Model-Based Learning of Local Image Features for Unsupervised Texture Segmentation

Martin Kiechle, Martin Storath, Andreas Weinmann, and Martin Kleinsteuber

***Abstract*— Features that capture well the textural patterns of a certain class of images are crucial for the performance of texture segmentation methods. The manual selection of features or designing new ones can be a tedious task. Therefore, it is desirable to automatically adapt the features to a certain image or class of images. Typically, this requires a large set of training images with similar textures and ground truth segmentation. In this paper, we propose a framework to learn features for texture segmentation when no such training data is available. The cost function for our learning process is constructed to match a commonly used segmentation model, the piecewise constant Mumford-Shah model. This means that the features are learned such that they provide an approximately piecewise constant feature image with a small jump set. Based on this idea, we develop a two-stage algorithm which first learns suitable convolutional features and then performs segmentation. We note that the features can be learned from a small set of images, from a single image, or even from image patches. The proposed method achieves a competitive rank in the Prague texture segmentation benchmark, and it is effective for segmenting histological images.**

***Index Terms*— Texture segmentation, feature vector, geometric optimization, Mumford-Shah model, unsupervised learning.**

I. Introduction

TEXTURE segmentation is a frequently occurring and challenging problem in image processing and computer vision. For textured images – such as many natural images [1], [2], histological images [3], or crystal structures [4] – the segmentation is typically performed in two stages. In the first stage, a (vector-valued) feature image is derived from the image. The corresponding features are designed to capture the local statistical properties or oscillatory patterns of a texture. Many classical features are based on linear filters [5], for example Gabor filters [6], wavelet frames [7], windowed Fourier transform [8], followed by a pointwise nonlinearity [9]. Other popular features are based on local spectral histograms [10], morphological filters [11], local statistical descriptors [12] or local binary patterns [13]. In the second stage, the feature image is segmented. Popular choices include k-means clustering [6], [7] or mean shift algorithms [14]. More sophisticated (variational) segmentation models additionally enforce spatial regularity of the segment boundaries: here, a prominent example is the piecewise constant Mumford-Shah model (or Potts model) [15], [16]; it has been used for texture segmentation, for instance in [4] and [17]-[19].

A. Motivation and Related Work

Besides the aforementioned works, there is a series of more recent contributions to unsupervised texture segmentation: Todorovic and Ahuja create a tessellation of texture superpixels (texels) and cluster them by a multiscale segmentation and a meanshift algorithm [12]. Galun *et al.* [20] utilize a multiscale aggregation of filter responses and shape elements. Haindl and Mikes employ a Gaussian MRF texture model [21] or a 3D auto regressive model [22], and they perform segmentation based on a Gaussian mixture model. Scarpa *et al.* [23] use features based on Markov chains, and then segment by recursively merging them according to their mutual interaction. Yuan *et al.* [24] use local spectral histograms as feature vectors and formulate the segmentation problem as a multivariate linear regression. In a follow-up work [25], non-negative matrix factorization is used for segmentation. Storath *et al.* [19] utilize monogenic curvelets as features and perform segmentation based on the piecewise constant Mumford-Shah model. In contrast to this work, the features in [19] are computed from a fixed system of handcrafted filters which are not learned. The method of Panagiotakis *et al.* [26], [27] is based on voting of blocks, Bayesian flooding and region merging. Mevenkamp and Berkels [4] use local Fourier features, which are tailored to images with crystal structures, and segment using a convex relaxation of the piecewise constant Mumford-Shah model. McCann *et al.* [3] utilize features derived from local histograms, and segment using nonnegative matrix factorization and image deconvolution.

It is a fundamental issue that the performance of the features depends strongly on the class of images or even on the

Manuscript received August 2, 2016; revised March 20, 2017 and October 29, 2017; accepted January 4, 2018. Date of publication January 12, 2018; date of current version January 26, 2018. This work was supported by the German Research Foundation through Grant KL 2189/9-1, Grant STO1126/2-1, and Grant WE5886/4-1. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Chunming Li. *(Corresponding author: Martin Kiechle.)*

M. Kiechle is with the Department of Electrical and Computer Engineering, Technical University of Munich, 80333 München, Germany (e-mail: martin.kiechle@tum.de).

M. Storath is with the Image Analysis and Learning Group, Universität Heidelberg, 69117 Heidelberg, Germany (e-mail: martin.storath@iwr.uni-heidelberg.de).

A. Weinmann is with the Department of Mathematics and Natural Sciences, Darmstadt University of Applied Sciences, 64295 Darmstadt, Germany, and also with the Institute of Computational Biology, Helmholtz Zentrum München, 85764 Neuherberg, Germany (e-mail: andreas.weinmann@h-da.de).

M. Kleinsteuber is with the Department of Electrical and Computer Engineering, Technical University of Munich, 80333 München, Germany, and also with Mercateo AG, 80331 Munich, Germany (e-mail: kleinsteuber@tum.de).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIP.2018.2792904

single image. For instance, good features for a natural image may perform poorly on a histological image. Even more, good features for one natural image may not perform as well on another natural image. Thus, the design of the features is a critical task and there are several approaches to this. A straightforward idea is to simply increase the number of features hoping that at least some features are well suited for the texture patterns of the processed image. Unfortunately, the computational effort for segmenting large feature spaces is very high in practice, in particular for segmentation methods which enforce regularity of the boundaries. To circumvent these problems, a commonly used strategy is to manually select a subset from the aforementioned larger set of features; see for example [18], [25]. However, the manual selection requires human supervision which typically results in an expensive, time-consuming task. In principle, for each new class of images one should reevaluate this selection. To avoid manual design of features for each image or each class of images, it seems natural to learn them from data. In a supervised learning setup, where a sufficiently large training set of images with similar characteristics and a ground truth segmentation is available, one can use generic methods; for example the super-pixelation based method of [28] or more recent methods based on convolutional neural networks [29]. In the present unsupervised setup, such a training set is not available. As a consequence, the challenge is to find a suitable objective function for the learning task and a practical numerical procedure to optimize the features accordingly.

B. Contribution

In this work, we develop a method for unsupervised texture segmentation where the features are learned from non-annotated data, i.e., from images without ground truth segmentation. The main contributions of this work are *(i)* a model for feature learning of image features for texture segmentation in the absence of annotated training data, and *(ii)* a practical algorithm for unsupervised texture segmentation based on that model.

Regarding the basic model *(i)*, our starting point is the observation that features are often designed such that the feature image is approximately constant on a texture segment. This allows utilizing segmentation algorithms based on a local homogeneity assumption. The basic idea of our model is to learn convolutional features in a way that they produce approximately piecewise constant feature images. Besides reasonable constraints on the filters, such as their norm and mutual coherence, the objective is to minimize the cost function of the popular piecewise constant Mumford-Shah segmentation model, i.e. the total length of the discontinuity set of the corresponding feature image. Regarding *(ii)*, learning filters based on the proposed model turns out to be a challenging optimization problem because it involves a non-smooth and non-convex cost function on the (non-convex) unit sphere. To make it computationally tractable, we decompose and relax this model: we obtain the two stages of filter learning and of segmentation. For the relaxed learning stage, we employ a smooth (yet non-convex) approximation of the cost function. To minimize this cost function, we adapted a geometric conjugate gradient descent method proposed in [30] and [31] such that it fits with the proposed model. For the segmentation stage, we employ the Lagrange formulation of the piecewise constant Mumford-Shah model. In particular, implied by the model, we consider a data term based on the Mahalanobis distance. To solve the corresponding problem, we extended the approach proposed in [32] and [33] in order to be able to deal with the Mahalanobis distance. Finally, a post-processing as in [25] merges small spurious regions to large ones.

We evaluate our method on different types of textured images. A standard benchmark for texture-based segmentation is the Prague texture segmentation benchmark [34]. Here, our method achieves a top rank. In particular, the proposed method gives significantly better results than many earlier methods [12], [20]–[24], and slightly better results than the more recent methods proposed in [4] and [25]. Further, we are competitive with the currently leading method PMCFA [26], [27]. Besides, our approach provides satisfactory segmentation results on the data set of histological images of [3]. We emphasize that, although this is a quite different image class, only minor adjustments have been necessary. This shows in particular the flexibility of our method, and the potential for segmenting quite different classes of textured images.

II. A Model for Unsupervised Filter Learning for Texture Segmentation

As mentioned in the introduction, our goal is to learn suitable features for texture segmentation when no training data with ground truth is available. Here, we focus on learning convolutional features. Convolutional filters are a natural choice because they describe the class of linear translation-invariant filters. A feature image is created by applying linear filtering followed by a (pointwise) nonlinear transform. More precisely, given an image $\mathbf{U} \in \mathbb{R}^{M\times N}$, we consider K different convolution filters $\mathbf{\Phi}_1, \ldots, \mathbf{\Phi}_K$, and the resulting filtered images, given in Matlab-type notation by

$$\mathcal{F}_{:,:,1} = \mathbf{\Phi}_1\mathbf{U}, \quad \ldots, \quad \mathcal{F}_{:,:,K} = \mathbf{\Phi}_K\mathbf{U}.$$

In short-hand notation, we write $\mathcal{F} = \mathbf{\Phi}\mathbf{U}$. Then, to each filter response, the same nonlinear transformation σ is applied pixel wise. In general, σ is chosen to be symmetric, i.e. $\sigma(x) = \sigma(-x)$. Further, it is required that it has fast decaying slope for large x in order to be robust towards outliers in the filter responses. The nonlinear transform has proven to be beneficial for texture segmentation: according to [9], its purpose is to translate differences in dispersion characteristics into differences in mean value. For further details on choosing σ we refer to [9]. In this paper we use a logarithmic non-linearity of the form $\sigma(x) := \log(1 + \mu x^2)$ with the free parameter $\mu > 0$. The nonlinear transform is considered to be fixed, and we are interested in finding suitable linear convolution operators $\mathbf{\Phi}_1, \ldots, \mathbf{\Phi}_K$, which define the features

$$\mathcal{V} = \sigma(\mathbf{\Phi}\mathbf{U}).$$

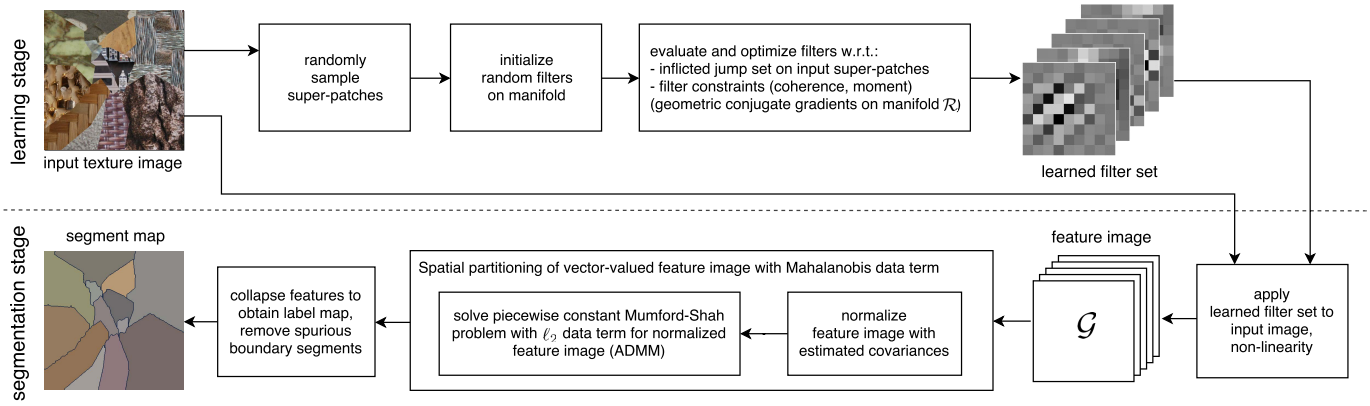


Fig. 1. Conceptual schematic of the proposed method with its learning and segmentation stages.

Here, $\mathcal{V}$ and $\mathbf{\Phi U}$ are three dimensional arrays in $\mathbb{R}^{M\times N\times K}$, and σ has to be understood as componentwise application of its scalar version.

Since we are in an unsupervised setup, we have no training data (i.e. no ground truth segmentation) for learning the $\mathbf{\Phi}_1, \ldots, \mathbf{\Phi}_K$ from. In particular, there is no straightforward way to devise a loss function for the learning process. We propose to utilize a loss function based on the segmentation model, which in our case is the piecewise constant Mumford Shah or Potts model: ideally, the features $\mathcal{V}$ are approximately constant on the texture, and the segment boundaries are sufficiently regular. The idea is to learn suitable filters $\mathbf{\Phi}$ in a way such that their responses (after applying the non-linearity) on the segments are approximately constant. We propose to minimize as a cost function the length of the discontinuity set of $\mathcal{V}$, denoted by $\|\nabla\mathcal{V}\|_0$. More precisely, we propose as a model for choosing the convolution kernels $\mathbf{\Phi}_1, \ldots, \mathbf{\Phi}_K$,

$$\min_{\mathcal{V},\Phi} \|\nabla\mathcal{V}\|_0 \quad \text{subject to } d(\mathcal{V}, \sigma(\mathbf{\Phi U})) \leq \varepsilon, \tag{1}$$

with $\varepsilon > 0$. Here, the minimum is taken with respect to both $\mathbf{\Phi}, \mathcal{V}$, where the $\mathbf{\Phi}_k$ have unit length, zero mean, and fulfill an incoherence and a certain center condition. (We elaborate on these constraints in Section III-D.) The symbol d denotes a metric, in our case the Mahalanobis distance as explained in Section IV. We note that an optimal pair $\mathbf{\Phi}^*, \mathcal{V}^*$ of Eq. (1) already consists of an optimal filter bank $\mathbf{\Phi}^*$ together with a corresponding optimal segmentation $\mathcal{V}^*$.

The model Eq. (1) is computationally hard to access. In particular, the simultaneous optimization w.r.t. both Φ and $\mathcal{V}$ is extremely demanding. As an approximative strategy, we propose a two stage approach as follows. As a first step, we optimize the filters $\mathbf{\Phi}$ using a relaxation of Eq. (1) as described in Section III. For the second step, we notice, that for fixed $\mathbf{\Phi}$, the Lagrange form of Eq. (1) is the piecewise constant Mumford-Shah model. Therefore, we perform a piecewise constant Mumford-Shah segmentation w.r.t. the Mahalanobis distance (described in Section IV) for the obtained feature image. We note that even this second step of solving the piecewise constant Mumford-Shah problem is known to be an NP hard problem on its own.

Figure 1 illustrates the conceptual flow of the proposed method with its filter learning and segmentation stages. For notational brevity, we describe the derivation of our method on gray-valued images $\mathbf{U} \in \mathbb{R}^{M\times N}$. The derivation for multi-channel images follows the same basic steps. The relevant modifications regarding the operators $\mathbf{\Phi}$ and the jump penalty are described in Section III-F.

III. Learning Stage

In this section, we discuss how to learn the filters $\mathbf{\Phi}$ from a given image. As a first step, we present a near anisotropic discretization of the jump penalty Eq. (1) in Section III-A. Then, we relax the model Eq. (1) to obtain a computationally better accessible surrogate problem to perform the learning task in Section III-B. Further, we incorporate learning from patch samples in Section III-C, and explain how to deal with the constraints imposed on the filters in Section III-D, respectively. Then, we sum up the simplified learning problem and discuss its numerics in Section III-E. Finally, we explain how to generalize the approach for multi-channel images in Section III-F.

As pointed out, we focus on sets of linear filters. Further, we assume that each filter has a fixed number of n coefficients $\mathbf{\Phi}_k \in \mathbb{R}^n$.

A. Near Isotropic Discretization

First, we deal with a near isotropic discretization of the jump penalty $\|\nabla\mathcal{V}\|_0$. As in [33], we use a finite difference discretization of the form

$$\|\nabla\mathcal{V}\|_0 = \sum_{s=1}^{S} \omega_s \|\nabla_{a_s}\mathcal{V}\|_0. \tag{2}$$

The vectors $a_s \in \mathbb{Z}^2 \setminus \{0\}$ belong to a finite difference system $\mathcal{N}$ with $S \geq 2$ elements. For $a \in \mathbb{Z}^2$, we let

$$\|\nabla_a\mathcal{V}\|_0 = |\{i = (i_1, i_2) : |\mathcal{V}_{i,:} - \mathcal{V}_{i+a,:}|_2 \neq 0\}|. \tag{3}$$

where we use the notation $\mathcal{V}_{i,:} = (\mathcal{V}_{i,1}, \ldots, \mathcal{V}_{i,K}) \in \mathbb{R}^K$ to denote the data located in the pixel with coordinates $i \in \mathbb{Z}^2$. Further, we use the symbol $|x|_2$ to denote the Euclidean norm $|x|_2 = (\sum_j x_j^2)^{1/2}$. Here we use an eight-connected neighborhood represented by the finite difference system

$$\mathcal{N} = \{(1, 0), (0, 1), (1, 1), (1, -1)\}. \tag{4}$$

with the weights $\omega_{1/2} = \sqrt{2} - 1$ and $\omega_{3/4} = 1 - \frac{\sqrt{2}}{2}$. For details, we refer to [33] and [35]..

B. Relaxation

Since solving Eq. (1) is computationally extremely hard, we impose the following simplifications to make it tractable: For the feature learning part, we propose to replace the strict ℓ_0 term in Eq. (3) by the smooth non-convex sparsity promoting surrogate function

$$\|\nabla_a \mathcal{V}\|_{0,\nu} = \sum_i \log(1 + \nu |\mathcal{V}_{i,:} - \mathcal{V}_{i+a,:}|_2^2), \tag{5}$$

which is a good approximation of the jump penalty with equality in the limit of its parameter ν, cf. [31]. Further, we let $\varepsilon = 0$ in Eq. (1) which leads to minimizing the (preliminary) cost function

$$f(\mathbf{\Phi}) = \sum_{s=1}^{S} \omega_s \|\nabla_{a_s} \sigma(\mathbf{\Phi}\mathbf{U})\|_{0,\nu} \tag{6}$$

for learning the filters. We note that the latter assumption frees us from performing segmentation during learning and thus allows us to proceed sequentially instead of in an alternating way. Further, the relaxation of the jump penalty imposes less penalty for small variations in the data which is due to the absence of regularization and favorable compared with the jump term here.

The non-linear transformation σ of the filter responses in Eq. (6) is realized via $\sigma(x) := \log(1 + \mu x^2)$ with parameter $\mu > 0$. Note, that σ is smooth and symmetric, and that it allows to attenuate outliers in the filter responses.

C. Learning From Patch Samples

We choose the training samples as a subset of image locations (and not all patches given by the image). This can be motivated as follows: first, when learning convolutional filters by minimizing Eq. (6) we evaluate the inner products of each filter kernel $\mathbf{\Phi}_k$ with the pixel neighborhood at all image locations (i, j) and sum them up w.r.t. i, j. Due to overlap, calculating the whole sum results in redundant computations. Secondly, since the data of interest consists of texture segments, we expect repeating patterns which in turn makes the full patch set look even more redundant. Based on this intuition, a randomly sampled subset of patches should suffice to learn the features from a texture image. Hence, we only consider a fixed number M of randomly sampled patches as training set.

Formally, we modify the data in the objective function Eq. (6) from the jump set over features of the entire image to the empirical mean of a set of randomly sampled super-patches $\mathbf{U}_i$ and we obtain

$$f(\mathbf{\Phi}) = \frac{1}{M} \sum_i^M \sum_{s=1}^{S} \omega_{a_s} \|\nabla_{a_s} \sigma(\mathbf{\Phi}\mathbf{U}_i)\|_{0,\nu}. \tag{7}$$

Here, the super-patches' support templates are extensions of the $\sqrt{n} \times \sqrt{n}$ support template of the filters which additionally take the considered finite difference stencil into account.

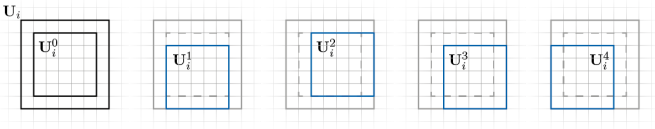


Fig. 2. Illustration of an extracted super patch $\mathbf{U}_i$ and its neighboring patches $\mathbf{U}_i^{a_s}$ with respect to the utilized finite difference system $\mathcal{N}$.

We refer to Figure 2 for a detailed visualization. For first order finite differences, a corresponding one pixel neighborhood of the considered $\sqrt{n} \times \sqrt{n}$ template is sufficient. We then generate different crops from these super-patches according to the direction of the finite difference discretization and evaluate the inner products w.r.t. these crops (and apply σ). Finally, we apply the respective finite difference operator to the obtained result.

D. Constraints

In order to avoid trivial solutions such as the zero kernel and redundancies, we impose several constraints on the filters.

1) Norm and Coherence Constraints: Following [30], we impose norm and coherence constraints. To prevent the filter coefficients from shrinking to zero, we require the Euclidean norm of each filter to equal one, i.e.,

$$\|\mathbf{\Phi}_k\|_2 = \sqrt{\sum_{i=1}^{n} (\mathbf{\Phi}_k)_i^2} = 1, \quad k = 1, \ldots, K. \tag{8}$$

Here, n is the number of coefficients in a single filter. For brevity, we consider 2D filters of quadratic support with size $\sqrt{n} \times \sqrt{n}$. The extension to filters supported on a rectangle is obvious. Geometrically, the norm constraint implies that each filter is an element of the $(n-1)$-dimensional sphere $\mathcal{S}_{n-1}$ in $\mathbb{R}^n$, and that the filter set constitutes a product of K such spheres. This structure is commonly referred to as oblique manifold, i.e. matrices in $\mathbb{R}^{n \times K}$ with normalized columns, denoted by

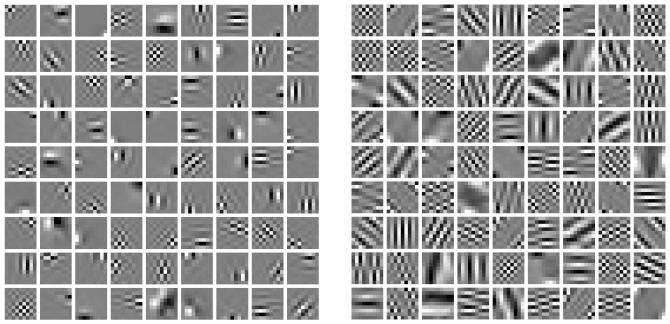
$$\mathbf{\Phi}^\top \in \mathcal{S}_{n-1}^{\times K}.$$

In addition, we use the coherence penalty, cf. [30],

$$r(\mathbf{\Phi}) = - \sum_{1 \le i \le j \le K} \log(1 - \langle \mathbf{\Phi}_i, \mathbf{\Phi}_j \rangle^2) \tag{9}$$

to well separate these vectors on the sphere. In particular, this soft constraint avoids pairwise collinear filters. We note that a minimum of that function is clearly achieved if the filters are orthogonal to each other, i.e., if the filter set lies in the corresponding Stiefel manifold. However, in the context of sparse coding, imposing such orthogonality directly as a hard constraint has turned out to be too restrictive, see for instance [30], [31].

2) Zero-Mean Constraint: The mean over the patch is a distinguished feature with special discriminative power. We consider it a seeded filter in the filter bank and learn the other filters in its orthogonal complement. This means that



(a) (b)

Fig. 3. Effect of the proposed central moment constraint. Two sets of filters learned from the same gray-scale cartoon image. (Dark and light pixels represent negative and positive filter coefficients respectively, while neutral grey indicates coefficients equal or close to zero.) The filters in *(a)* were learned with the coherence constraint from Eq. (9) but without the centroid constraint in Eq. (13). In contrast, the filters in *(b)* were learned using both the coherence constraint Eq. (9) and the central moment constraint Eq. (13). It is clearly visible that the effective support sizes of many filters in *(a)* are in fact much smaller than 9×9, and that some shifted versions of the same filter can be identified among all filters. These undesirable effects are significantly reduced in *(b)*.

we learn filters with vanishing first order moments, i.e., filters whose coefficients sum up to zero,

$$\sum_{i=1}^{n} \mathbf{\Phi}_{k,i} = 0. \tag{10}$$

We note that these filters do not see the patch mean which might vary, for instance, due to small differences in lighting or contrast. Geometrically, the filters that satisfy Eq. (10) are contained in the hyperplane which contains the origin and which is orthogonal to $\mathbb{1}_n = (1, \ldots, 1)$. Hence, the set of feasible solutions is a Riemannian manifold as well [31]. We denote it by

$$\mathcal{R} = \left(\mathcal{S}_{n-1} \cap \mathbb{1}_n^{\perp}\right)^{\times K} \tag{11}$$

in the following. The Riemannian structure is important for the optimization procedure used later on.

3) Central Moment Constraint: It might happen that there are two minimizers of Eq. (6), which adhere to norm and coherence constraints, and which are shifted versions of each other; see Figure 3a. Also note that, there, the effective support size of many filters is much smaller than the prescribed maximum 9×9 filter size.

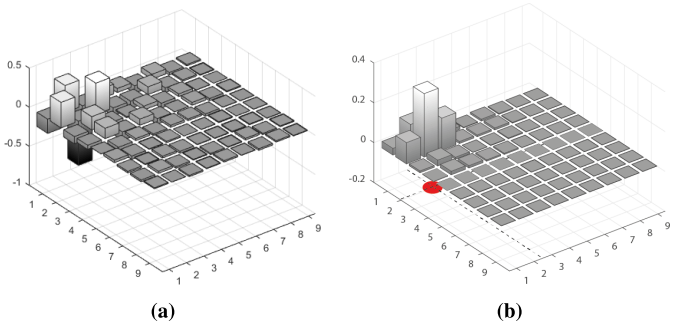
To avoid learning shifted versions of the same filter, we propose a constraint on the centroid of the squared filter coefficients. Intuitively speaking, by penalizing off-centered centroids of the pointwise squared (real-valued) filters, we prevent learning filters that are shifted versions of their centered twin. To be more precise, we consider a filter $\mathbf{\Phi}_k$ and notice that, by the employed normalization, we have $\hat{\mathbf{\Phi}}_k^{\top} \hat{\mathbf{\Phi}}_k = \sum_i \sum_j (\mathbf{\Phi}_k)_{ij}^2 = 1$ where $\hat{\mathbf{\Phi}}_k$ denotes the vectorized 2D filter $\mathbf{\Phi}_k$. Thus, the pointwise square $\mathbf{\Psi}_k$ defined by $(\mathbf{\Psi}_k)_{ij} = (\mathbf{\Phi}_k)_{ij}^2$ can be viewed as a discrete 2D probability distribution. Hence, we may compute the components of the center of mass of this distribution by

$$\bar{c}_{k,x} = \hat{\mathbf{\Phi}}_k^{\top} \mathbf{P}_x \hat{\mathbf{\Phi}}_k, \quad \bar{c}_{k,y} = \hat{\mathbf{\Phi}}_k^{\top} \mathbf{P}_y \hat{\mathbf{\Phi}}_k. \tag{12}$$

Here $\mathbf{P}_x$ is a diagonal matrix realizing the first moment with respect to the x-direction $\sum_{ij} i(\mathbf{\Psi}_k)_{ij}$, and $\mathbf{P}_y$ is given analogously. We further employ the normalization $c_{k,x} = (\bar{c}_{k,x} - \frac{\sqrt{n}+1}{2})/\frac{\sqrt{n}-1}{2})$ and the analogous normalization for $c_{k,y}$ to obtain quantities $c_{k,x}, c_{k,y}$ centered at 0 with range between -1 and 1. For a filter centered around the origin, we require $c_{k,x}, c_{k,y}$ to be close to zero. To this end, we here use the (convex) penalty

$$h(\mathbf{\Phi}_k) = \sum_{k=1}^{K} -\log[(1 - c_{k,x}^2)(1 - c_{k,y}^2)] + \frac{1}{2}(c_{k,x} - c_{k,y})^2. \tag{13}$$

The effects of the central moment conditions are illustrated in Figure 3 and in Figure 4.



(a) (b)

Fig. 4. The coefficients $(\mathbf{\Phi}_1)_{ij}$ of the first filter $\mathbf{\Phi}_1$ from the learned set $\mathbf{\Phi}$ depicted in Figure 3a without the centroid constraint (a) and its mass distribution (b). The red circle denotes the centroid $(c_{1,x}, c_{1,y})$.

E. Simplified Learning Problem and Numerical Optimization

Summing up the considerations of this section, we propose the relaxed objective Eq. (7) with the soft coherence constraint Eq. (9) and the soft shift constraint Eq. (13) which reads

$$E(\mathbf{\Phi}) = f(\mathbf{\Phi}) + \lambda r(\mathbf{\Phi}) + \kappa h(\mathbf{\Phi}). \tag{14}$$

Here λ, κ are positive parameters. The soft coherence constraint r makes the filters disentangled. The soft shift constraints h pulls the center of mass of the filters towards the origin. The learning objective Eq. (14) is a smooth non-convex function. The hard constraints (norm constraints, vanishing first moments) are encoded in the manifold $\mathcal{R}$ defined by Eq. (11). Equipped with this notation, the learning task reads

$$\mathbf{\Phi}^{\star} \in \arg\min_{\mathbf{\Phi}^{\top} \in \mathcal{R}} E(\mathbf{\Phi}). \tag{15}$$

In order to solve Eq. (15) numerically, we apply an efficient scheme that exploits the geometric structure of the manifold $\mathcal{R}$ [30], [31]. For a general introduction to gradient methods on matrix manifolds, we refer to [36]. The approach of [30] and [31] consists of a geometric variant of the conjugate gradients method with backtracking line-search and an Armijo step-size rule. For a detailed explanation, we refer

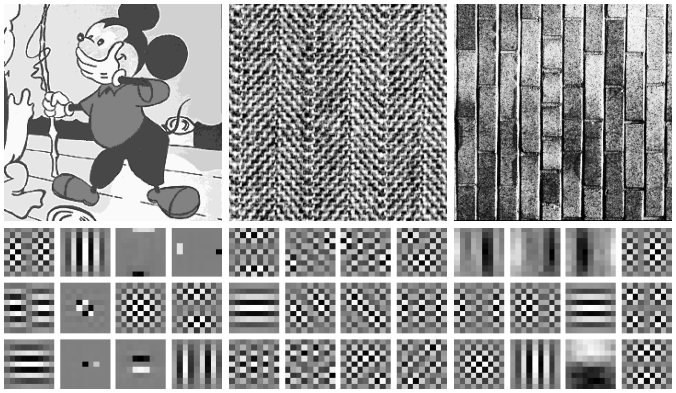


Fig. 5. Filter sets (bottom) learned from different input images (top). (a) cartoon image. (b) Brodatz 4. (c) Brodatz 12.

the reader to the aforementioned references. In our setup, the steps of one iteration are as follows:

1) compute the Euclidean gradient of the learning function Eq. (14) at the current estimate (the derivation of the Euclidean gradient can be found in the appendix);
2) project the Euclidean gradient onto the tangent space of the manifold at the estimate to obtain the Riemannian gradient;
3) compute the new descent direction by linear combination of the Riemannian gradient and the descent direction of the previous iteration via parallel transport;
4) perform a backtracking line search along the geodesic in the descent direction emanating from the current estimate to obtain an optimal step size using the Armijo rule. That means, we iteratively reduce the step size until the objective function decreases;
5) update the estimate.

We start the procedure with a random initialization in $\mathcal{R}$ and iterate until the Frobenius norm of the Riemannian gradient falls below the threshold of 10^{-5}. For illustration purposes, Figure 5 depicts the learned filter sets for different images.

F. Extension to Vector-Valued Images

So far, we only considered gray-scale texture images whereas textured images often have multiple channels, for instance RGB color images. We extend our method for the case when the image $\mathbf{U}$ is vector-valued with L channels, that is, if $\mathbf{U}_{ij} \in \mathbb{R}^L$. Let $\mathbf{U}_{i,l}^{a_s}$ the i-th patch cropped according to direction a_s in channel $l \in 1, \ldots, L$. Intuitively, different channels of an image should require different filter sets such that spatial homogeneity of filter responses can be achieved. To that end, we first extend the formulation of the patch-based filter operation

$$\mathbf{\Phi}_k \mathbf{U}_i^{a_s} = \begin{bmatrix} \mathbf{\Phi}_{k,1} & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \mathbf{\Phi}_{k,l} \end{bmatrix} \begin{bmatrix} \mathbf{U}_{i,1}^{a_s} \\ \vdots \\ \mathbf{U}_{i,l}^{a_s} \end{bmatrix}. \tag{16}$$

In this work, we consider RGB images as examples of multi-channel images. Since the red, the green and the blue channels are in general highly correlated, we assume that the patch structure within each channel will be similar and set $\mathbf{\Phi}_{k,R} = \mathbf{\Phi}_{k,G} = \mathbf{\Phi}_{k,B} = \mathbf{\Phi}_k$. Thus, the learned filters act on the different channels in the same way. We note that this does not hinder jumps in a single channel to be detected.

IV. Segmentation Stage

After relaxing the model Eq. (1) in Section III to determine suitable filters $\mathbf{\Phi}$, we here discuss the segmentation given a set of filters. To segment the vector-valued feature image $\sigma(\mathcal{F})$ we consider the (formal) Lagrangian version of the discretization of Eq. (1) for fixed $\mathbf{\Phi}$ to obtain the problem

$$\underset{\mathcal{V}}{\operatorname{argmin}}\, \gamma \sum_{s=1}^{S} \omega_s \|\nabla_{a_s} \mathcal{V}\|_0 + d(\mathcal{V}, \sigma(\mathcal{F})). \tag{17}$$

Here, $\gamma > 0$ is a parameter for tuning the trade-off between data fitting and regularity, and $\mathcal{F}$ are the filter responses of $\mathbf{U}$, i.e., $\mathcal{F} = \mathbf{\Phi}\mathbf{U}$.

A. Filter Weighting Based on the Mahalanobis Distance

We recall that all filters were constrained to have unit norm in the learning stage. As a result, all filter outputs are weighted equally regardless of their discriminative power. To account for this, we utilize a data fidelity term based on the Mahalanobis distance d. Here we use the covariance matrix of all feature vectors (after applying the non-linearity). With slight abuse of notation, let

$$\Sigma = \operatorname{cov}(\mathcal{G})$$

the $K \times K$ covariance matrix of all feature vectors in $\mathcal{G} = \sigma(\mathcal{F})$. To define the corresponding Mahalanobis distance, we write $\Sigma\mathcal{G}$ for the action of a $K \times K$ matrix Σ on the third index of the feature image $\mathcal{G}$, i.e., $(\Sigma\mathcal{G})_{ijk} = (\Sigma\mathcal{G}_{ij})_k$. Then, the Mahalanobis data fidelity reads

$$\begin{aligned} d(\mathcal{V}, \mathcal{G}) &= \sum_{ij} |(\Sigma^{-1/2}(\mathcal{V}_{ij} - \mathcal{G}_{ij})|_2^2 \\ &= \|\Sigma^{-1/2}(\mathcal{V} - \mathcal{G})\|_2^2. \end{aligned} \tag{18}$$

We observed that the results slightly improve when we normalize $\Sigma^{-1/2}$ by $\max_{ij} (\Sigma^{-1/2})_{ij}$.

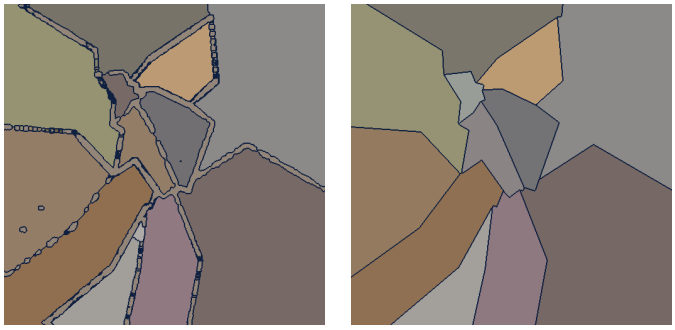
B. Variational Partitioning of the Feature Images

By the previous considerations in Section IV-A, we have to solve the minimization problem Eq. (17) with the Mahalanobis data term. To that end, we plug $\mathcal{V} = \Sigma^{1/2}\mathcal{U}$ into Eq. (17) to obtain the problem

$$\underset{\mathcal{U}}{\operatorname{argmin}}\, \gamma \sum_{s=1}^{S} \omega_s \|\nabla_{a_s} \Sigma^{1/2}\mathcal{U}\|_0 + \|\mathcal{U} - \Sigma^{-1/2}\mathcal{G}\|_2^2. \tag{19}$$

We observe that the ℓ_0 prior is invariant to invertible matrices acting in the third dimension, i.e., $\|\nabla_{a_s} \Sigma^{1/2}\mathcal{U}\|_0 = \|\nabla_{a_s}\mathcal{U}\|_0$. Therefore, the problem Eq. (19) is equivalent to the problem

$$\mathcal{U}^* = \underset{\mathcal{U}}{\operatorname{argmin}}\, \gamma \sum_{s=1}^{S} \omega_s \|\nabla_{a_s}\mathcal{U}\|_0 + \|\mathcal{U} - \Sigma^{-1/2}\mathcal{G}\|_2^2. \tag{20}$$



(a) (b)

Fig. 6. In a postprocessing step, small spurious segments are merged into their neighboring segments. (a) Raw segmentation; (b) final segmentation after region merging.

We observe that this constitutes a classical (vector-valued) piecewise constant Mumford-Shah problem for data $\Sigma^{-1/2}\mathcal{G}$ with an ℓ_2 norm data term. This is a challenging optimization problem in its own, but there are well-working approximate strategies available. Here, we utilize the ADMM-based method developed in [32], [33]. Although computationally more demanding than other recent approaches [37]-[39], this method currently gives the best quality in practice; see the comparison in [39].

C. Obtaining the Label Map

The result obtained from treating problem Eq. (18) is a vector-valued piecewise constant function. To obtain the final label map (scalar field), we simply utilize the sum of the vector in a pixel as (real-valued) index for a segment, i.e., we sum up the coefficients along the feature vector at every pixel location. We observe that segment boundaries often lead to high filter responses which results in small spurious segments at the boundaries. To remove these, we adopt the simple post-processing step from [25], where small regions are merged with neighbors based on their boundary ratios. We note that the merging is not a hierarchical approach. Figure 6 depicts the final segmentation before and after the boundary refinement step.

V. Experimental Results

We implemented the proposed learning and segmentation method in Matlab. For the segmentation step described in Section IV-B, we used of the toolbox Pottslab.[1] In addition, we utilize the region merging implementation from [25] as post-processing. The experiments were conducted on a desktop computer with an Intel i7-3930K processor with 3.2 GHz.

We compare the segmentation results produced by our method with existing algorithms on two different datasets. For a quantitative comparison, we use the well-known Prague texture segmentation dataset which comprises mosaics of color and grayscale textures. In addition, we show that the same method is also effective in segmenting the histology images from [3].

[1] Available at http://pottslab.de.

A. Prague Texture Dataset

The Prague texture segmentation dataset [40] consists of 80 texture mosaics which are synthetically generated from random compositions of 114 different textures from 10 thematic categories. Color (RGB) and grayscale versions of this dataset are available along with the respective ground truth segment map and each texture mosaic is of size 512×512 pixels and the number of segments varies between 3 and 12. For a quantitative comparison, we produce segmentations of the large color texture dataset – used in the ICPR 2014 contest – and evaluate them against their ground truth using region-based metrics correct segmentation (CS), over-segmentation (OS), under-segmentation (US), missed error (ME), noise error (NE); pixel-based metrics 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); and consistency-based metrics global consistency error (GCE) and local consistency error (LCE). If available, we also report the Mirkin metric (dM), Van Dongen metric (dD) as well as the variation of information (dVI). For computing these metrics, we use the benchmark provided by the authors of the Prague dataset at [34] and where a detailed definition of above metrics can be found.

For each of the 80 texture mosaics in the benchmark, we learn a separate set of filters $\mathbf{\Phi}$ and compute the segmentation based on the these filter outputs subsequently. The parameters for learning the features and performing the Potts segmentation are set empirically and remain fixed for all instances in the dataset. The learned filter sets contain $K = 41$ filters of size 9×9 each and are learned from $M = 50\,000$ patches that are drawn from the mosaic (uniform random sampling). By setting the parameter M to a large value, we effectively consider all image patches for learning. However, we observed that results did not improve beyond $M = 50K$. In principle, the objective function Eq. (1) does not require the filters $\mathbf{\Phi}_1, \ldots, \mathbf{\Phi}_K$ to be of equal size. For simplicity, we used filters of identical size, and note that filters of smaller size are included in the utilized filter set by zero-padding. As is common practice in patch-based methods (for example [41]), we weight all pixels in the patch by a Gaussian mask to give more weight to the central pixel which leads to slightly better localized segment boundaries. In the learning problem (15) we set the parameter of the non-linearities to $\mu = \nu = 2000$ and the weights of the coherence and moment-centering penalties to $\lambda = 10$ and $\kappa = 10$. In the Potts segmentation that follows we require the weight that trades data fidelity against spatial homogeneity of the solution and therefore effectively influences the degree of over-segmentation. Empirically, we find $\gamma = 0.03$ to provide a good trade-off between over- and under-segmentation over all benchmark images. The texture mosaic needed in average 35 min for the learning stage and 9 min for the segmentation stage.

To assess the performance of our approach, we compare our results to several state-of-the-art algorithms that were used for segmentation on the Prague texture

TABLE I

RESULTS ON THE PRAGUE COLOR TEXTURE DATASET (ICPR2014 CONTEST). EACH ROW CORRESPONDS TO A SEGMENTATION QUALITY METRIC, AND THE ARROW INDICATES IF HIGH OR LOW VALUES ARE BETTER. THE FIRST RANK IS MARKED BY BOLDFACE, THE SECOND RANK IS MARKED BY AN ASTERISK

Method	TS	SWA	GMRF	AR3D	TFR	TFR+	RS	FSEG	PMCFA	PCA-MS	Proposed
↑ CS	59.13	27.06	31.93	37.24	46.13	51.25	46.02	69.02	75.32*	72.27	**77.73**
↓ OS	10.89	50.21	53.27	59.53	**2.37**	5.84*	13.96	17.30	11.95	18.33	15.92
↓ US	18.79	**4.53**	11.24	8.86	23.99	7.16	30.01	11.85	9.65	9.41	6.31*
↓ ME	10.45	25.76	14.97	12.54	26.70	31.64	12.01	6.28	4.57	4.19*	**3.93**
↓ NE	9.93	27.50	16.91	13.14	25.23	31.38	11.77	5.66	4.63	**3.92**	**3.92**
↓ O		33.01	36.49	35.19	27.00	23.60	35.11	10.79	**4.51**	7.25*	7.68
↓ C		85.19	12.18	11.85	26.47	22.42	29.91	13.75	8.87*	**6.44**	24.24
↑ CA		54.84	57.91	59.46	61.32	67.45	58.75	77.50	**83.50**	81.13	82.80*
↑ CO		60.67	63.51	64.81	73.00	76.40	68.89	84.11	**88.16**	85.96	86.89*
↑ CC		88.17	89.26	91.79*	68.91	81.12	69.30	86.89	90.73	91.24	**93.65**
↓ I.		39.33	36.49	35.19	27.00	23.60	31.11	15.89	**11.84**	14.04	13.11*
↓ II.		2.11	3.14	3.39	8.56	4.09	8.63	2.60	**1.47**	1.59	1.50*
↑ EA		66.94	68.41	69.60	68.62	75.80	65.87	83.99	**88.10**	87.08	88.03*
↑ MS		53.71	57.42	58.89	59.76	65.19	55.52	78.25	**83.98**	81.84	**83.98**
↓ RM		6.11	4.56	4.66	7.57	6.87	10.96	4.51	3.76*	4.45	**3.27**
↑ CI		70.32	71.80	73.15	69.73	77.21	67.35	84.71	88.74*	87.81	**89.03**
↓ GCE		17.27	16.03	12.13	15.52	20.35	11.23	10.82	**6.51**	8.33	7.40*
↓ LCE		11.49	7.31	6.69	12.03	14.36	7.70	7.51	**3.92**	5.61*	5.62
↓ dD							18.52		10.13	9.06*	**8.57**
↓ dM							23.67		6.41	5.88*	**5.30**
↓ dVI							**13.31**		15.80	14.54*	14.88

mosaics such as the Texel-based Segmentation (TS) [12], Segmentation by Weighted Aggregation (SWA) [20], Gaussian MRF Model With EM (GMRF) [21], 3D Auto Regressive Model With EM (AR3D) [22], Texture Fragmentation and Reconstruction (TFR) and (TFR+) [23], Regression-based Segmentation (RS) [24], Factorization-Based Texture Segmentation (FSEG) [25], Priority Multi-Class Flooding Algorithm (PMCFA) [26], [27] and Variational Multi-Phase Segmentation (PCA-MS) [4]. Table I provides the segmentation accuracy benchmark results as reported on the benchmark website [34] and in [25] as well as in [4]. In addition, Figure 7 depicts some of the segmentations produced by the four top-performing methods including our results for visual comparison.

B. Parameter Sensitivity

We explore the sensitivity of our method with respect to the most influential parameters. To that end, we conduct an evaluation of our method with varying parameters on a representative subset of images from the Prague benchmark dataset drawn from the different categories all, bark, flowers, glass, nature, stone and textile. We examine the filter size, the number of learned filters K, the weight of the filter coherence penalty λ, and the parameters μ and ν of the employed non-linearities. We vary each of them while keeping the others fixed at the values described in Subsection V-A.

We start with the parameter λ which controls the maximum coherence between all filter pairs and which is given in Eq. (15). Table II shows that if λ is close to zero which effectively disables this constraint, segmentation results deteriorate significantly. For λ larger than 1, we observe only negligible changes in segmentation results across all quality metrics. These results underline the importance of the constraint in our learning objective but also reveal that the choice of its exact value is not critical as long as it is large enough.

Next, we investigate the influence of the number of filters K. We conclude from Table III that the segmentation quality increases for up to 41 filters and deteriorates for larger numbers. The initial improvement might be explained by the increased discriminatory power obtained from a larger number of different filters. The deterioration of the quality for a larger number of filters might be explained by an over-segmentation caused by irrelevant features.

We continue by studying the influence of the filter size. The choice of the filter size should relate to the scale of the texture. Although the Prague texture mosaics expose a relatively large variety of texture scales, we find that filter sizes of 7 and 9 pixels achieve the best results in average (see Table IV), which confirms the choice of other works, e.g. [4] and [25]. For small filter sizes, within-texture variations are similar to variations at texture boundaries which leads to undersegmentation in the segmentation stage. For our method, we also observe that large filters lead to a decreased localization of texture boundaries, and to larger spurious segments at texture boundaries as depicted in Figure 6. Now we consider the non-linearity parameter ν which we used for relaxation of the ℓ_0 jump penalty for filter learning in Eq. (5). We recall that for large ν the surrogate function approximates the original sparsifying function well [41], [42]. Table V lists segmentation results over a large range of ν. The segmentation fails for small values of ν and improves when increasing it. Due to the decreasing slope of the surrogate function for large values of ν, the learning algorithm converges more slowly. Our choice in the experiments reflects a trade-off between convergence speed and approximation accuracy.

Fig. 7. Exemplary segmentation results on the Prague texture segmentation dataset. *From top to bottom:* input image, ground truth, FSEG [25], PMCFA [26], [27], PCA-MS [4] and the proposed method.

Finally, we investigate the sensitivity of the corresponding parameter μ in the non-linearity σ of Eq. (6) in Table VI. We find that the overall segmentation results are robust over a large range of choices of μ and segmentation quality only starts suffering for very large values of μ where the shape of σ degenerates.

C. Histology Dataset

We apply our method to the histology dataset used in [3]. The dataset contains 36 color images of size 128×128 pixels of stained tissue along with segmentations by an expert. Instead of the adaptive color quantization used in [3], we simply convert the image to gray-scale prior to processing. Since the images are considerably smaller than the Prague texture mosaics, we reduce the filter size to 5×5, learn only 13 filters and therefore adjust the trade-off parameter in the segmentation stage to a fixed $\gamma = 0.8$ but otherwise use the same setup as in the Prague texture experiment. We point out that switching to this quite different class of images only required the adjustment of these few parameters. The learning stage requires approximately 2 minutes and the

TABLE II
SENSITIVITY W.R.T. THE FILTER COHERENCE PENALTY λ

λ	0.1	1	5	10	15	20	50	100
↑ CS	57.38	72.58	70.97	71.59	72.54	65.85	68.56	75.70
↓ OS	9.67	18.64	18.69	18.73	18.63	18.71	18.67	14.81
↓ US	34.65	7.98	3.66	7.78	3.67	14.58	11.72	7.98
↓ ME	2.97	5.19	10.04	9.26	10.04	5.19	5.19	5.19
↓ NE	0.42	5.97	10.42	10.13	10.41	6.14	6.23	5.96
↓ O	26.14	9.44	12.12	12.97	10.04	11.57	11.36	9.34
↓ C	26.22	31.05	31.07	31.19	31.03	31.84	33.88	31.12
↑ CA	68.49	79.52	78.90	79.59	79.75	75.48	77.23	81.27
↑ CO	78.31	83.99	83.13	84.34	84.14	81.47	82.55	85.80
↑ CC	74.22	90.96	91.19	90.45	90.77	86.06	88.20	90.96
↓ I.	21.69	16.01	16.87	15.66	15.86	18.53	17.45	14.20
↓ II.	5.00	1.49	1.38	1.54	1.52	1.99	1.88	1.52
↑ EA	74.10	85.14	84.67	85.41	85.37	81.54	83.22	86.58
↑ MS	69.46	79.69	79.14	79.79	79.78	75.78	77.27	81.47
↓ RM	7.32	3.85	3.83	3.79	3.73	4.71	4.09	3.59
↑ CI	75.11	86.22	85.81	86.33	86.33	82.58	84.23	87.42
↓ GCE	6.34	8.87	8.69	9.86	8.80	9.58	9.78	8.77
↓ LCE	5.31	5.90	5.99	6.83	6.12	5.95	6.05	5.78
↓ dD	12.76	9.99	10.40	10.21	9.93	11.19	10.71	9.06
↓ dM	11.19	5.39	5.48	5.47	5.31	6.10	5.87	4.96
↓ dVI	14.02	15.37	15.50	15.27	15.33	15.15	15.28	15.17

TABLE III
SENSITIVITY W.R.T. THE NUMBER OF FILTERS K

K	11	21	31	41	61	81	122	162
↑ CS	32.37	52.23	70.25	72.59	59.13	43.36	17.43	0.47
↓ OS	6.93	13.94	15.13	18.73	14.74	16.10	21.06	0.00
↓ US	66.81	28.78	13.65	7.78	8.49	3.65	30.43	46.97
↓ ME	0.15	14.43	11.91	5.19	16.14	35.38	29.49	51.15
↓ NE	0.00	14.29	12.53	6.18	15.99	35.68	27.38	47.06
↓ O	51.79	42.58	14.76	10.17	17.22	27.10	52.82	76.49
↓ C	36.68	31.94	22.52	31.26	31.78	46.46	76.02	64.75
↑ CA	44.39	61.58	75.33	80.02	71.65	63.04	38.22	21.89
↑ CO	59.30	71.38	82.67	84.27	77.58	70.48	49.76	38.53
↑ CC	46.42	68.23	81.84	91.25	88.68	84.58	68.67	42.50
↓ I.	40.70	28.62	17.33	15.73	22.42	29.52	50.24	61.47
↓ II.	14.82	6.25	2.83	1.38	2.16	3.69	9.53	15.08
↑ EA	49.77	67.15	80.61	85.61	78.86	71.84	49.06	31.98
↑ MS	39.39	58.60	75.53	80.28	72.36	62.84	32.61	11.44
↓ RM	17.04	10.77	5.81	3.82	5.27	6.13	14.13	18.71
↑ CI	51.20	68.39	81.38	86.62	80.79	74.35	53.51	35.65
↓ GCE	3.21	7.71	7.75	8.65	12.27	17.57	19.34	25.64
↓ LCE	2.27	5.35	6.46	5.88	7.88	12.06	14.01	17.93
↓ dD	21.09	16.48	11.23	9.80	14.39	19.70	30.97	37.73
↓ dM	38.36	18.27	7.81	5.28	7.77	11.91	31.97	43.74
↓ dVI	11.81	13.52	14.41	15.34	15.85	16.05	15.42	13.46

segmentation stage around 3 seconds. Some of the results are given in Figure 8.

D. Discussion

From Table I we observe that the proposed method significantly improves upon most existing approaches in the Prague texture segmentation benchmark. Moreover, the segmentations obtained by the proposed method are competitive with the previously best performing method PMCFA. PMCFA and the proposed method yield a comparable number of first and second ranks. The segmentation examples in Figure 7 indicate that our method gives very satisfactory results for segments with clear repeated texture patterns, as for instance in the first three examples. Erroneous segmentations appear mostly when quite different patterns such as the red blossoms on green background in the fifth image are present in a segment. A possible cause for this is that the blossoms are interpreted as a texture on its own on a smaller scale. Qualitatively, we observe that our method tends to a slight oversegmentation when large color contrasts are present. This is not the case for PMCFA. Compared to PMCFA, the proposed method produces smoother boundaries.

In addition to the Prague texture segmentation benchmark, the algorithm produces useful segmentations of the tissue images of [3]. It is mostly very close to the expert annotations. We stress that we only had to adapt the maximum number of filters K, their maximum size and the segmentation

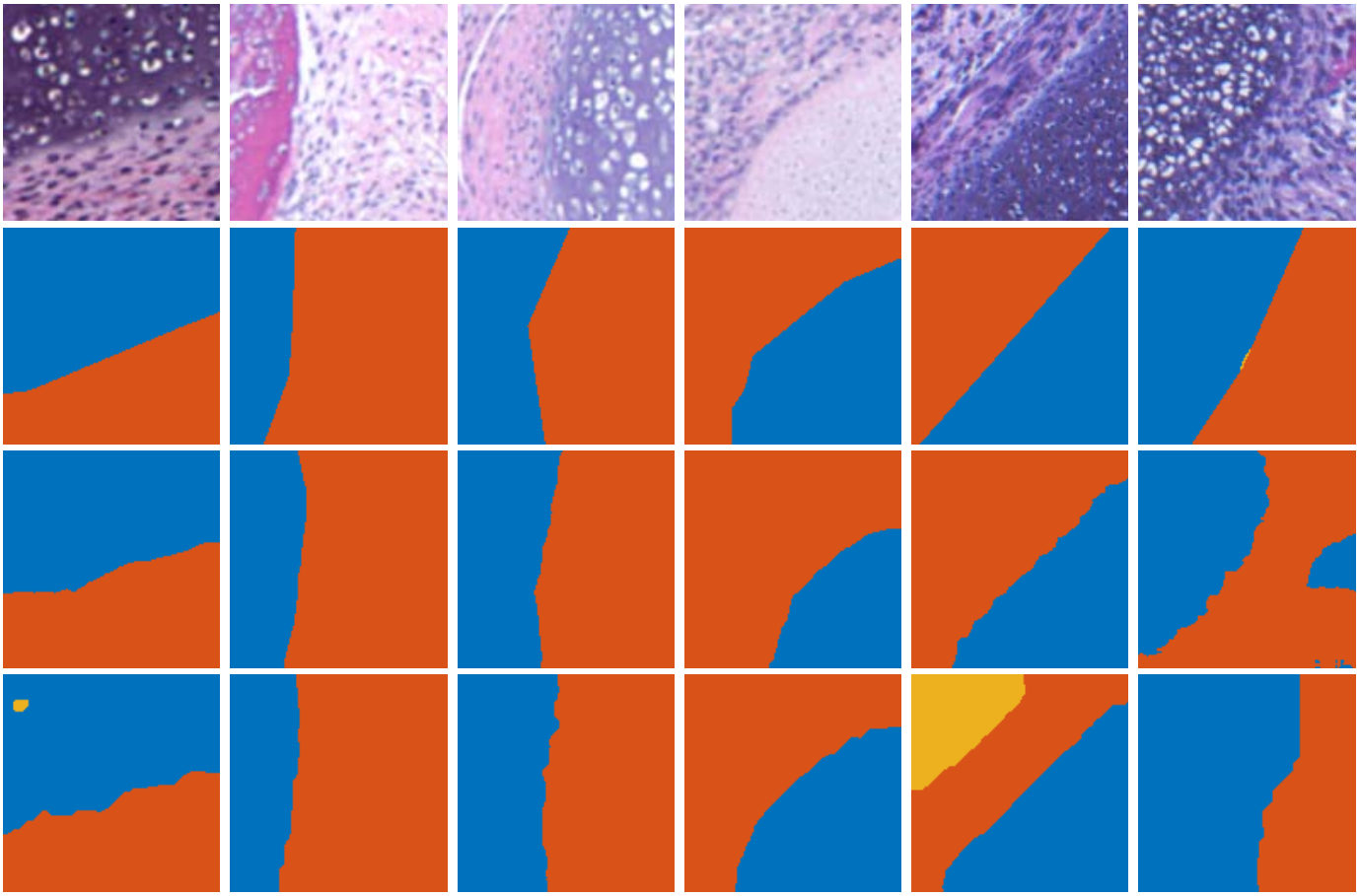


Fig. 8. Segmentation results on the histology dataset from [3]. *From top to bottom:* input image, ground truth, ORTSEG [3], and the proposed method.

TABLE IV
SENSITIVITY W.R.T. THE FILTER SIZE

filter size	5	7	9	11	13
↑ CS	55.23	69.74	71.59	59.68	54.99
↓ OS	9.22	14.05	18.73	13.06	8.56
↓ US	39.69	14.28	7.78	16.11	9.31
↓ ME	6.63	8.39	9.26	17.14	25.10
↓ NE	5.93	7.22	10.13	17.95	25.22
↓ O	38.39	6.18	12.97	12.81	21.91
↓ C	21.64	16.97	31.19	33.25	32.46
↑ CA	62.27	78.59	79.59	71.45	66.58
↑ CO	73.21	84.33	84.34	78.28	74.54
↑ CC	64.40	87.46	90.45	86.10	81.01
↓ I.	26.79	15.67	15.66	21.72	25.46
↓ II.	8.39	1.66	1.54	2.30	3.55
↑ EA	66.87	83.50	85.41	78.82	74.44
↑ MS	60.30	78.59	79.79	72.27	66.01
↓ RM	10.50	5.42	3.79	5.64	6.31
↑ CI	67.78	84.57	86.33	80.37	76.00
↓ GCE	4.07	7.60	9.86	12.27	15.64
↓ LCE	3.59	4.80	6.83	8.01	11.30
↓ dD	14.56	9.97	10.21	14.00	17.09
↓ dM	21.38	5.96	5.47	7.75	10.35
↓ dVI	12.90	14.79	15.27	15.58	15.51

TABLE V
SENSITIVITY W.R.T. THE NON-LINEARITY PARAMETER ν

ν	100	500	1000	2000	3000
↑ CS	0.00	29.57	49.31	71.56	69.02
↓ OS	0.00	0.00	6.83	18.71	23.61
↓ US	99.93	69.31	47.39	7.69	7.99
↓ ME	0.00	0.00	0.00	10.04	3.10
↓ NE	0.00	0.00	0.00	10.42	3.86
↓ O	100.00	58.36	44.93	10.50	13.68
↓ C	71.63	40.48	46.06	31.09	32.54
↑ CA	8.43	38.42	53.34	78.44	77.09
↑ CO	28.05	54.02	64.42	83.80	81.35
↑ CC	8.44	39.29	57.00	87.97	91.25
↓ I.	71.95	45.98	35.58	16.20	18.65
↓ II.	24.62	16.39	13.10	2.65	1.40
↑ EA	12.80	43.06	57.62	84.16	83.57
↑ MS	−7.92	31.03	47.78	78.78	77.82
↓ RM	30.45	19.14	15.15	4.40	4.36
↑ CI	15.25	44.71	59.04	84.97	84.86
↓ GCE	0.14	2.16	3.13	8.58	9.00
↓ LCE	0.14	1.78	2.55	6.04	6.20
↓ dD	36.01	23.55	18.61	10.04	11.23
↓ dM	75.36	43.91	35.86	7.01	5.93
↓ dVI	9.14	11.20	12.43	15.02	15.74

hyperparameter γ to obtain the presented results. This indicates the potential of the proposed method for segmenting different classes of images.

The main trade-off of our method is a relatively long processing time per image. In contrast to most other methods where a fixed set of features is used for segmentation, we here

TABLE VI
SENSITIVITY W.R.T. NON-LINEARITY PARAMETER μ

μ	10	100	2000	5000	10000
↑ CS	74.69	71.79	72.59	65.99	24.30
↓ OS	18.44	22.18	18.73	15.33	6.80
↓ US	7.97	7.89	7.78	6.89	36.01
↓ ME	0.00	0.00	5.19	16.82	29.83
↓ NE	0.66	0.75	6.18	16.76	29.64
↓ O	13.33	13.96	10.17	15.18	59.31
↓ C	31.08	31.16	31.26	30.92	51.44
↑ CA	79.50	79.04	80.02	72.85	41.24
↑ CO	83.65	82.92	84.27	79.35	53.97
↑ CC	91.40	91.46	91.25	84.99	60.72
↓ I.	16.35	17.08	15.73	20.65	46.03
↓ II.	1.43	1.39	1.38	2.73	10.29
↑ EA	84.96	84.83	85.61	78.93	49.49
↑ MS	80.21	79.55	80.28	72.51	36.61
↓ RM	2.95	3.79	3.82	5.22	14.07
↑ CI	86.12	85.93	86.62	80.42	52.84
↓ GCE	8.22	8.12	8.65	13.34	15.58
↓ LCE	5.60	5.83	5.88	8.56	12.02
↓ dD	9.85	10.19	9.80	13.50	27.86
↓ dM	5.31	5.41	5.28	8.21	29.43
↓ dVI	15.59	15.57	15.34	15.26	14.53

run the learning stage prior to segmentation, which increases the overall running time. The computational complexity of the learning stage is primarily determined by size and number of the filters as well as the number of patches and their channels that are used for learning. A speed-up could be achieved by reducing the number of training samples. We observed that reducing the number of samples for training from $50K$ to $10K$ only slightly decreased the segmentation quality. Also, filter learning is so far started with a random initialization. In practical applications, the filter set can be initialized with prelearned filters which could bring down the required number of iterations during learning and therefore significantly decrease the overall running time. A further speed-up might be obtained by an optimized implementation.

VI. CONCLUSION

We have developed a method for unsupervised texture segmentation where the features are learned from images without ground truth segmentation. Our first main contribution was the development of a corresponding model based on local homogeneity assumptions. We learn convolutional features in a way that they produce approximately piecewise constant feature images and combine this with the piecewise constant Mumford-Shah model. Our second main contribution was the development of a practical algorithm for unsupervised texture segmentation based on that model. To make the problem computationally tractable, we relaxed it and we decomposed it into a filter learning stage and a segmentation stage. In the filter learning stage, we employed a geometric conjugate gradient descent method, whereas in the segmentation stage, we used the Lagrange formulation of the piecewise constant Mumford-Shah model which we proposed to augment with a Mahalanobis distance as data term. The proposed algorithm yields competitive results on the standard benchmark dataset for unsupervised texture segmentation. Furthermore, switching to the quite different class of histological images only required the adjustment of a few parameters. The improved segmentation quality underpins the idea of learning features adapted to the image under consideration. The proposed approach may be especially valuable in situations where creating large training sets of accurate ground truth segmentations or hand-crafting features is expensive.

Topics of future research include speeding-up the proposed method as explained in the discussion section as well as approaching the proposed non-smooth model more directly, that is, employing fewer relaxation steps.

APPENDIX

We derive the Euclidean gradient required in the numerical optimization of the filter learning problem described in Eq. (14).

The cost function to minimize in the learning stage consists of three terms, one each for the approximated cost of the jump set $f(\mathbf{\Phi})$, the centroid penalty $r(\mathbf{\Phi})$ and the coherence penalty $h(\mathbf{\Phi})$.

A. Sparsity Objective

First, we provide the derivative of the approximated cost of the jump set. Explicitly, let $\mathbf{\Phi}_k \in \mathbb{R}^n$ a vectorized 2D filter of size $\sqrt{n}\times\sqrt{n}$ that is applied on a set of M vectorized 2D image patches $\mathbf{U} \in \mathbb{R}^{n\times M}$ by taking their standard inner product $\mathbf{\Phi}_k^\top \mathbf{U} \in \mathbb{R}^{1\times M}$ and for a set of filters we get $\mathbf{\Phi U} \in \mathbb{R}^{K\times M}$ accordingly. By denoting $\mathbf{D}_{a_s} = \nabla_{a_s}\sigma(\mathbf{\Phi U})$ shorthand for the difference of features along $\mathbf{a}_s$ and using $\odot$ for the Hadamard product, we obtain

$$\begin{aligned}\frac{\partial}{\partial \mathbf{\Phi}} f(\mathbf{\Phi}) &= 4\nu\mu \sum_{s=1} \omega_{a_s}\left[\left(\frac{\mathbf{D}_{a_s}}{1+\nu\|\mathbf{D}_{a_s}\|^2}\odot\frac{\mathbf{\Phi U}_{a_s}}{1+\mu(\mathbf{\Phi U}_{a_s})^2}\right)\mathbf{U}_{a_s}^\top\right.\\ &\quad\left.-\left(\frac{\mathbf{D}_{a_s}}{1+\nu\|\mathbf{D}_{a_s}\|^2}\odot\frac{\mathbf{\Phi U}_0}{1+\mu(\mathbf{\Phi U}_0)^2}\right)\mathbf{U}_0^\top\right] \qquad (21)\end{aligned}$$

for the derivative of f. Here, $\mathbf{U}_{a_s}$ is the $n\times M$ data matrix containing vectorized patches $\mathbf{U}_i^{a_s}$ cropped from the M sampled super-patches in direction $\mathbf{a}_s$ according to Figure 2.

B. Centroid Penalty

Second, we require the derivative of the centroid constraint Eq. (13). This constraint acts on each filter independently. For the individual filter, we get

$$\frac{\partial}{\partial \mathbf{\Phi}_k} h(\mathbf{\Phi}) = \frac{4}{w_-}\left[\frac{c_{k,x}\mathbf{P}_x}{1-c_{k,x}^2}+\frac{c_{k,y}\mathbf{P}_y}{1-c_{k,y}^2}+\frac{1}{2}(c_{k,x}-c_{k,y})(\mathbf{P}_x-\mathbf{P}_y)\right]\mathbf{\Phi}_k$$

by using $w_- = \frac{\sqrt{n}-1}{2}$ as a shorthand notation for the half width of the filter. By stacking the individual derivatives we can then write the derivative of h with respect to the entire filter set as

$$\frac{\partial}{\partial \mathbf{\Phi}} h(\mathbf{\Phi}) = \left[\frac{\partial}{\partial \mathbf{\Phi}_1} h(\mathbf{\Phi}), \ldots, \frac{\partial}{\partial \mathbf{\Phi}_K} h(\mathbf{\Phi})\right]^\top. \qquad (22)$$

C. Coherence Penalty

Last, the gradient of the coherence penalty is given in [30, eq. 9] and reads as

$$\frac{\partial}{\partial \mathbf{\Phi}} r(\mathbf{\Phi}) = \left[\sum_{1 \leq i < j \leq k} \frac{2 \mathbf{\Phi}_i^\top \mathbf{\Phi}_j}{1 - \left(\mathbf{\Phi}_i^\top \mathbf{\Phi}_j \right)^2} \left(\mathcal{E}_{ij} + \mathcal{E}_{ji} \right) \right] \mathbf{\Phi}. \quad (23)$$

Here $\mathcal{E}_{ij}$ is a matrix with a one in component ij and zero elsewhere.

Finally, we obtain the gradient of the cost function Eq. (14) by combining Eq. (21), Eq. (22) and Eq. (23)

$$\nabla E(\mathbf{\Phi}) = \frac{\partial}{\partial \mathbf{\Phi}} f(\mathbf{\Phi}) + \lambda \frac{\partial}{\partial \mathbf{\Phi}} r(\mathbf{\Phi}) + \kappa \frac{\partial}{\partial \mathbf{\Phi}} h(\mathbf{\Phi}).$$

REFERENCES

[1] D. R. Martin, C. C. Fowlkes, and J. Malik, "Learning to detect natural image boundaries using local brightness, color, and texture cues," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 5, pp. 530–549, May 2004.

[2] H. Mobahi, S. R. Rao, A. Y. Yang, S. S. Sastry, and Y. Ma, "Segmentation of natural images by texture and boundary compression," *Int. J. Comput. Vis.*, vol. 95, no. 1, pp. 86–98, 2011.

[3] M. T. McCann, D. G. Mixon, M. C. Fickus, C. A. Castro, J. A. Ozolek, and J. Kovačević, "Images as occlusions of textures: A framework for segmentation," *IEEE Trans. Image Process.*, vol. 23, no. 5, pp. 2033–2046, May 2014.

[4] N. Mevenkamp and B. Berkels, "Variational multi-phase segmentation using high-dimensional local features," in *Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV)*, Mar. 2016, pp. 1–9.

[5] T. Randen and J. H. Husøy, "Filtering for texture classification: A comparative study," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 21, no. 4, pp. 291–310, Apr. 1999.

[6] A. K. Jain and F. Farrokhnia, "Unsupervised texture segmentation using Gabor filters," *Pattern Recognit.*, vol. 24, no. 12, pp. 1167–1186, 1991.

[7] M. Unser, "Texture classification and segmentation using wavelet frames," *IEEE Trans. Image Process.*, vol. 4, no. 11, pp. 1549–1560, Nov. 1995.

[8] R. Azencott, J.-P. Wang, and L. Younes, "Texture classification using windowed Fourier filters," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 19, no. 2, pp. 148–153, Feb. 1997.

[9] M. Unser and M. Eden, "Nonlinear operators for improving texture segmentation based on features extracted by spatial filtering," *IEEE Trans. Syst., Man, Cybern.*, vol. 20, no. 4, pp. 804–815, Jul. 1990.

[10] X. Liu and D. Wang, "Image and texture segmentation using local spectral histograms," *IEEE Trans. Image Process.*, vol. 15, no. 10, pp. 3066–3077, Oct. 2006.

[11] Y. Xia, D. D. Feng, and R. Zhao, "Morphology-based multifractal estimation for texture segmentation," *IEEE Trans. Image Process.*, vol. 15, no. 3, pp. 614–623, Mar. 2006.

[12] S. Todorovic and N. Ahuja, "Texel-based texture segmentation," in *Proc. IEEE Int. Conf. Comput. Vis.*, Sep./Oct. 2009, pp. 841–848.

[13] T. Ojala, M. Pietikäinen, and T. Mäenpää, "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 7, pp. 971–987, Jul. 2002.

[14] M. Özden and E. Polat, "Image segmentation using color and texture features," in *Proc. 13th Eur. Signal Process. Conf.*, Sep. 2005, pp. 1–4.

[15] S. Geman and D. Geman, "Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-6, no. 6, pp. 721–741, Nov. 1984.

[16] D. Mumford and J. Shah, "Optimal approximations by piecewise smooth functions and associated variational problems," *Commun. Pure Appl. Math.*, vol. 42, no. 5, pp. 577–685, 1989.

[17] M. Rousson, T. Brox, and R. Deriche, "Active unsupervised texture segmentation on a diffusion based feature space," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2. Madison, WI, USA, Jun. 2003, pp. II-699–II-704.

[18] Z. Kato and T.-C. Pong, "A Markov random field image segmentation model for color textured images," *Image Vis. Comput.*, vol. 24, no. 10, pp. 1103–1114, 2006.

[19] M. Storath, A. Weinmann, and M. Unser, "Unsupervised texture segmentation using monogenic curvelets and the Potts model," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Oct. 2014, pp. 4348–4352.

[20] M. Galun, E. Sharon, R. Basri, and A. Brandt, "Texture segmentation by multiscale aggregation of filter responses and shape elements," in *Proc. 9th IEEE Int. Conf. Comput. Vis.*, Oct. 2003, pp. 716–723.

[21] M. Haindl and S. Mikeš, "Model-Based Texture Segmentation," *Image Analysis and Recognition*. Berlin, Germany: Springer, 2004, pp. 306–313.

[22] M. Haindl and S. Mikes, "Unsupervised texture segmentation using multispectral modelling approach," in *Proc. IEEE Int. Conf. Pattern Recognit. (ICPR)*, Aug. 2006, pp. 203–206.

[23] G. Scarpa, R. Gaetano, M. Haindl, and J. Zerubia, "Hierarchical multiple Markov chain model for unsupervised texture segmentation," *IEEE Trans. Image Process.*, vol. 18, no. 8, pp. 1830–1843, Aug. 2009.

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

[25] J. Yuan, D. Wang, and A. M. Cheriyadat, "Factorization-based texture segmentation," *IEEE Trans. Image Process.*, vol. 24, no. 11, pp. 3488–3497, Nov. 2015.

[26] C. Panagiotakis, I. Grinias, and G. Tziritas, "Natural image segmentation based on tree equipartition, Bayesian flooding and region merging," *IEEE Trans. Image Process.*, vol. 20, no. 8, pp. 2276–2287, Aug. 2011.

[27] C. Panagiotakis, I. Grinias, and G. Tziritas. (2014). *Texture Segmentation Based on Voting of Blocks, Bayesian Flooding and Region Merging*. Accessed: Jun. 14, 2016. [Online]. Available: https://sites.google.com/site/costaspanagiotakis/research/imagesegmentation

[28] X. Ren and J. Malik, "Learning a classification model for segmentation," in *Proc. IEEE 9th Int. Conf. Comput. Vis.*, Oct. 2003, pp. 10–17.

[29] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2015, pp. 3431–3440.

[30] S. Hawe, M. Kleinsteuber, and K. Diepold, "Analysis operator learning and its application to image reconstruction," *IEEE Trans. Image Process.*, vol. 22, no. 6, pp. 2138–2150, Jun. 2013.

[31] M. Kiechle, T. Habigt, S. Hawe, and M. Kleinsteuber, "A bimodal co-sparse analysis model for image processing," *Int. J. Comput. Vis.*, vol. 114, nos. 2–3, pp. 233–247, 2015.

[32] M. Storath, A. Weinmann, and L. Demaret, "Jump-sparse and sparse recovery using potts functionals," *IEEE Trans. Signal Process.*, vol. 62, no. 14, pp. 3654–3666, Jul. 2014.

[33] M. Storath and A. Weinmann, "Fast partitioning of vector-valued images," *SIAM J. Imag. Sci.*, vol. 7, no. 3, pp. 1826–1852, 2014.

[34] M. Haindl and S. Mikes. (2016). *The Prague Texture Segmentation Datagenerator and Benchmark*. Accessed: Jun. 13, 2016. [Online]. Available: http://mosaic.utia.cas.cz

[35] A. Chambolle, "Finite-differences discretizations of the Mumford-Shah functional," *ESAIM, Math. Model. Numer. Anal.*, vol. 33, no. 02, pp. 261–288, 1999.

[36] P.-A. Absil, R. Mahony, and R. Sepulchre, *Optimization Algorithms on Matrix Manifolds*. Princeton, NJ, USA: Princeton Univ. Press, 2009.

[37] L. Xu, C. Lu, Y. Xu, and J. Jia, "Image smoothing via L_0 gradient minimization," *ACM Trans. Graph.*, vol. 30, no. 6, 2011, Art. no. 174.

[38] X. Cheng, M. Zeng, and X. Liu, "Feature-preserving filtering with L_0 gradient minimization," *Comput. Graph.*, vol. 38, pp. 150–157, Feb. 2014.

[39] R. M. H. Nguyen and M. S. Brown, "Fast and effective L_0 gradient minimization by region fusion," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 208–216.

[40] M. Haindl and S. Mikes, "Texture segmentation benchmark," in *Proc. IEEE Int. Conf. Pattern Recognit. (ICPR)*, Dec. 2008, pp. 1–4.

[41] C. Nieuwenhuis, S. Hawe, M. Kleinsteuber, and D. Cremers, "Co-sparse textural similarity for interactive segmentation," in *Proc. ECCV*, 2014, pp. 285–301.

[42] M. Kiechle, S. Hawe, and M. Kleinsteuber, "A joint intensity and depth co-sparse analysis model for depth map super-resolution," in *Proc. Int. Conf. Comput. Vis. (ICCV)*, 2013, pp. 1545–1552.

Martin Kiechle received the Diploma degree in electrical and computer engineering and the Honours degree in technology management both from Technische Universit; München, in 2012. Since 2013, he has been a Researcher with the Research Group for Geometric Optimization and Machine Learning, Technische Universität, München. His research interests include signal and image processing, representation learning, geometric optimization, and inverse problems.

Martin Storath received the Diploma degree in mathematics in 2008, the Honours degree in technology management in 2009, and the Ph.D. degree in mathematics in 2013, all from Technische Universität München. From 2010 to 2013, he was a Researcher with the Institute of Biomathematics, Helmholtz Zentrum München. From 2013 to 2016, he was a Post-Doctoral Researcher with the Biomedical Imaging Group, École Polytechnique Fédérale de Lausanne, Switzerland. He is currently a Post-Doctoral Researcher with Image Analysis and Learning Group, Universität Heidelberg. His research interests include signal and image processing, biomedical imaging, variational methods, and inverse problems.

Andreas Weinmann received the Diploma degree (Hons.) in mathematics and computer science from Technische Universität, München, in 2006, and the Ph.D. degree (Hons.) from Technische Universität, Graz, in 2010. He is currently with the Helmholtz Center Munich and the Department of Mathematics and Natural Sciences, Hochschule Darmstadt. His research interests are applied analysis and signal and image processing.

Martin Kleinsteuber received the Ph.D. degree in mathematics from the University of Würzburg, Germany, in 2006. After post-doctoral positions at National ICT Australia Ltd., the Australian National University, Canberra, Australia, and the University of Würzburg, he has been appointed as an Assistant Professor for geometric optimization and machine learning with the Department of Electrical and Computer Engineering, Technical University of Munich, Germany, in 2009. He was a recipient of the SIAM Student Paper Prize in 2004 and the Robert-Sauer-Award of the Bavarian Academy of Science in 2008 for his research on Jacobi-type methods on Lie algebras. Since 2016, he has been a Lead with the Data Science Group, Mercateo AG, Munich.