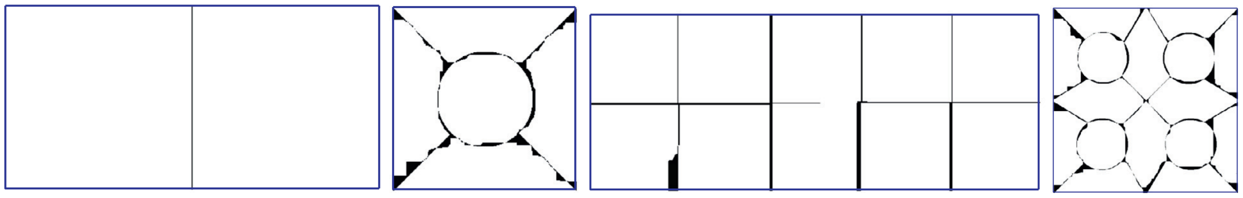
The segmentation results most of the tests in Experiment #1 are visualized in Figs. 8 and 9. We can observe that the segmentation results are very good, even those of the sixteen-class texture mosaics, with the misclassified pixels concentrated at the region boundaries.

4.3. Comparative evaluation

Motivated by the comprehensive experimental evaluation and the striking classification performance of the proposed approach in the previous section, in this section we wish to evaluate our proposed approach on the benchmark datasets which have been used by several researchers to evaluate the performance of texture segmentation. Specifically we will focus on the test mosaics in Experiment #2.

Table 8
Texture segmentation performance compared to existing state of the art segmentation algorithms on Mosaics # 1 – # 12. Results are error rates in percentage. The "Best of [34]" column lists some of the best classification results from [34], where dozens of texture features were evaluated. For the proposed method with LMF, all of the parameters are just set to their defaults, nevertheless nearly outperforming nearly every other method. The results obtained with GC post processing do have parameter λ_{GC} varying with mosaic.

Mosaic	#1	#2	#3	#4	#5	#6	#7	#8	#9	#10	#11	#12	Average
Best of [34]	0.7	0.2	2.5	7.2	18.9	20.6	16.8	17.2	32.3	27.8	34.7	41.7	18.38
LBP [28]	0.3	1.0	10.6	6.0	18.0	12.1	9.7	11.4	22.7	19.4	17.0	20.7	12.41
FTCM [39]	NA	NA	NA	5.5	7.3	13.2	5.6	10.5	18.9	21.4	17.1	17.2	NA
[21]	0.36	1.33	1.14	3.37	16.05	13.03	6.62	8.15	21.96	9.61	18.66	21.67	10.16
R (Gaussian) [25]	0.35	0.58	1.36	2.22	24.66	10.20	6.66	5.26	13.27	18.85	16.88	19.32	9.97
R (GC) [25]	**0.17**	0.73	0.37	1.69	36.5	5.49	4.60	4.32	11.80	21.88	15.50	21.89	10.41
D (Gaussian) [25]	0.20	**0.41**	1.97	1.89	16.38	9.11	3.79	5.10	14.77	10.12	12.91	11.44	7.34
D (GC) [25]	**0.17**	0.60	0.78	1.61	16.42	**4.15**	**3.67**	4.58	**2.24**	**2.04**	9.04	**8.80**	4.51
Proposed: RadDiff (LMF)	0.50	0.55	1.37	2.23	11.55	9.43	6.40	4.73	8.88	7.88	6.54	11.88	5.96
Proposed: RadDiff (GC)	0.19	0.60	**0.20**	**1.36**	**5.20**	9.17	5.55	**3.71**	**2.24**	2.94	**5.78**	9.90	**3.90**



(a) Mosaic # 3 (b) Mosaic # 5 (c) Mosaic # 10 (d) Mosaic # 11

Fig. 10. The mislabeled pixels in the segmentation of some mosaics in Experiment # 2. The RadDiff feature with GC post processing are used.

Table 8 shows the classification errors, given as a percentage of wrongly classified pixels, comparing our proposed approach with published results on Mosaics #1 to #12. The best classification results from [34][1] are shown in the first row of Table 8. The same mosaics were also used in other work [21,25,28,39], and results from these are also included as comparison. As can be seen from Table 8, our proposed approach "RadDiff (GC)" performs very well, the best overall averaged result.

Even the improvement of the proposed approach ("RadDiff (LMF)") over the methods presented in [21,25,28,39] and the pure sparse reconstruction methods "R (Gaussian)" and "R (GC)" [25] is noteworthy, particularly because they have utilized filtering methods more sophisticated than LMF. That is, "RadDiff (LMF)" is using a *single* feature extraction method (radial differences), a single, constant set of parameters, and a very simple post smoothing method (LMF), and nevertheless outperforming most of the compared approaches, showing the strength of the underlying RadDiff+*K*Means+SingleBoW approach.

Regarding to the relative performance of our proposed approach and discriminative sparse reconstruction ("D (Gaussian)" and "D (GC)") [25], of the simpler post-filtering approaches, Table 8 shows that "RadDiff (LMF)" outperforms "D (Gaussian)", whereas with GC filtering, "RadDiff (GC)" outperforms "D (GC)". Note that the discriminative reconstruction approaches do depend on some parameters tuning for each test mosaic [25], including the complexity of the sparse model, the required sparsity level, the properness of a sparse reconstruction algorithm and the trade-off between the reconstruction and discriminative terms. Moreover, the computational complexity of the discriminate sparse approach is higher than that of the pure sparse reconstruction, whose computational complexity was shown in Table 7.

The results of our experiments suggest that no advantage is gained by imposing sparsity at run-time, and that a computationally expensive reconstruction procedure via optimization can therefore be replaced by our proposed efficient approach.

Although RadDiff GC slightly outperforms D (GC), on average, the results for the individual test mosaics can vary significantly. One can observe that the proposed approach significantly improves the classification rate for mosaics #5 and #11, but that D (GC) performs better on mosaics #6 and #7. A few qualitative results for the proposed approach are visualized in Fig. 10.

Finally, in terms of the images Mosaics #13 to #15, Table 9 compares our proposed approach with those of the state of the art approach proposed in [32]. The table shows that both RadDiff-LMF and RadDiff-GC significantly outperform the method proposed in [32].

[1] These best results for all the twelve test mosaics are achieved by different approaches, not achieved by a single feature extraction method. For Mosaics #1 through #12, the best result was achived, respectively, by the following: "Opt. repr. Gabor filter bank", "f16b (d) (full rate)" "F_2_1_smpl (d) (full rate)", "f32d (d) (full rate)", "f16b (d) (full rate)", "JMS", "JU", "JF", "f32d (a) (full rate)", "DCT", "Dyadic Gabor filter bank", "F_2_1_smpl (d) (full rate)", "Prediction error filter" and "f16b (d) (full rate)".

Table 9
Texture segmentation performance compared to the recent work of [32] on Mosaics # 13 – # 15.

Method	Mosaic #13	Mosaic #14	Mosaic #15
Rad-Diff ($\lambda_{GC} = 50$)	**0.10**	**0.20**	**0.30**
Rad-Diff + BoW (LMF)	0.85	1.44	0.56
MIRGS [32]	1.09	1.89	4.86

5. Conclusions

We have proposed a conceptually simple and highly efficient BoW approach that can effectively segment natural texture images having complex content.

Compared to multiple state of the art segmentation methods, the proposed method produces superior segmentations, with modest complexity. The low computational complexity of the proposed method is due to the following factors: (1) a local radial difference feature which is very fast to compute; (2) efficient *K*Means dictionary learning compared with the *K*SVD method; (3) the new single BoW histogram learning method which significantly reduces the computational burden, compared with traditional model selection such as *k*medoids or greedy methods.

There is no theoretical or empirical reason to expect that sparse reconstruction approaches will improve texture segmentation accuracy or robustness, and indeed the extensive experiments carried out in this paper clearly demonstrate this point. Given the high computational burden involved in sparse coding, this conclusion may impact design strategies for texture descriptors.

Possible ways of effectively extending the proposed approach to object recognition and scene segmentation is the topic of future work.

Acknowledgments

This work has been supported by the National Natural Science Foundation of China under contract No. 61202336.

References

[1] D. Achlioptas, Database-friendly random projections, in: Proceedings of the Twentieth ACM Symposium on Principles of Database Systems, 2001, pp. 274–281.
[2] M. Aharon, M. Elad, A. Bruckstein, *K*SVD: an algorithm for designing overcomplete dictionries for sparse representation, IEEE Trans. Signal Process. 54 (1) (2006) 4311–4322.
[3] R. Baraniuk, M. Davenport, R. DeVore, M. Wakin, A simple proof of the restricted isometry property for random matrices, Constructive Approximation 28 (3) (2008) 53–263.
[4] Y. Boykov, V. Kolmogorov, An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision, IEEE Trans. Pattern Anal. Mach. Intell. 26 (9) (2004) 1124–1137.
[5] Y. Boykov, O. Veksler, R. Zabih, Fast approximate energy minimization via graph cuts, IEEE Trans. Pattern Anal. Mach. Intell. 23 (11) (2001) 1222–1239.
[6] P. Brodatz, Texture: A Photographic Album for Artists and Designers, Dover, New York, 1966.
[7] E.J. Candés, T. Tao, Decoding by linear programming, IEEE Inform. Theory. 51 (12) (2005) 4203–4215.
[8] E.J. Candés, T. Tao, Near-optimal signal recovery from random projections: universal encoding stratigies? IEEE Inform. Theory. 52 (12) (2006) 5406–5425.
[9] G. Cross, A.K. Jain, Markov random field texture models, IEEE Trans. Pattern Anal. Mach. Intell. 5 (1) (1983) 25–39.
[10] S. Dasgupta, Learning mixture of gaussians, University of California at Berkeley, 2000 Ph.D thesis.
[11] I. Dhillon, S. Mallela, R. Kumar, A divisive information-theoretic feature clustering algorithm for text classification, J. Mach. Learn. Res. 3 (1) (2003) 1265–1287.
[12] C. Ding, J. Choi, D. Tao, L. Davis, Multi-directional multi-level dual-cross patterns for robust face recognition, IEEE Trans. Pattern Anal. Mach.Intell. 38 (3) (2016) 518–531.
[13] D.L. Donoho, Compressed sensing, IEEE Inform. Theory. 52 (4) (2006) 1289–1306.
[14] R.O. Duda, P.E. Hart, D.G. Stork, Pattern Classification, Second ed., John Wiley and Sons, 2001.
[15] Z. Jiang, Z. Lin, L.S. Davis, Learning a discriminative dictionary for sparse coding via label consistent k-svd, in: International Conference on Computer Vision and Pattern Recognition (CVPR), 2011, pp. 1697–1704.
[16] W.B. Johnson, J. Lindenstrauss, Extensions of lipschitz mappings into a hilbert space, in: International Conference on Modern Analysis and Probability, 1984, pp. 189–206.
[17] V. Kolmogorov, R. Zabih, What energy functions can be minimized via graph cuts? IEEE Trans. Pattern Anal. Mach. Intell. 26 (2) (2004) 147–159.
[18] S. Lazebnik, C. Schmid, J. Ponce, A sparse texture representation using local affine regions, IEEE Trans. Pattern Anal. Mach. Intell. 27 (8) (2005) 1265–1278.
[19] F. Lehmann, Turbo segmentation of textured images, IEEE Trans. Pattern Anal. Mach. Intell. 30 (1) (2011) 16–29.
[20] T. Leung, J. Malik, Representing and recognizing the visual appearance of materials using three-dimensional textons, Int. J. Comput. Vision 43 (1) (2001) 29–44.
[21] A. Lillo, G. Motta, J. Storer, Texture classification based on discriminative features extracted in the frequency domain, Int. Conf Image Process.(ICIP) 2 (2007) 53–56.
[22] L. Liu, P. Fieguth, Texture classification from random features, IEEE Trans. Pattern Anal. Mach. Intell. 34 (3) (2012) 574–586.
[23] L. Liu, P. Fieguth, G. Kuang, D. Clausi, Sorted random projections for robust rotation invariant texture classification, Pattern Recognit. 45 (6) (2012) 2405–2418.
[24] L. Liu, P. Fieguth, G. Zhao, M. Pietikäinen, D. Hu, Extended local binary patterns for face recognition, Inf. Sci. 358 (1) (2016) 358–359.
[25] J. Mairal, F. Bach, J. Ponce, G. Sapiro, A. Zisserman, Discriminative learned dictionaries for local image analysis, in: International Conference on Computer Vision and Pattern Recognition (CVPR), 2008a, pp. 1–8.

[26] J. Mairal, F. Bach, J. Ponce, G. Sapiro, A. Zisserman, Supervised dictionary learning, Neural Information Processing Systems (NIPS), 2008b.
[27] T. Ojala, M. Pietikäinen, T. Mäenpää, Multiresolution gray-scale and rotation invariant texture classification with local binary patterns, IEEE Trans. Pattern Anal. Mach. Intell. 24 (7) (2002) 971–987.
[28] T. Ojala, K. Valkealahti, E. Oja, M. Pietikäinen, Texture discrimination with multidimensional distributions of signed gray-level differences, Pattern Recognit. 34 (3) (2001) 727–739.
[29] B.A. Olshausen, D.J. Field, Sparse coding with an overcomplete basis set: a strategy employed by v1? Vis. Res. 37 (1997) 3311–3325.
[30] M. Petrou, C. Petrou, Image Processing: The Fundamentals, Wiley, 2010.
[31] G. Peyré, Sparse modeling of textures, J. Math. Imaging Vis. 34 (1) (2009) 17–31.
[32] A. Qin, D. Clausi, Multivariate image segmentation using semantic region growing with adaptive edge penalty, IEEE Trans. Image Process. 19 (8) (2010) 2157–2170.
[33] I. Ramirez, P. Sprechmann, G. Sapiro, Classification and clustering via dictionary learning with structured incoherence and shared features, in: International Conference on Computer Vision and Pattern Recognition (CVPR), 2010, pp. 3501–3508.
[34] T. Randen, J. Husøy, Filtering for texture classification: a comparative study, IEEE Trans. Pattern Anal. Mach. Intell. 21 (4) (1999) 291–310.
[35] T.R. Reed, J.M.H.D. Buf, A review of recent texture segmentation and feature extraction techniques, CVGIP: Image Understanding 57 (3) (1993) 359–372.
[36] R. Rigamonti, M. Brown, V. Lepetit, Are sparse representations really relevant for image classification? in: International Conference on Computer Vision and Pattern Recognition (CVPR), 2011, pp. 1545–1552.
[37] Y. Rubner, J. Puzicha, C. Tomasi, J.M. Buhmann, Empirical evaluation of dissimilarity measures for color and texture, Comput. Vis. and Image Und. 84 (2001) 25–43.
[38] Q. Shi, A. Eriksson, A. Hengel, C. Shen, Is face recognition really a compressive sensing problem? in: International Conference on Computer Vision and Pattern Recognition (CVPR), 2011, pp. 553–560.
[39] K. Skretting, J. Husøy, Texture classification using sparse frame-based representations, EURASIP J. Appl. Signal Process. 2006 (1) (2006) 1–11.
[40] M. Tuceryan, A.K. Jain, Texture analysis, in: C. Chen, L. Pau, P. Wang (Eds.), Handbook Pattern Recognition and Computer Vision, World Scientific, Singapore, 1993, pp. 235–276. Ch. 2
[41] M. Tuceryan, A. Jain, Texture segmentation using voronoi polygons, IEEE Trans. Pattern Anal. Mach. Intell. 12 (2) (1990) 211–216.
[42] M. Varma, A. Zisserman, A statistical approach to material classification using image patches, IEEE Trans. Pattern Anal. Mach. Intell. 31 (11) (2009) 2032–2047.
[43] M. Varma, A. Zisserman, A statistical approach to texture classification from single images, Int. J. Comput. Vision. 62 (1–2) (2005) 61–81.
[44] J. Wright, A.Y. Yang, A. Ganesh, S.S. Sastry, Y. Ma, Robust face recognition via sparse representation, IEEE Trans. Pattern Anal. Mach. Intell. 31 (2) (2009) 210–227.
[45] X. Xie, M. Mirmehdi, A galaxy of texture features, in: M. Mirmehdi, X. Xie, J. Suri (Eds.), Handbook of Texture Analysis, Imperial College Press, 2008, pp. 375–406.
[46] Y. Xu, X. Yang, H. Ling, H. Ji, A new texture descriptor using multifractal analysis in multi-orientation wavelet pyradmid, in: International Conference on Computer Vision and Pattern Recognition (CVPR), 2010, pp. 161–168.
[47] J. Yu, R. Hong, M. Wang, J. You, Image clustering based on sparse patch alignment framework, Pattern Recogn. 47 (11) (2014a) 3512–3519.
[48] J. Yu, Y. Rui, Y. Tang, D. Tao, High-order distance-based multiview stochastic learning in image classification, IEEE Trans. Cybern 44 (12) (2014b) 2431–2442.
[49] J. Yu, Y. Rui, D. Tao, Click prediction for web image reranking using multimodal sparse coding, IEEE Trans. Cybern. 23 (5) (2014c) 2019–2032.
[50] Q. Zhang, B. Li, Discriminative KSVD for dictionary learning in face recognition, in: International Conference on Computer Vision and Pattern Recognition (CVPR), 2010, pp. 2691–2698.
[51] J. Zhang, M. Marszalek, S. Lazebnik, C. Schmid, Local features and kernels for classification of texture and object categories: a comprehensive study, Int. J. Comput. Vision 73 (2) (2007) 213–238.
[52] The vistex database, http://vismod.media.mit.edu/vismod/imagery/VisionTexture/vistex.html.