Content-based Propagation of User Markings for Interactive Segmentation of Patterned Images

Vedrana Andersen Dahl[a,1]**, Camilla Himmelstrup Trinderup**[1]**, Monica Jane Emerson**[1]**,
Anders Bjorholm Dahl**[1]

[1]Technical University of Denmark, Department of Applied Mathematics and Computer Science, Richard Petersens Plads, Kgs. Lyngby, Denmark

arXiv:1809.02226v1 [cs.CV] 6 Sep 2018

Abstract Efficient and easy segmentation of images and volumes is of great practical importance. Segmentation problems which motivate our approach originate from imaging commonly used in materials science and medicine. We formulate image segmentation as a probabilistic pixel classification problem, and we apply segmentation as a step towards characterising image content. Our method allows the user to define structures of interest by interactively marking a subset of pixels. Thanks to the real-time feedback, the user can place new markings strategically, depending on the current outcome. The final pixel classification may be obtained from a very modest user input. An important ingredient of our method is a graph that encodes image content. This graph is built in an unsupervised manner during initialisation, and is based on clustering of image features. Since we combine a limited amount of user-labelled data with the clustering information obtained from the unlabelled parts of the image, our method fits in the general framework of semi-supervised learning. We demonstrate how this can be a very efficient approach to segmentation through pixel classification.

1 Introduction

In this paper we propose an interactive method for probabilistic classification of pixels, which can be used for segmentation of 2D and 3D images. Our approach is especially advantageous for detecting patterns, a situation regularly occurring in images of materials and medical samples. Such images often show a collection of objects which are to be separated from the background. One example we extensively use in this paper is concerned with detection of fibres in volumetric data of composite materials, see Fig 1. Another example is detection and segmentation of cells in histological images.

[a]e-mail: vand@dtu.dk

When segmenting images showing a collection of similar objects, an established strategy involves extensive modelling of the objects' appearance, usually leading to a highly specialised method. Another common strategy is to learn the appearance of the objects from a large amount of prelabelled data, often with high computational requirements during the training phase. Here we aim for a general method that requires limited computation, as well as modest user-labelling.

Our method fits into the framework of semi-supervised learning, combining two ingredients: a model for image content created in an unsupervised manner from the image features, and a modest input from the user. When a user marks a structure in the image as belonging to a class, our method propagates the marks to similar structures in the rest of the image. The output is a layered image which at every pixel position contains the probabilities of belonging to each of the defined classes. We call this output *pixelwise probabilities* of belonging to segmentation classes. From pixelwise probabilities, the segmentation is readily obtained by selecting the most probable class for each pixel. The method is highly flexible and captures the features which are of interest to the user; an example with various image features is shown in Fig. 2. Our approach allows easy segmentation of complex structures, that would otherwise require the development of algorithms targeted at specific problems.

An important property of our model is real-time feedback, allowing the user to place new markings strategically, depending on the current result. For this to work without delay, the segmentation must be updated very fast. Our method relies on an efficient update of the parameters used for pixel classification, and equally efficient update of the classification results. With results shown promptly, the user can continue adding marks until the desired outcome is learned by the algorithm. Having learned the desired outcome, the classification model can be applied to other images of the same type in an unsupervised manner, that is, without additional

user input. Such pipeline has many applications, for example, in microscopy or when segmenting slices from a volumetric image.

Our prototype implementation, including a graphical user interface, is in Matlab, and we made the code available through MathWorks File Exchange (search for InSegt) - *to come*.

1.1 Related work

Benefits of user input with real-time feedback have been recognised in image segmentation. A comprehensive summary of interactive approaches can be found in Boykov [1]. Here we review some important advances to place our method in the existing framework, and to explain how our method differs from the current trends in interactive segmentation.

Early interactive techniques for segmentation of highly complex images include intelligent scissors [2] or live wire [3], where the user cuts out an object by placing markers along its boundary. Based on edge information, the algorithm traces the boundary by finding the shortest path in an edge-weighted graph. These algorithms are computationally cheap, but require a lot of user effort to obtain a segmentation. Less user input is required when using interactive graph cuts [4][5], which often give very impressive results with only a few seeds provided by the user. The algorithm separates the foreground from the background based on the boundary and region properties of segments. In the GrabCut method [6] the user provides a bounding rectangle, often leading to very precise foreground-background separation. Optional editing using brush strokes can be carried out to correct finer details. Extensions of GrabCut include shape priors [7] and and improvement to graph cut energy representation [8]. An alternative to combinatorial graph-based solutions is the use of continuous representation of segmentation boundaries. Such interactive active contours often minimise an energy functional in a variational framework [9][10].

Common to the described methods is the focus on segmenting relatively large foreground objects, which justifies using regularisation on the length or the curvature of the segmentation boundary. In some applications it is, however, not possible to use a strong regulariser. For example, when segmenting the circular fibres shown in Fig. 1, regularisation could remove or merge small regions. The need for segmenting a number of small objects is often seen in areas like microscopy for life science or materials science. Appearance of such images can vary significantly, with texture as well as intensity carrying information that is useful for obtaining the desired segmentation. A specialist would use such clues to distinguish amongst structures, but automating the segmentation task typically requires highly sophisticated and problem-adapted methods. While there are situations which justify the development of a specialised method, in many cases a reasonable result with modest interactive effort would be preferred.

When segmenting small image structures, e.g. cells, a well-suited approach is classification of pixels. This is the basis for the ilastik segmentation tool [11], which employs a random forest classifier [12] trained on image features including colour, edges, orientation and texture. The features are computed from the image before starting the interactive labelling of image structures, while parameters of the random forest classifier are learned from the manual labelling. When a user updates the labels to improve the segmentation, the parameters of the classifier need to be re-learned, which is computationally costly and causes a noticeable delay in the feedback. Another specialised tool for segmentation of microscopy images is the trainable Weka segmentation [13] (a part of the Fiji [14] distribution of ImageJ) which utilises a data mining and machine learning toolkit for solving pixel classification problems. A user can choose from a variety of image features and interactively re-train the classifier.

Frameworks using neural networks are increasingly popular in pixel classification, and often yield impressive results [15]. A neural network operates on features extracted locally from the image. This input is fed through a series of multidimensional linear functions, with a non-linear activation between them, ending up in a probabilistic output. The weights of the linear functions need to be trained by optimising the performance on the usually large set of prelabelled data. This provides an extreme flexibility to the method and, provided an adequate training, neural networks may solve pixel classification problems as accurately as specialists. However, neural networks are dependent on large training sets and require computationally costly training, which makes them less convenient for the task of segmenting a small set of images.

Our approach shares some similarities with neural networks. We also feed the input through linear functions with non-linear steps in between. However, we use the features extracted from the image to construct the linear functions in a preprocessing step. The functions are then kept fixed, while they operate on the interactively provided user input, resulting in a probabilistic output. Due to the fixed linear functions, our method is not as adaptable as neural networks. For example, our approach is less fit for semantic segmentation of photographs. Nevertheless, we achieve excellent results when segmenting patterned images, without requiring a large set of labelled data and without performing a costly optimisation during interactive update.

The foundation of our method is a linear operator encoding image content. The linear operator is described in terms of a dictionary, as it assigns image pixels to dictionary pixels. The relation between the image and the dictionary can be formulated as a bipartite graph and represented using a biadjacency matrix. The approach has been used for evolv-
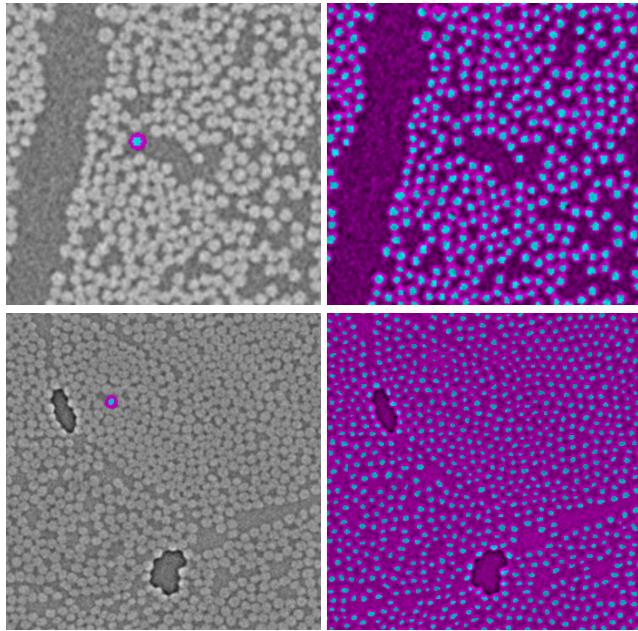
Fig. 1 Detecting glass and carbon fibers using our interactive pattern-based method. On the *left* input images and a very small subset of pixels manually labelled as either being close to a fibre centre (cyan), or not being close to a fibre centre (magenta). On the *right*, the manual labelling has been propagated to the whole image and the result is obtained by selecting the most probable class for each pixel.

ing deformable models in [16][17][18]. An early version of the method, without the interactive update, proved valuable for quantifying composite materials [19]. In this work we use the image–dictionary relationship to propagate the brush strokes provided by the users.

2 Method

Our method combines two sources of information, the structure in the image and the user-provided partial labelling. The structure in the image is captured in the preprocessing step, namely clustering, which we describe in 2.1. After that, in 2.2, we explain how clustering is used for transforming the user-provided partial labelling into pixelwise probabilities of belonging to each of the classes. The central part of our segmentation, the interactive update, covered in 2.3, is obtained by immediately displaying the result of the transformation and allowing the user to repeatedly improve the partial labelling.

Postprocessing choices, covered in 2.4, are concerned with the outputs of the interactive update. The first and most obvious output is a probability image. The probability image can give the image segmentation, but other postprocessing methods may be utilised as well. For the second output, which we call *dictionary probabilities*, the user-provided partial labellings are propagated to the dictionary patches which were obtained in the preprocessing step. This encodes the

learned information used for transforming the intensity image to the probability image, and can be used for subsequent automatic processing of similar images.

Our method comes in a range of flavours, governed by the features used for clustering and the strategy used for calculating pixelwise probabilities. In this section we only explain the simplest variant, the other possibilities are covered in Sec. 3.

Notation. Throughout the paper we consider an image I defined on an X-by-Y image grid with pixel values in either grayscale or RGB colour space. During the interactive part, the user will be placing marks in the image grid, to indicate the pixels which belong to one of the C segmentation classes. We chose to represent this user-provided information with a layered *label image* L, where $L(x,y,c) = 1$ if the user indicated that pixel (x,y) belongs to class c, and 0 otherwise.

2.1 Clustering image patches

The aim of preprocessing is to find the structures in the image without considering the user-provided labels. In the framework of semi-supervised learning, a cluster assumption states that, if points are in the same cluster, they are likely to be of the same class – which does not imply that each class forms a single cluster [20]. For our purpose, we assume that image features tend to form discrete clusters and that image features in the same cluster are more likely to share a class. However, we do not assume that each class is represented by only one cluster, so we will need many more clusters than classes. Therefore, we create a multitude of clusters to capture the variety of features present in the image.

In Sec. 3 we will explain the implementation details and some more advanced ways of accomplishing clustering. In this section we outline the basic approach, which operates on intensity patches. For this case, only two parameters are required: the number of clusters K and the size of the patches M. The number of clusters should be large, measured in hundreds or thousands, and is roughly reflecting the variability in the image. The size of the patches should reflect the scale of the distinctive image features and could, for example, be 9 pixels. For simplicity, we always assume that the size of the image patches M is odd and patches are centred around the central pixel.

For clustering, we extract patches of size M-by-M from the image I, treat each patch as a vector containing the pixel intensities and group those vectors into K clusters, e.g. using k-means clustering based on Euclidean distance. The resulting collection of cluster centres represents the content of the image. As these basic elements are inferred by grouping features from image, we call the collection of K cluster centres
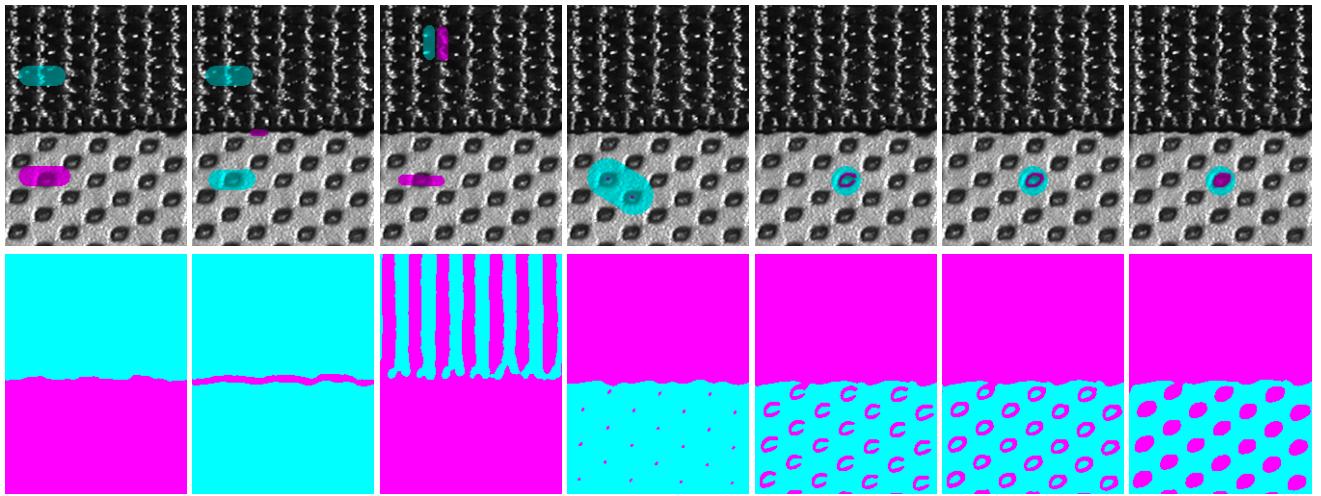
Fig. 2 An example demonstrating the flexibility of our method. In the *top* row, two manually labelled classes (cyan and magenta), corresponding to different image features. In the *bottom* row, a resulting image segmentation obtained by binarizing a probability image.

an *intensity dictionary*, and each of its elements (each cluster centre) is denoted *dictionary patch*. Every image pixel (x, y) in the centre of an M-by-M image patch is, by means of clustering, uniquely assigned to one cluster. We represent this using an *assignment image A*. For boundary pixels we define $A(x, y) = 0$.

2.2 Relation between image and dictionary

According to the cluster assumption, image patches assigned to the same dictionary patch are more likely to belong to the same class. Unique for our method is that we use this assumption on a pixel level, and not on a patch level. That is, if two image patches are assigned to the same dictionary patch, their *corresponding* pixels (i.e. the pixels at the same position in the patch) are more likely to belong to the same class. In other words, for every dictionary patch there is a certain (unknown) classification of its pixels, which all assigned patches are likely to share.

To exploit this assumption, we define a binary relation between corresponding pixels assigned to the same dictionary pixel. For example, a central pixel of an image patch assigned to a certain dictionary patch relates to central pixels of all other patches assigned to the same dictionary patch. Likewise, the pixel directly above the central pixel relates to corresponding pixels in other patches, and the similar relation extends to all positions in a patch. This results in $M^2 K$ cliques of pixels, one for every pixel in the intensity dictionary. Due to the overlap between image patches, every non-boundary pixel belongs to M^2 different cliques.

The central part of our method is concerned with transforming a user-provided partial labelling to pixelwise probabilities. The transformation matrix we use has a very simple decomposition, which makes our method efficient and allows for immediate feedback to the user. The construction of the transformation matrix is therefore fundamental for our method. However, describing how this matrix is constructed provides little intuition about our method, so we start by motivating our approach.

As covered previously, the assignment image A, obtained in an unsupervised manner, contains information on clusters of structures in the image I. At the same time, image I is accompanied by the user-provided partial labelling L. To combine the two sources of information, we create a dictionary of labels to accompany our intensity dictionary. For each dictionary patch $k \in \{1, \dots, K\}$ we use A to identify the locations of all image patches assigned to it. At those locations in the image grid we extract all corresponding patches but from the labelling image L. For the set of related labelling patches we compute a pixelwise average for every layer. As a result, every M-by-M dictionary patch now has a corresponding M-by-M labelling representation consisting of C layers.

When the image is fully labelled, the label image L sums to one in every pixel, as only one out of C classes has a label of 1. Consequently, the labelling representation of every dictionary patch also sums to one in every pixel. However, due to the pixelwise averaging, the values of this representation are not binary, they instead encode the normalised frequency of a dictionary pixel being labelled as belonging to class c in the current labelling image. For this reason, we think of this labelling representation as of pixelwise probabilities of belonging to class c, and we call them *dictionary probabilities*.

Dictionary probabilities can now be pasted back into an X-by-Y image grid, again using the location information from A, and again averaged in every pixel. This results in an X-by-Y probability image P consisting of C layers, where P is a diffused version of L. In other words, we use the self-

similarity information encoded by A to propagate the user-provided markings from L onto the rest of the image.

In light of this motivation, now we turn to explaining the construction of the transformation matrices used for efficient computation of dictionary probabilities and image probabilities. Fundamental for this transformation is the relation between the X-by-Y image grid and the M-by-M-by-K dictionary grid. This relation will be encoded using an n-by-m biadjacency matrix $\mathbf{B}$, where $n = XY$ and $m = M^2K$. For this purpose, we need a linear (single) index for the pixels in the image and the pixels in the dictionary grid.

The linear index of an image pixel (x, y) is

$$i = x + (y - 1)X. \tag{1}$$

As for the dictionary grid, we use $(0, 0, k)$ for the central pixel of the k-th dictionary element, and coordinates of other pixels in the patch are defined in terms of within-patch displacements Δx and Δy, both from $\{-s, \ldots, 0, \ldots, s\}$ with $s = (M-1)/2$. A dictionary pixel at coordinates $(\Delta x, \Delta y, k)$ has a linear index

$$j = (\Delta x + s) + (\Delta y + s)M + (k - 1)M^2. \tag{2}$$

Each assignment of an image patch centered around (x, y) to a k-th dictionary patch centered around $(0, 0, k)$ induces a relation between the M^2 image pixels and the M^2 dictionary pixels, see Fig. 3. Using $\sim$ for denoting a relation between image pixels and dictionary pixels gives

$$A(x, y) = k \quad \Rightarrow \quad \begin{array}{l} (x + \Delta x, y + \Delta y) \sim (\Delta x, \Delta y, k), \\ \text{for all } \Delta x \text{ and } \Delta y \end{array}. \tag{3}$$

Since image patches are overlapping, every non-boundary image pixel relates to M^2 dictionary pixels. Image pixels in a boundary relate to less than M^2 dictionary pixels, and the four corner pixels relate to only one dictionary pixel. In total there are $(X - 2s)(Y - 2s)M^2$ relations between the image pixels and the dictionary pixels.

We represent the relations between n image pixels and m dictionary pixels using an n-by-m biadjacency matrix $\mathbf{B}$, with elements

$$b_{ij} = \begin{cases} 1 & i \sim j \\ 0 & \text{otherwise} \end{cases}, \tag{4}$$

where i and j are linear indices of an image pixel and a dictionary pixel. The algorithm for constructing $\mathbf{B}$ is summarised in Alg. 1.

The biadjacency matrix $\mathbf{B}$ defines the linear mapping used to propagate the information from the image to the dictionary and vice versa. Consider a quantity defined on the image grid (e.g. user-provided markings indicating pixels which belong to class 1) arranged into a length n vector $\mathbf{v}$ such that the i-th element contains the value of the i-th

---

Algorithm 1 Construction of $\mathbf{B}$

1: Initiate $\mathbf{B}$ as an n-by-m matrix with $b_{ij} = 0$
2: **for** an non-boundary pixel (x, y) **do**
3:     Retrieve pixel assignment $k = A(x, y)$
4:     **for** within-patch displacement $(\Delta x, \Delta y)$ **do**
5:         compute i for $(x + \Delta x, y + \Delta y)$ using Eq. (1)
6:         compute j for $(\Delta x, \Delta y, k)$ using Eq. (2)
7:         assign $b_{ij} = 1$
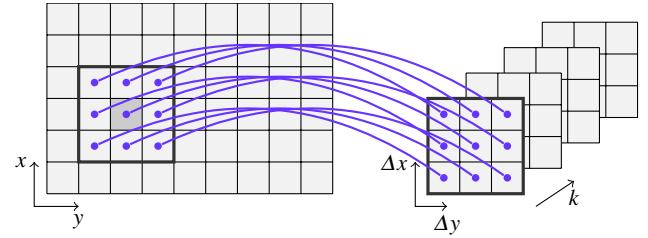8:   **end for**
9: **end for**

Fig. 3 A subset of relations between a 9×6 image and a $3 \times 3 \times 4$ dictionary caused by the framed patch centered around the pixel shaded darker being assigned to the first dictionary patch.

image pixel. Propagating these values to the dictionary is carried out by calculating a length m vector

$$\mathbf{d} = \text{diag}(\mathbf{B}^\mathsf{T}\mathbf{1}_{n \times 1})^{-1}\mathbf{B}^\mathsf{T}\mathbf{v}, \tag{5}$$

where $\mathbf{1}$ denotes a column vector of ones, while $\text{diag}(\cdot)$ denotes a diagonal matrix with the diagonal defined by the argument. The j-th element of $\mathbf{d}$ contains the value of the j-th dictionary pixel computed by averaging the values of the related image pixels. The summation is accomplished by multiplying with $\mathbf{B}^\mathsf{T}$ while the diagonal matrix accomplishes the division with the total number of related pixels.

For this reason we define the m-by-n transformation matrix for mapping from the image to the dictionary as

$$\mathbf{T}_1 = \text{diag}(\mathbf{B}^\mathsf{T}\mathbf{1}_{n \times 1})^{-1}\mathbf{B}^\mathsf{T}. \tag{6}$$

Similarly, mapping from the dictionary to the image is given by the n-by-m matrix

$$\mathbf{T}_2 = \text{diag}(\mathbf{B}\mathbf{1}_{m \times 1})^{-1}\mathbf{B}. \tag{7}$$

Those two transformation matrices are fundamental for our method. The propagation of user-provided markings (as described in the motivational paragraphs) is computed as

$$\mathbf{P} = \mathbf{T}_2\mathbf{T}_1\mathbf{L}, \tag{8}$$

where $\mathbf{L}$ is the user-provided labelling L arranged in a n-by-C matrix, while the resulting n-by-C matrix $\mathbf{P}$ needs to be arranged back into a layered image P.

2.3 Interactive update

When equipping our method with the user-provided interactive update, we run into choices with regards to: i) the way in which we treat unlabelled pixels, ii) the number of applied diffusion steps, and the way of treating intermediate results between the steps, and iii) the possibility of changing the number of segmentation classes. After testing many types of interactive updates, we kept three main versions. In all versions the number of classes C is chosen during initialisation and kept fixed during the update.

The way in which we handle pixels that have not been labelled by the user is also common to all versions. Such pixels are initially assigned equal probability of belonging to each class. As a result, before the user places the first label, all probabilities are equal and no segmentation is possible.

The user starts the interaction by choosing a pencil corresponding to one of the C classes applies markings to some pixels. The partial labelling information is immediately transformed to the probability image and shown to the user as an image segmentation, such that every pixel is placed in the class with the highest probability. After the first pencil stroke, only one class will have values larger than $\frac{1}{C}$ in the label image L, and the same applies for the probability image P computed using (8). Thus, at first, many pixels will belong to the first marked class and no pixels will be assigned to the classes that have not used yet. As user adds markings for the other classes, those will appear in probability image P.

Thanks to the real-time feedback, the user can quickly improve the result by placing markings in misclassified regions (the regions that have been incorrectly classified). With many unlabelled pixels in L, the image P will typically have many values that only differ slightly from $\frac{1}{C}$. Those small deviations carry the information needed for inferring the class of the unlabelled pixels.

As for the number of applied diffusion steps, we use either one or two. When using two diffusion steps, instead of continuing to diffuse the (already diffused) probability image, we can apply additional non-linear operations in between the two diffusions. Very good results are obtained if we apply *binarisation* of the labels between the two diffusion steps. For binarisation we identify the class of the highest probability for each pixel, and apply $\{0, 1\}$ labelling. If there are pixels with no clear probability maximum, we let them retain their unresolved labels. Consequently, for the second iteration of the diffusion, many pixels act as labelled, and this improves the quality of the result.

Another additional operation for the two-step diffusion involves the subset of pixels which has been labelled by the user. After the first diffusion step, the $\{0, 1\}$ labelling of those pixels has probably dispersed, and some might even have a maximal probability in a class different from the markings indicated by the user. The operation of *overwriting* imposes the original user-provided labelling to all labelled pixels in between the two diffusion steps.

The options for the two-step diffusion, binarisation and overwriting are implemented in our segmentation tool, such that the user can quickly switch between the variants of the method and decide which one yields the best results for the data at hand. Likewise, the user can quickly determine whether the quality of the results is sufficient or additional markings should be placed.

The user can choose to see the output of the classification displayed as a final segmentation based on the resulting probability image. Alternatively, there is an option for inspecting the C probability images, which often gives a better insight into the quality of the result. In some cases, the final classification may seem incorrect, but the probability images do contain useful information which can be postprocessed for obtaining the desired result.

2.4 Postprocessing

Our approach allows for various postprocessing options, which may be grouped into two postprocessing strategies. One strategy involves processing the probability image to obtain the segmentation, or detection of interesting features from the probability image. These operations are application-driven and examples are illustrated in Sec. 4.

The second strategy involves reusing the information stored in the dictionary and the associated dictionary probabilities. The linear transformation (8), which is core to our method, first transforms the user-provided markings from L to the dictionary space (using matrix $\mathbf{T}_1$) and then back to the image space (using matrix $\mathbf{T}_2$). Consider only the first product

$$\mathbf{D} = \mathbf{T}_1 \mathbf{L}.$$

This is an m-by-C matrix containing the pixelwise probabilities of the dictionary pixels (i.e. the dictionary probabilities) which can be useful for processing a previously unseen image similar to I.

Processing a new image $\hat{I}$ requires extracting all M-by-M patches for every pixel of $\hat{I}$ and assigning those patches to the *existing* dictionary, i.e. the dictionary created using patches from I. Just like before, this assignment defines an image-to-dictionary and we can compute the two associated transformation matrices. Here we are interested in the dictionary-to-image transformation $\hat{\mathbf{T}}_2$. To compute the probability image corresponding to the unlabelled image $\hat{I}$ we therefore need to compute

$$\hat{\mathbf{P}} = \hat{\mathbf{T}}_2 \mathbf{D}.$$

and rearrange the result into $\hat{P}$.

This way of using our method fits into the framework of supervised learning. The original image I and the computed

labelling L can in this context be seen as a (labelled) training set (ignoring the fact that the labelling is computed in a semi-supervised way). Our method is then capable of producing the probability image $\hat{P}$ for the new, unlabelled image $\hat{I}$. The approach will work as long as the initial clustering captures the features present in $\hat{I}$, which holds for similar images.

3 Implementation details

When developing a framework for interactive propagation of markings we made a number of implementational choices governed by the performance of our method. The part of the method concerning the update includes a few of options mentioned earlier (running the diffusion step twice and discretising between diffusion steps).

As for the clustering step for preprocessing the data, our experience leads to two conclusions. First, our method is rather robust to the *quality* of the clustering, so using an approximate clustering will generally not deteriorate the output. Second, the features used for clustering need to reflect the distinction in the appearance of the classes we want to separate. For many types of images, an intensity-based approach as sketched in Sec. 2 will perform reasonably well. However, in challenging cases, more elaborate image features might provide better results. In this section, we briefly touch upon different possibilities.

With the method being robust with respect to the quality of the clustering, we focus on efficiency when building the dictionary. Therefore, we choose to use a k-means tree [21], which is built from consecutive k-means clusterings. In this implementation, the size of the dictionary is defined in terms of the branching factor b and the number of layers t. Since each node in the tree makes up a dictionary element, the total number of dictionary elements is given by $K = \frac{b^{t+1}-1}{b-1}$.

Our experience is that good performance is obtained also without running the k-means until convergence for each three layer, and therefore a fixed number of iterations is chosen, e.g. 10 iterations. Furthermore, in order to limit the computational burden and memory usage, we extract only a subset of M-by-M patches from the image when building the dictionary.

As for producing A given the clustering represented by a k-means tree, the patch vector is compared with the nodes in the first layer to find the match. The patch vector is then compared to the children of this node, and the most similar node is again chosen. This process is repeated until a leaf node or an empty node is reached. The patch vector is assigned to the most similar node along this path.

Apart from clustering image patches, we also experimented with different image features. Some of the results we show in Sec. 4 are based on SIFT [22], but other features can also be incorporated in our method. The approach is as follows. First, image features represented by vectors are extracted from all pixel positions in the image and clustered in K clusters. For speed, it often suffices to consider only a subset of pixels for clustering, as long as we capture the variability in the image. Every position (x, y) from the image grid can now be uniquely assigned to one of the k clusters – the cluster that is closest to the feature vector extracted at (x, y). This results in an assignment image A. The only additional information we need for building the transformation matrices is a value M, which earlier represented the size of the extracted image patches. The value M now determines the size of the overlap when linking the image to the dictionary. While we now freely chose M, it is reasonable to use a value which corresponds to the size of the extracted features.

4 Results

In Fig. 4 we show a three-class classification of a volumetric X-ray image of peripheral nerves appearing as tubular structures. Using a purely intensity-based approach to pixel classification, it would be difficult to differentiate between the bright background and the bright regions inside the dark tubes. Furthermore, a significant bias field makes it difficult to choose a global threshold. Our approach utilises a very limited user input in just one slice of the volume to differentiate between three classes: background, tubes and inside. Moreover, the dictionary probabilities learned from processing one slice can be used for automatic classification of all other slices in the volume, yielding a volumetric segmentation.

Fig. 5 shows an example of segmenting a volumetric image of a fibre composite into two classes: background class and fibre centre class. Using our method, a huge number of individual fibres can be segmented with a modest user input. The probability image of a fibre centre class precisely indicates a region for each fibre centre, and can readily be used in postprocessing for obtaining quantitative information about the spatial distribution of fibres. In this example we also use the result of single-slice segmentation for batch processing of a whole volume stack. In principle, this yields the centre line of each individual fibre. For comparison, we also show a result obtained by thresholding the intensity image. This nicely illustrates a challenge in segmenting densely packed fibres, when the image resolution does not suffice to clearly delineate the boundary of every individual fibre. In such a case, a successful segmentation requires utilisation of the repetitive patterns in the image. Our method accomplishes this via clustering.

In Fig. 6 we show a three-class segmentation of onion cells. Since cell walls and nuclei both appear dark, a purely intensity-based method would not distinguish these two classes–

a task which our method successfully solves with only a modest user input.

In Fig. 7 we show the use of our method for counting cells in a stained microscopy image. Unlike other examples, this is a colour (RGB) image. To utilise colour information, the features extracted from every image patch contain three colour channels concatenated inyo a single feature vector. Since the final goal is to count and measure the size of cells, we postprocess the probability images to obtain individual cell segmentation. We detect the centre of each cell by computing the local maxima of the centre-class probability image and we estimate the extent of each cell by considering both the centre-class and boundary-class probability images, coupled with the distance from the previously detected cell centres.

5 Conclusion

We propose a method for interactive labeling of image pixels. Instrumental for our method is a pair of closely related transformations which propagate the information from the image grid to a dictionary, and back to the image. The transformations are constructed such that the propagation is strong between image pixels with similar appearance, captured by extracted image features.

In this paper we present an algorithm for building an efficient matrix representation of those transformations, allowing a real-time processing. We demonstrate how propagation of user-provided labelling can be used for an interactive image segmentation. Furthermore, a segmentation of one image allows for subsequent automatic processing of similar images.

With only a modest user input, our method can yield good results when segmenting patterned images. We find this extremely useful for many tasks in materials and life sciences.

Fig. 4 Volumetric segmentation of peripheral nerves. In the *top* row, a slice from the volumetric data with overlayed limited user input and the three-class segmentation dividing the pixels into background (cyan), tubes (purple) and inside (magenta). The *middle* row shows two layers of probability images, corresponding to the tube class and the inside class. High intensity indicates a hight probability of belonging to the class. The *bottom* row shows the 3D visualization of the volumetric data obtained by processing a full image stack and assigning each voxel to the class of the highest probability. This experiment was performed using $M = 9$, $K = 4000$ and a clustering based on SIFT features. Small connected components of less than 10^4 voxels were removed during postprocessing.

References

1. Y. Boykov, in *Computer Vision, A Reference Guide* (2014), pp. 416–422
2. E.N. Mortensen, W.A. Barrett, in *Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques* (ACM, 1995), pp. 191–198
3. A.X. Falcão, J.K. Udupa, S. Samarasekera, S. Sharma, B.E. Hirsch, R.d.A. Lotufo, Graphical models and image processing **60**(4), 233 (1998)
4. Y.Y. Boykov, M.P. Jolly, in *IEEE International Conference on Computer Vision*, vol. 1 (IEEE, 2001), vol. 1, pp. 105–112
5. Y. Boykov, G. Funka-Lea, International journal of computer vision **70**(2), 109 (2006)
6. C. Rother, V. Kolmogorov, A. Blake, ACM Transactions on Graphics (TOG) **23**(3), 309 (2004)
7. B.L. Price, B. Morse, S. Cohen, in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (IEEE, 2010), pp. 3161–3168
8. M. Tang, L. Gorelick, O. Veksler, Y. Boykov, in *IEEE International Conference on Computer Vision* (IEEE, 2013), pp. 1769–1776
9. M. Unger, T. Pock, W. Trobin, D. Cremers, H. Bischof, in *Proceedings of the British Machine Vision Conference*, vol. 31 (Citeseer, 2008), vol. 31, pp. 44–46
10. J. Santner, T. Pock, H. Bischof, in *Asian Conference on Computer Vision* (Springer, 2010), pp. 397–410
11. C. Sommer, C. Straehle, U. Koethe, F. Hamprecht, et al., in *IEEE International Symposium on Biomedical Imaging: From Nano to Macro* (IEEE, 2011), pp. 230–233
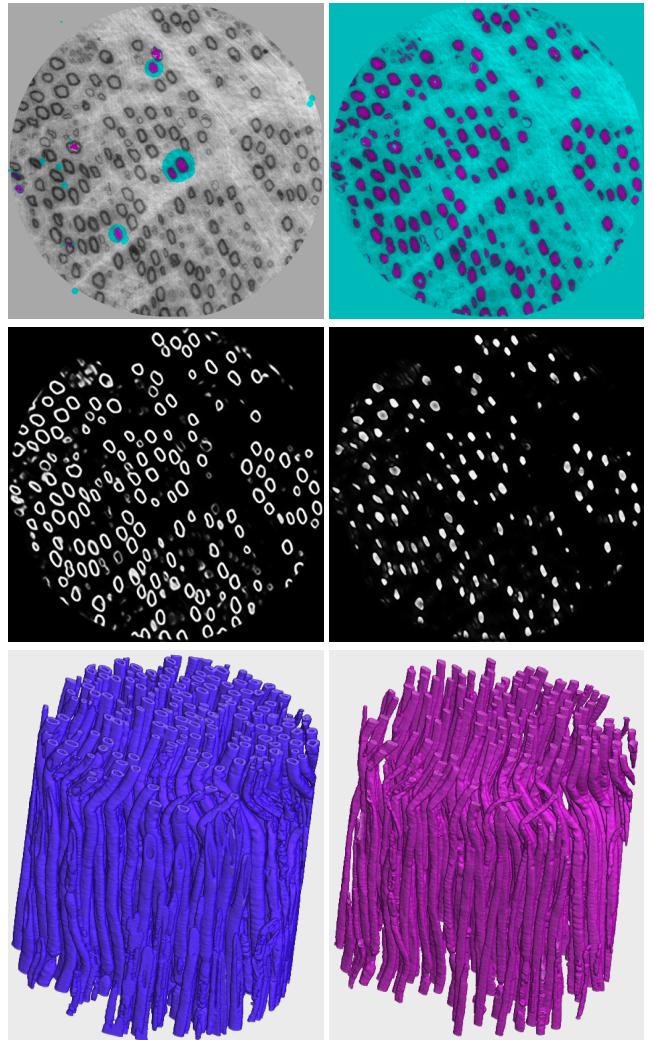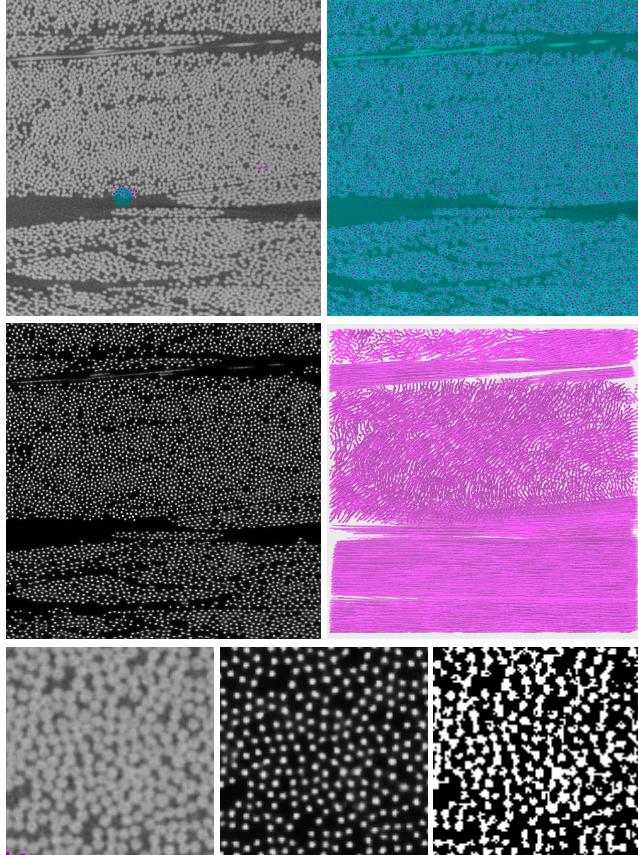12. L. Breiman, Machine learning **45**(1), 5 (2001)

Fig. 6 A three-class segmentation of an image showing onion cells. In the *top* row an image with manual input and a segmentation into three classes: background (cyan), nucleus (purple) and wall (magenta). In the *bottom* row the probability images for the wall and the nucleus class. Settings used are $M = 9$ and $K = 4000$.

Fig. 5 Volumetric segmentation of glass fibres. In the *top* row a slice with manual input indicating fibre centres (magenta) on a background (cyan) class, together with a resulting two-class segmentation. The *middle* row shows a layer of a probability image corresponding to the fibre centres class. On the *middle* to the *right* is the output of processing a full volumetric stack. From the 3D visualisation it is evident that fibres form clusters of different orientations. The *bottom* row shows a zoom-in on the central part of the image slice, together with the corresponding probability image and (for comparison) a corresponding segmentation obtained by directly thresholding the image intensities. Settings used in this experiment are $M = 9$ and $K = 4000$.

Fig. 7 A three-class segmentation of a histopathology image. In the *top* row an original colour image. The extent of the manual input and a corresponding segmentation into three classes are shown on the *right*, with a frame cropped to show the central column of the image. Likewise, in the *bottom* row we show the probability images for the two classes for the central part of the image. In the *bottom right* the final result obtained through additional postprocessing to distinguish individual cells. Settings used are $M = 5$ and $K = 4000$.

13. I. Arganda-Carreras, V. Kaynig, C. Rueden, K.W. Eliceiri, J. Schindelin, A. Cardona, H. Sebastian Seung, Bioinformatics p. 180 (2017)
14. J. Schindelin, I. Arganda-Carreras, E. Frise, V. Kaynig, M. Longair, T. Pietzsch, S. Preibisch, C. Rueden, S. Saalfeld, B. Schmid, et al., Nature methods **9**(7), 676 (2012)
15. Y. LeCun, Y. Bengio, G. Hinton, Nature **521**(7553), 436 (2015)
16. A.B. Dahl, V.A. Dahl, in *22nd International Conference on Pattern Recognition (ICPR)* (IEEE, 2014), pp. 142–147
17. A.B. Dahl, V.A. Dahl, in *Scandinavian Conference on Image Analysis* (Springer, 2015), pp. 26–37
18. V.A. Dahl, A.B. Dahl, in *International Conference on Scale Space and Variational Methods in Computer Vision* (Springer, 2017), pp. 421–432
19. M.J. Emerson, V.A. Dahl, K. Conradsen, L.P. Mikkelsen, A.B. Dahl, Composites Science and Technology **160**, 208 (2018). DOI 10.1016/j.compscitech.2018.03.027
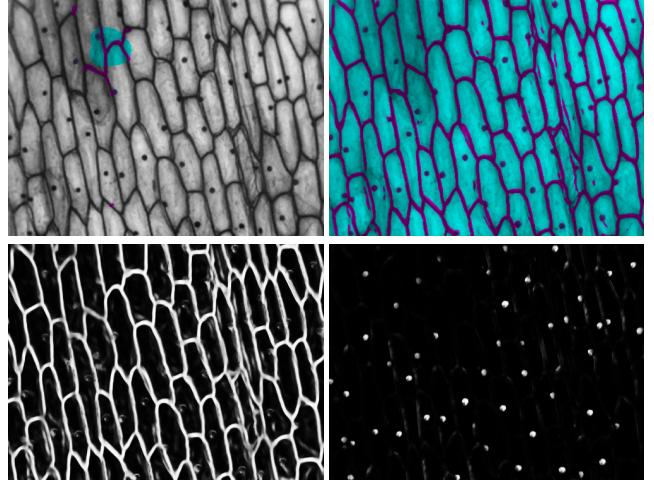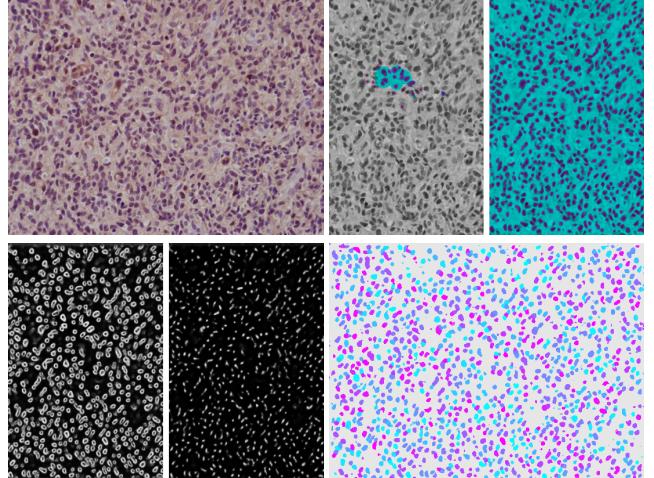
20. O. Chapelle, B. Scholkopf, A. Zien (eds.), *Semi-supervised learning* (The MIT Press, 2006)
21. D. Nister, H. Stewenius, in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2 (IEEE, 2006), vol. 2, pp. 2161–2168
22. D.G. Lowe, International journal of computer vision **60**(2), 91 (2004)