Fast segmentation from blurred data in 3D fluorescence microscopy

Martin Storath, Dennis Rickert, Michael Unser, Andreas Weinmann

Abstract—We develop a fast algorithm for segmenting 3D images from linear measurements based on the Potts model (or piecewise constant Mumford-Shah model). To that end, we first derive suitable space discretizations of the 3D Potts model which are capable of dealing with 3D images defined on non-cubic grids. Our discretization allows us to utilize a specific splitting approach which results in decoupled subproblems of moderate size. The crucial point in the 3D setup is that the number of independent subproblems is so large that we can reasonably exploit the parallel processing capabilities of the graphics processing units (GPU). Our GPU implementation is up to 18 times faster than the sequential CPU version. This allows to process even large volumes in acceptable runtimes. As a further contribution, we extend the algorithm in order to deal with non-negativity constraints. We demonstrate the efficiency of our method for combined image deconvolution and segmentation on simulated data and on real 3D widefield fluorescence microscopy data.

Index Terms—Image segmentation, 3D images, Potts model, piecewise constant Mumford-Shah model, parallelization, GPU, non-negativity constraints

I. INTRODUCTION

SEGMENTATION is an important component of many image processing pipelines. For instance, a segmentation of the image can be used to extract objects of interest such as, e.g., nuclei of cells in microscopic images. Furthermore, it can serve for dimensionality reduction by aggregating related pixels into larger regions (“superpixels”). In many imaging modalities however, the image of interest cannot be recorded directly, but it is reconstructed from indirect linear measurements given by the device. The relation between the imaged object and the data is given by some imaging operator; for example, the data acquisition in computed tomography is described by the Radon transform, and the operators in imaging modalities are often assumed to be shift-invariant which results in operators of convolution type.

Segmentation of indirectly measured images is typically performed in two steps: first the image is reconstructed from the measured data by (stabilized) inversion of the imaging operator; then, the resulting image is partitioned or segmented. The advantage of this two stage procedure is that there is a large number of methods available for the separate steps. For instance in 3D fluorescent microscopy, the deblurring stage can be accomplished by the Tikhonov-Miller method, the (regularized) Richardson-Lucy method [1–3], or wavelet methods [4], and the segmentation stage by k-means, active contours [5], or graph cuts [6], to mention only a few. On the flipside, the results of such two stage approaches are suboptimal since the distortions of the reconstruction step propagate to the segmentation step. In particular when data is poor, combined approaches, i.e. joint reconstruction and segmentation, exhibit better performance [7–12]. We here focus on a classical, yet challenging, variational segmentation approach called the Potts model or the piecewise-constant Mumford-Shah model [13–15]. For indirectly measured images, it is given by

\[ u^* = \arg \min_u \gamma \|\nabla u\|_0 + \|Au - f\|_2^2. \quad (\mathcal{P}_\gamma) \]

Here, A denotes the mathematical model for the imaging operator, and f denotes the measured data. The inverse Potts model is an optimization problem of two concurring objectives. The term \(\|Au - f\|_2^2\) ensures fidelity to the measured data whereas the Potts prior \(\|\nabla u\|_0\) enforces a small jump set (or discontinuity set) of a minimizer. The relative weight of the two target functionals is regulated by the parameter \(\gamma\); the higher \(\gamma\), the smaller the jump set of a solution. The value of the jump-penalty \(\|\nabla u\|_0\) can only be small if \(u\) is a piecewise constant function. Thus, a solution \(u^*\) can be regarded as reconstruction from data \(f\) with a piecewise constant prior; see also [7, 12]. The desired segmentation is encoded in the piecewise constant function \(u^*\): the constant regions of \(u^*\) correspond to the segments, and the jump set of \(u^*\) corresponds to the segment boundaries. Moreover, the function value of \(u^*\) on a segment gives an estimate on the mean intensity on that segment.

A. Prior and related work

The Potts model is named after R. Potts who considered a related regularizer in his seminal work in statistical mechanics [13]. The classical Potts model, i.e., \(\mathcal{P}_\gamma\) with \(A = id\), was introduced to image processing by Geman and Geman [14] in a statistical framework. From a calculus of variation point of view, the problem was first considered by Mumford and Shah [15]; there it is also known as piecewise constant Mumford-Shah model. In recent work, \(\mathcal{P}_\gamma\) is also referred to as the \(\ell_0\) gradient model [16, 17]. The Potts problem is – even for \(A = id\) – computationally challenging: it is known to be NP hard in dimension greater than one [18, 19]. Nevertheless, the Potts model is a very popular model for image segmentation; see [14, 19–21] to mention only some examples. For the case \(A = id\), classical algorithms are based on simulated annealing.
and on approximation by elliptic functionals [12]. A non-exhaustive selection of popular approaches for $A = \text{id}$ are active contours [5, 23–25], graph cuts [6, 19], convex relaxations [26–29], semi-global matching [30, 31], iterative thresholding [16, 32], fused coordinate descent [33], region fusion [17], and alternating direction method of multipliers (ADMM) [34]. Although the latter approach is computationally more demanding than other recent approaches [16, 17, 33], it gives the best quality in practice; see [17]. In [6, 35], graph cuts are used for partitioning 3D images with discretized label spaces.

In this work, we propose an efficient algorithmic framework for joint 3D-image reconstruction and multilabel segmentation based on the Potts model. We show its applicability for joint reconstruction and segmentation in 3D fluorescence microscopy. In particular, we contribute by addressing the challenges (i) and (ii) as follows. Regarding (i), we consider the discretization of the Potts prior in 3D for non-cubic regular grids reflecting the different resolutions w.r.t. different directions. Our discretization is based on weighted directional difference scheme which has been introduced in the two-dimensional case by Chambolle [51] and which has been refined in a previous work of the authors [34]. For non-cubic regular grids, we here propose a new discretization that takes the different resolutions w.r.t. the different directions into account. We here show the practical benefits in numerical experiments on synthetic and real data.

Regarding (ii), we extend a splitting strategy which was originally developed by the authors for 2D image partitioning [12]. It relies on the alternating directions method of multipliers (ADMM) which leads to subproblems that can be efficiently solved by dynamic programming. As mentioned before, the ADMM has shown the largest potential with respect to quality but is computationally demanding in 3D. The crucial point of the proposed design is that the algorithm is parallelizable in a way that it leads to a significant speedup on the GPU. We demonstrate the computational advantage of the subproblem parallelization by implementing a tailored parallel GPU version of the algorithm which is, on a single GPU, up to around 18 times faster than a sequential implementation. This is important to be able to process larger volumes in reasonable time.

Besides the specifics of the 3D setup, we further address the following general issue: (iii) The gray-values (or color-values) of a reconstructed image often correspond to physical quantities that cannot take negative values; for example the intensities in fluorescence microscopy. To take this into account, we impose a non-negativity constraint which leads to the non-negative Potts model

$$\arg \min_u \gamma \|\nabla u\|_0 + \|Au - f\|_2^2, \quad \text{subject to } u \geq 0. \quad (\mathcal{P}_\gamma^\ast)$$

For this model, we derive a new algorithm which is virtually as efficient as the unconstrained version.

Finally, we stress that our method neither requires a priori knowledge on the number of segments and labels nor on the gray-values of a solution. Further, the method does not require any initial guess of the solution.

In our numerical experiments, we focus on 3D image segmentation from blurred measurements. We first evaluate the capabilities of our method on synthetic datasets with ground truth. Then we apply it to real data of a wide field fluorescence microscope. Using our parallel GPU implementation, we achieve a joint deconvolution and segmentation of a $400 \times 400 \times 50$ image in less than three minutes. We note that the experiments are based on depth-invariant point spread functions. More sophisticated models for 3D fluorescence microscopy use depth-variant point spread functions which can give more accurate restorations [52, 53]. Further, we have restricted ourselves to imaging modalities which are

### B. Challenges in 3D and Contribution

Here, we focus on the inverse Potts problem for 3D images. Besides the problems already encountered in 2D, there are additional challenges in 3D: (i) In many imaging modalities, the resolution in the $x_3$-direction is typically much lower than in the other directions. This has to be taken into account when discretizing the Potts prior $\|\nabla u\|_0$. (ii) The main challenge for 3D data is the high computational effort due to two reasons: the number of pixels/voxels is typically much larger in 3D volumes than in planar images, and the 3D neighborhood system is bigger than the 2D system (e.g., 8-connected neighborhood in 2D vs. 26-connected neighborhood in 3D). Hence there is need for efficient algorithms that are able to process even large 3D images in reasonable time. In particular, algorithms that can be parallelized well are important to obtain feasible runtimes without loss of segmentation quality.
modeled by linear measurements, and in particular convolution operators. We note that the present algorithm can be adapted to nonlinear imaging operators, e.g., those appearing in electrical impedance tomography [54]. The relevant condition is that we can solve the corresponding Tikhonov-regularized problem within reasonable time.

C. Organization of the paper

In Section II we derive a suitable discretization of the Potts prior for 3D images defined on cubic and non-cubic regular grids. In Section III we present the proposed computational approach to \([x^2]\), and we elaborate on the solution of the arising subproblems. Section IV deals with numerical experiments on synthetic data and on real data.

II. Discretization of the Potts Prior for 3D Images

As a first step, we derive a suitable discretization of the jump penalty \(\|\nabla u\|_0\). In the following, we assume that \(u\) is a vector-valued \(N\)-dimensional image, i.e., \(u \in X = \mathbb{R}^{d_1 \times d_2 \times \ldots \times d_N \times C}\). Here, \(C\) denotes the number of channels of the image; for example, \(C = 3\) for color images. We use a finite difference discretization of the form

\[
\|\nabla u\|_0 = \sum_{i=1}^{S} \omega_i \|\nabla_{a_i} u\|_0.
\]

(1)

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

\[
\|\nabla_{a_i} u\|_0 = |\{i = (i_1, \ldots, i_N) : u_{i_i} \neq u_{i+e_{a_i}}\}|,
\]

where we use the notation \(u_{i_i} = (u_{i_1}, \ldots, u_{i_C}) \in \mathbb{R}^C\) to denote the data sitting in the voxel with coordinates \(i\). \(\nabla_{a_i} u\) can be seen as a finite difference approximation of the directional derivative of \(u\) with respect to the direction \(a\). Optimized discretizations of this form were initiated for planar images \(N = 2\) by Chambolle [51]. In the simplest case, \(\mathcal{N}\) consists of the coordinate unit vectors \(e_1, \ldots, e_N\) and the weights \(\omega_1, \ldots, \omega_N\) are all equal to one. Unfortunately, this simple discretization corresponds to measuring lengths in the Manhattan metric which may lead to undesired block building in the results; see [34, 51].

A. Near-isotropic discretization for 3D images defined on cubic grids

To get more isotropic discretizations, we utilize larger finite difference systems. Such a system can be chosen rather arbitrarily in principle; the only hard constraint is that the system spans \(\mathbb{R}^N\). In practice however, it is reasonable to start with the coordinate unit vectors \(e_1, \ldots, e_N\) and to successively enlarge the neighborhood system by adding additional directions. These extra directions can be planar diagonal finite difference vectors such as \(e_1 + e_2, e_1 + e_3, \ldots\), and space diagonals such as \(e_1 + \ldots + e_N\).

In the remainder, we will focus on 3D. There, 26-connected neighborhood systems are frequently used. Identifying antipodal points, this leads to the finite difference system

\[
\mathcal{N} = \{(1, 0, 0), (0, 1, 0), (0, 0, 1),
\]

\[
(1, 1, 0), (1, -1, 0), (1, 0, 1),
\]

\[
(1, 0, -1), (0, 1, 1), (0, 1, -1),
\]

\[
(1, 1, 1), (1, 1, -1), (1, -1, -1), (-1, 1, -1)\}.
\]

(2)

The system contains three coordinate directions, six planar diagonal directions, and four volumetric diagonals, which amounts to 13 orientations in total. We now derive appropriate weights \(\omega\). To this end, we note that a finite difference system gives rise to a metric on \(\mathbb{R}^N\) induced by the norm

\[
\|x\|_\mathcal{N} = \sum_{i=1}^{S} \omega_i \|x, a_i\|_1, \quad x \in \mathbb{R}^N.
\]

(3)

A reasonable condition for the weights is that the length of the vectors in the neighborhood system coincides with their Euclidean length [12]; that is,

\[
\|a_i\|_\mathcal{N} = \|a_i\|_2 \quad \text{for all } s = 1, \ldots, S.
\]

(4)

Plugging these \(S\) conditions into (3), we get the linear system of equations

\[
T \omega = r,
\]

(5)

where \(T\) is an \(S \times S\) matrix given by \(T_{ij} = \langle a_i, a_j \rangle\) and the righthand side by \(r_s = \|a_i\|_2\).

**Proposition 1.** The weights associated with the 26 connected neighborhood system in (2) are given by

\[
\omega_1 = \ldots = \omega_3 = \frac{2}{\sqrt{3}} - 1,
\]

\[
\omega_4 = \ldots = \omega_9 = \frac{1}{6} (3 \sqrt{3} - 2 \sqrt{5}),
\]

\[
\omega_{10} = \ldots = \omega_{13} = \frac{1}{6} (3 - 3 \sqrt{2} + \sqrt{5}).
\]

(6)

A proof is provided in the appendix. Next, we quantify the gain of this discretization. A reasonable measure of isotropy is the ratio of the shortest to the longest vector (measured in the Euclidean length) on the unit ball with respect to \(\|\cdot\|_\mathcal{N}\),
see [51]. The value 1 corresponds to perfect isotropy. For the simple 6-connected neighborhood we obtain the unfavorable ratio 1/√3 ≈ 0.58. For the proposed neighborhood system, the ratio is approximately 0.91, which is much closer to the desired value 1. This is illustrated in Figure [1].

B. Discretization for 3D images defined on non-cubic regular grids

The sampling rates of 3D images are often not equal with respect to all directions. For example in microscopic images, the sampling density in the x1-x2-plane is typically much higher than in the x3 direction. Then the above discretization can lead to distortions of the length measure in x3 direction. Therefore, we here design weights that account for images with direction-dependent sampling densities. Let us assume that the distance between two pixels in x, direction is given by \( \Delta; \) i.e., the voxels are cubes with the side lengths \( \Delta_1, \Delta_2, \Delta_3 \), and the size of these cubes do not change w.r.t. location (homogeneity). Thus, the area \( \delta_1 \) of the interface between two adjacent pixels in x1 direction is given by \( \delta_1 = \Delta_2 \Delta_3 \). Likewise, the interface areas of the other directions are given by \( \delta_2 = \Delta_1 \Delta_3 \) and \( \delta_3 = \Delta_1 \Delta_2 \), respectively. In order to account for the voxel size properly, a jump with respect to the coordinate direction \( x_n \) should get the cost \( \delta_n \). This motivates to use \( \|x\|_\delta = \sqrt{\sum_n (\delta_n x_n)^2} \) as jump penalty for the direction \( x \in \mathbb{R}^N \). Considering the directions \( x = a_s \) for \( s = 1, \ldots, S \), leads to the new right hand side \( r' \) in (5) given by

\[
r'_s = \|a_s\|_\delta, \quad \text{for } s = 1, \ldots, S.
\]

Note that the left hand side of (5) does not change because it encodes the pixel neighborhood system.

When we solve the linear system (5) using the adapted righthand side \( r' \) from (7), it can happen that the solution \( \omega \) has negative components. This happens for example for the sampling steps \( \Delta = (1, 1, 2) \), and its corresponding unit jump penalty vector \( \delta = (2, 2, 1) \). To avoid this undesired effect, we impose a non-negativity constraint, and we solve the linear system in a least squares sense; that is, we construct the weights by

\[
\min_{\omega} \|T \omega - r\|_2^2, \quad \text{s.t. } \omega \geq 0.
\]

This problem can be solved using for example an active set method [55] Chapter 23. If one of the weights gets equal to zero, it does not contribute to the overall penalty. Therefore, we remove all zero weights and the corresponding finite difference vector. In the above example, \( \Delta = (1, 1, 2) \), the weight that corresponds to the orientation \((0, 0, 1)\) is equal to zero. Thus, we remove that from our system \( \mathcal{N} \) and end up with 12 finite difference vectors. In Figure 2 we visualize the jump penalties that correspond to the weights computed via (5). The practical advantage of incorporating the non-equidistant sampling steps into the weight design is illustrated in Figure 3 for synthetic data and in Figure 4 for real data. At any rate the computational cost for solving the constrained least squares problem (8) is negligible because its dimension is low and because it has to be performed only once per imaging setup.

III. A PARALLEL ALGORITHM FOR SEGMENTING 3D IMAGES FROM LINEAR MEASUREMENTS

Based on the above discretization (4) of the Potts prior, we now develop our strategy for the Potts problem with non-negativity constraints (5). We first present a splitting based on ADMM, and then provide the solvers for the arising subproblems. We then discuss the computational complexity and the potential for parallelization. The adaption to the unconstrained version (4) is straightforward.

A. A splitting approach based on ADMM

Our first step is to reformulate (4) with the discretization (1) as the constrained optimization problem

\[
\begin{align*}
\text{minimize} & \quad \sum_{i=1}^S \gamma \omega_s \|\nabla a_s u_s\|_2 + \|Av - f\|_2^2, \\
\text{subject to} & \quad v - u_s = 0, \quad \text{for all } 1 \leq s \leq S \\
& \quad u_s \geq 0, \quad \text{for all } 1 \leq s \leq S.
\end{align*}
\]

Fig. 2: Near-isotropic discretization on the non-cubic regular grid \( \Delta = (1, 1, 2) \). A point \( x \) on the surface corresponds to the direction \( x/|x| \), and its penalty into that direction is \( |x| \). The desired penalties define an ellipsoid (left) with the semi-axis lengths \( \delta = (2, 2, 1) \). The anisotropic discretization (middle) reproduces the Euclidean length in the coordinate directions but puts too much weight on the diagonals. The proposed weight design (right) yields a satisfactory approximation to the desired Euclidean length.
was blurred by a Gaussian kernel of standard deviation 10 and corrupted by i.e., the sampling steps

\[ \Delta = \frac{192}{x} \times \frac{192}{x} \]

Fig. 3: Practical effects of the non-equidistant weight design. The image was blurred by a Gaussian kernel of standard deviation 10 and corrupted by Gaussian white noise with standard deviation \( \sigma = 0.1 \). The image has size 192 \times 192 \times 96, and we assume half of the \( x_1, x_2 \) resolution in \( x_3 \) direction, i.e., the sampling steps \( \Delta = (1, 1, 2) \). We observe that incorporating the non-equidistant sampling produces a better result than the equidistant one. (See Figure 5 for a comparison on real data.)

Here, \( u_s \) denotes the split variable corresponding to the direction \( a_s \), where \( 1 \leq s \leq S \). We form the augmented Lagrangian \( \mathcal{L}_\mu \) of the target functional including the equality constraints which reads as

\[
\mathcal{L}_\mu((u_s)_{s=1}^S, v, (A_s)_{s=1}^S) = \sum_{s=1}^S \gamma_{s} \|\nabla a_s u_s\|_0 + \langle A_s, v - u_s \rangle + \frac{\mu}{2} \|v - u_s\|_2^2 + \|Av - f\|_2^2.
\]

The non-negativity persists as constraints for \( u_s \geq 0 \), for all \( 1 \leq s \leq S \). Here, the parameter \( \mu > 0 \) control how strongly the splitting variables are coupled. The \( A_s \in \mathbb{R}^m \), \( 1 \leq s \leq S \), take the role of Lagrange multipliers. This specific splitting allows us to use ADMM for (\ref{3}). (See, e.g., \cite{48} for an extensive overview on ADMM.) In each iteration we minimize \( \mathcal{L} \) sequentially with respect to \( u_1, \ldots, u_S \), and \( v \) and subject to the inequality constraints. Then, we perform gradient ascent steps in the Lagrange multipliers \( A_1, \ldots, A_S \).

Then, we obtain by some algebraic manipulation (see appendix) that

\[
\arg \min_{u_s \geq 0} \mathcal{L}_\mu = \arg \min_{u_s \geq 0} \frac{2\gamma_{s}}{\mu} \|\nabla a_s u_s\|_0 + \|u_s - w_s\|_2^2.
\]

where \( w_s = v + \frac{1}{\mu} \). For the minimization with respect to \( v \), we obtain

\[
\arg \min_v \mathcal{L}_\mu = \arg \min_v \|Av - f\|_2^2 + \frac{\mu}{2} \left\| v - \frac{1}{S} \sum_{s=1}^S (u_s - \frac{\lambda_s}{\mu}) \right\|_2^2. \tag{12}
\]

Iterating these minimization steps and the gradient ascent steps in the Lagrange multipliers, we obtain the following procedure:

\[
\begin{align*}
\lambda_1^{k+1} &= \arg \min_{\lambda_1 \geq 0} \frac{2\gamma_{s}}{\mu} \|\nabla a_1 u_1\|_0 + \|u_1 - (v^k + \frac{\lambda_1}{\mu})\|_2^2, \\
\vdots \\
\lambda_s^{k+1} &= \arg \min_{\lambda_s \geq 0} \frac{2\gamma_{s}}{\mu} \|\nabla a_s u_s\|_0 + \|u_s - (v^k + \frac{\lambda_s}{\mu})\|_2^2, \\
\lambda_s^{k+1} &= \frac{1}{\mu} \sum_{s=1}^S (u_s^{k+1} - \frac{\lambda_s^{k+1}}{\mu})^2, \quad \text{for all } 1 \leq s \leq S. \tag{13}
\end{align*}
\]

We use as coupling parameters an increasing sequence \( (\mu_k)_{k \in \mathbb{N}} \) which we will specify later on.

A pseudocode of the procedure is given in Algorithm [1]. First we initialize the variable \( v \) with a solution of the corresponding minimization problem in (\ref{13}). In the main loop, we calculate the solutions of the subproblems in the first \( S+1 \) lines of (\ref{13}), and then perform a gradient ascent step in the Lagrange multiplier. This loop is iterated until we reach a maximum number of iterations or another stopping criterion such as the maximum squared \( \ell^2 \) distance of the split variables \( \|u_s - u_t\|_2^2 \) for \( s \neq t \). As final result, we return the average of the split variables \( u_1, \ldots, u_S \). Next we explain how to solve the subproblems involved in Algorithm [1].

**B. Solution of the Tikhonov step**

Let us first consider the subproblem in the \( (S+1) \)-th line of (\ref{13}). This is a Tikhonov problem of the form

\[
\nu^* = \arg \min \|Av - f\|_2^2 + \tau (v - z)_+^2.
\]

Differentiating the functional with respect to \( v \) reveals that a minimizer \( \nu^* \) satisfies

\[
2A^*(Av^* - f) + 2\tau (v^* - z) = 0,
\]

where \( A^* \) denotes the adjoint of \( A \) (or the transpose if \( A \) is given as real-valued system matrix). Resolving this equation gives us the linear system

\[
(A^* A + \tau I) \nu^* = A^* f + \tau z, \tag{14}
\]

where \( I \) denotes the identity. Thus, the computation of \( \nu^* \) reduces to solving a linear system of equations. As the linear system is typically very large we use iterative solvers such as the conjugate gradients methods. To ensure fast convergence we utilize a “warmstart”. That is, we use the solution of the previous iteration \( \nu^* \) as initial guess for the computation of \( \nu^{k+1} \).
Algorithm 1: Proposed algorithm for the inverse Potts model in 3D with non-negativity constraints

**Input:** Measured data \( f \in \mathbb{R}^3 \), model parameter \( \gamma > 0 \)

**Output:** 3D image \( u \in \mathbb{R}^{d_1 \times d_2 \times d_3} \)

begin

\( u_0, \lambda_0 \) with 0 and \( v = \arg \min_{u \in \mathbb{R}^3} \| Az - f \|^2 + \frac{\mu_v}{2} \| z \|^2; \)

\( k \leftarrow 1; \)

repeat

/ Potts steps */

for \( s = 1, \ldots, S \) do

\( u_s \) Solve Potts problems with data \( g = v + \lambda_s / \mu_k \)

line-wise w.r.t. the direction \( a_s \), using the parameter \( \gamma' = \frac{2 \mu_v}{\mu_k} \), see Algorithm 2

(All univariate solvers can be executed in parallel)

end

/ Tikhonov step */

\( v' = \arg \min_{u \in \mathbb{R}^3} \| Az - f \|^2 + \frac{\mu_v}{2} \| z - \frac{1}{3} \| \sum_{i=1}^3 (u_i - \frac{1}{\mu_k}) \|^2; \)

/ Lagrange multiplier update step */

for \( s = 1, \ldots, S \) do

\( \lambda_s = \lambda_s + \mu_k (v - u_s); \)

end

\( k \leftarrow k + 1; \)

until stopping criterion reached;

return \( u = \frac{1}{3} \sum_{i=1}^3 u_i; \)

end

In various setups, the structure of the imaging operator can be exploited for a significant acceleration. In the simplest case of \( A \) being the identity we have that

\[
\hat{\gamma}^* = \frac{\hat{f} + \hat{z}}{1 + \tau},
\]

which is simply a weighted average of \( f \) and \( z \). If \( A \) describes the (circular) convolution with some convolution kernel \( \phi \), i.e., \( Au = \phi * u \), then we can use Fourier methods. Applying the Fourier transform on both sides of (13) implies that the Fourier transform of \( \gamma^* \), denoted by \( \hat{\gamma}^* \), is given by

\[
\hat{\gamma}^*_j = \frac{\phi^*_j \hat{f}_j + \tau \hat{z}_j}{|\phi^*_j|^2 + \tau},
\]

where \( j \) runs over all indices in the domain of \( \hat{\gamma}^* \).

For vector-valued images, we take advantage of the fact that the Tikhonov problem is separable with respect to the components. That means, we can solve it componentwise for each channel.

C. Solving the Potts subproblems with non-negativity constraints

Now we look at the first \( S \) lines of the proposed algorithm (13). There, we solve problems of the form

\[
\arg \min_{u \in \mathbb{R}^3} \gamma' \| \nabla u \|_0 + \| u - g \|^2, \]

with some \( \gamma' > 0 \), an image \( g \in X = \mathbb{R}^{d_1 \times d_2 \times \cdots \times d_N}; \) and a direction \( a \in \mathbb{Z}^N \setminus \{0\} \). For notational simplicity, we first consider the case of a single channel image, i.e., \( C = 1 \).

We observe that the problem decomposes into univariate Potts problems along paths indicated by the finite difference vector \( a \). That is, it remains to solve univariate Potts problems with non-negativity constraints of the form

\[
\arg \min_{u \in \mathbb{R}^3, u \geq 0} \gamma' \| \nabla u \|_0 + \| u - h \|^2. \tag{15}
\]

Here, \( h \) is a vector of length \( n \) where \( n \) is the length of a one dimensional line in direction \( a \in \mathbb{R}^3 \) with starting pixel \( b \in \mathbb{R}^3 \). To fix the ideas let us look at the example \( a_1 = (1, 0, 0) \). All 1D problems in these directions are parametrized by the offset pixels \( b = (1, i_2, i_3) \) for \( i_2 = 1, 2, \ldots, d_2 \) and \( i_3 = 1, 2, \ldots, d_3 \). (Recall that the image \( g \) is of dimension \( d_1 \times d_2 \times d_3 \).) For such an offset pixel \( b = (1, i_2, i_3) \), the data \( h \) is given by \( h_i = g_{i_2,i_3} + g_{b+ja} \). Thus we solve \( d_2 \cdot d_3 \) times a univariate Potts problem (15) with data \( h \) specified by the offset pixels \( b = (1, i_2, i_3) \) for \( i_2 = 1, 2, \ldots, d_2 \) and \( i_3 = 1, 2, \ldots, d_3 \). For a general direction \( a \), we proceed analogously with respect to the data

\[
h_j = g_{b+ja}
\]

and all suitable offset pixels \( b \). Note that for the diagonal directions, offset pixels from more than one face of the cube are involved to parametrize all 1D lines through the cube, and the lengths \( n \) of the 1D lines depend on the offset pixel and on the dimensions.

It is well known that the unconstrained version of the univariate Potts problem, i.e., \( \arg \min_{u \in \mathbb{R}^3} \gamma' \| \nabla u \|_0 + \| u - h \|^2 \), can be solved exactly by dynamic programming [15, 57–60]. A quadratic time and linear space algorithm for the univariate Potts problem was proposed by Friedrich et al. [61] which we describe next. To this end, it is convenient to use the Matlab-like notation \( h_{1:r} = (h_1, \ldots, h_r) \). We assume that we have already computed minimizers \( u^i \) of the Potts functionals \( P_r(u) = \gamma' \| \nabla u \|_0 + \| u - h_{1:r} \|^2 \) associated with the partial data \( h_{1:r} \) for each \( l = 1, \ldots, r-1 \) and some \( r \) smaller than the number \( n \) of elements of \( h \). Using this information, the Potts functional associated to data \( h_{1:r} \) can be efficiently minimized via

\[
\min_{u \in \mathbb{R}^3} P_r(u) = \min_{l=1, \ldots, r} P_r(u^l) + \gamma + \epsilon_{l:r}, \tag{16}
\]

where \( \epsilon_{l:r} \) is the quadratic deviation of the data \( h_{l:r} \) from its mean. Here we use the convention that \( u^0 \) is an empty vector and \( P_r(u^0) = -\gamma \). A corresponding minimizer reads \( u^l = (u^{l-1}, \mu_{r}, \ldots, \mu_{r}) \) where \( \mu_{r} \) is a minimizing argument of the righthand side of (15) and \( \mu_{r} \) denotes the mean value of \( (h_{r}, \ldots, h_{l}) \). We obtain a minimizer for full data \( d_b \) by successively computing \( u^l \) for each \( r = 1, \ldots, n \). The described method can be implemented in \( O(n^3) \) time and \( O(n) \) memory using pre-computation of the moments and storing only jump locations [61]. It has been shown that a significant amount of configurations are unreachable and thus can be skipped [54, 62] which improves the practical runtime by a fourfold to fivefold factor.

We next adapt the above algorithm such that it incorporates the non-negativity constraints. The basic dynamic programming principle carries over. To account for the non-negativity constraints we need to compute the quadratic deviation from the best constant approximation \( \epsilon_{l:r}^* \) on all intervals. To this
end, we observe that the best non-negative constant approximation \( \mu^*_{i,r} \) for \( h_{i,r} \) is given by

\[
\mu^*_{i,r} = \arg \min_{\mu \geq 0} \sum_{i=1}^{r} (\mu - h_i)^2 = \frac{\max(0, \sum_{i=1}^{r} h_i)}{r - l + 1}.
\]

This leads to the expression for the approximation error

\[
\epsilon^+_{i,r} = \min_{\mu \geq 0} \sum_{i=1}^{r} (\mu - h_i)^2
\]

\[
= \sum_{l=1}^{r} \mu^2_{l,r} - 2\mu_{l,r} h_i + h_i^2
\]

\[
= (r - l + 1) \mu^2_{l,r} - 2\mu_{l,r} h_i + \sum_{i=1}^{r} h_i^2
\]

\[
= - (r - l + 1) \mu^2_{l,r} + \sum_{i=1}^{r} h_i^2
\]

\[
= - \frac{\max(0, \sum_{i=1}^{r} h_i)}{r - l + 1} + \sum_{i=1}^{r} h_i^2
\]

The value \( \epsilon^+_{i,r} \) can be evaluated in \( O(1) \) when we precompute the cumulative sums \( M_r = \sum_{i=1}^{r} h_i \) and \( S_r = \sum_{i=1}^{r} h_i^2 \). Hence, we obtain

\[
\epsilon^+_{i,r} = \frac{\max(0, M_r - M_{i-1})^2}{r - l + 1} + S_r - S_{l-1}.
\]

We provide the complete pseudocode of the solver in Algorithm 2.

The solver can be easily adapted to deal with vector valued data, that is, when \( h_i \in \mathbb{R}^C \) for all \( i \). To this end we consider the squared approximation errors \( \epsilon^+_{i,r,k} \) on each channel \( k = 1, \ldots, C \), which are given by \( \epsilon^+_{i,r,k} = \min_{\mu \geq 0} \sum_{i=1}^{r} (\mu - h_{i,k})^2 \). The necessary value of \( \epsilon^+_{i,r,k} \) then consists of the sum of the channelwise errors, thus \( \epsilon^+_{i,r} = \sum_{k=1}^{C} \epsilon^+_{i,r,k} \). We stress that this procedure is different to channelwise solution of the Potts problem. In particular, the presented procedure enforces a common jump set of all channels.

D. Computational complexity and parallelization

Assume for simplicity that the number of pixels is equal in each direction, i.e., that \( d = d_1 = \ldots = d_N \). In each step of \( l \), we have to solve \( d^{N-1} \) Potts problems of size \( d \). With the quadratic complexity of the Potts solver we get the complexity \( O(d^{N+1}) = O(M^{(N+1)/N}) \) where \( M \) denotes the number of pixels. For vector valued data with \( C \) channels we get \( O(M^{(N+1)/NC}) \). It remains to treat the Tikhonov step which is typically the most time consuming part. Here, a linear system of equations has to be solved which amounts to cubic complexity w.r.t. \( M \) in general. Frequently, the special structure of \( A \) can be exploited for a significant speedup. If we are in the above described case that we can utilize fast Fourier transforms, then the evaluation of \( \nu^* \) is \( O(CM \log M) \) where \( M \) is the number of pixels and \( C \) the number of channels. The complexity grows linearly in the number of channels for vector-valued data.

The time complexity can be reduced significantly by parallelization. In fact, since the univariate Potts solver acts in each step on nonintersecting lines on the data, we can execute all \( d^{N-1} \) Potts solvers in parallel. For example, for \( N = 3 \) and \( d = 100 \), \( d^2 = 10000 \) univariate problems can be solved in parallel each step. Currently available graphics processing units have typically around thousand processing units. Thus, GPU parallelization can even be almost fully exploited in relatively small problem sizes.

IV. Experimental results

We illustrate the potential of our method for joint deconvolution and segmentation of 3D wide field fluorescence microscopy images. Before that, we discuss the choice of the parameters and our implementation.

A. Algorithmic setup

In our numerical experiments, we used the sampling-step adapted weights of Section II-B with the ordering of the \( p_1 \) given by \( \mu \). We initialized all \( \nu \) by arg \( \min_{\nu} ||Av - f||^2 + \frac{\alpha_k}{2} ||\nu||^2 \) and all other variables \( u_s, \lambda_s \) by 0. We did not observe significant changes in the results for other orderings of \( \alpha_k \) or other reasonable initializations. By contrast, the choice of the coupling sequence \( \mu_k \) can have a visible effect on the result and the runtime: the lower the increments of \( \mu_k \), the better the results but also the higher the runtime. The condition

\[
\begin{algorithm}
\textbf{Input:} Data vector \( h \in \mathbb{R}^\text{n} \), model parameter \( \gamma' > 0 \).
\textbf{Output:} Global minimizer \( u^* \) of the univariate Potts problem with non-negativity constraint \( \text{[15]} \).
\begin{enumerate}
\item[\textbf{begin}]
\item[\(*\text{ Find the optimal jump locations }*/]
\item[\( M_0 \leftarrow 0; \)]
\item[\( S_0 \leftarrow 0; \)]
\item[\(*\text{ init cumulative moments }*/]
\item[\( \text{for } r \rightarrow 1, \ldots, n \ \text{do} \)]
\item[\( M_r \leftarrow M_{r-1} + h_r; \)]
\item[\( S_r \leftarrow S_{r-1} + h_r^2; \)]
\item[\(*\text{ cumulative first and second moments }*/]
\item[\( P_r \leftarrow S_r - \max(M^2_r/r, 0); \)]
\item[\(*\text{ init } \nu^* \text{ with } \epsilon^+_{i,r} \text{ */}]
\item[\( J_r \leftarrow 0; \)]
\item[\(*\text{ init right-most jump location }*/]
\item[\( \text{for } l \rightarrow r, r-1 \ldots 2 \ \text{do} \)]
\item[\(*\text{ non-neg. mean squared deviation of } h_r \text{ */}]
\item[\( \epsilon^+ \leftarrow S_r - S_{l-1} - \max(M_r - M_{l-1}, 0)^2/(r - l + 1); \)]
\item[\(*\text{ Skip unreachable configurations }*/]
\item[\( \text{if } d + \gamma' > P_r \ \text{then } \)]
\item[\(*\text{ compute candidate value } p \text{ */}]
\item[\( p \leftarrow P_{r-1} + \gamma' + \epsilon^+; \)]
\item[\(*\text{ store new best Potts value }*/]
\item[\( J_r \leftarrow l; \)]
\item[\(*\text{ update right-most jump location }*/]
\item[\textbf{end}]
\item[\textbf{end}]
\item[\(*\text{ Reconstruct the minimizer from the optimal jump locations }*/]
\item[\( r \leftarrow n; l \leftarrow J_r; \)]
\item[\textbf{while } r > 0 \ \text{do} \]
\item[\(*\text{ set } \mu_{i,r} \text{ on partition } l+1, \ldots, r \text{ */}]
\item[\( u^*_{i,r} \leftarrow \max(0, (M_r - M_l)/(r - l)); \)]
\item[\(*\text{ go to next jump } */]
\item[\( r \leftarrow l; l \leftarrow J_r; \)]
\item[\textbf{end}]
\item[\textbf{return } u^*; \]
\item[\textbf{end}]
\end{enumerate}
\end{algorithm}
\[ \sum_k \mu_k^{-1/2} < \infty \] ensures that the split variables converge to each other. We omit the proof, as it is a straightforward adaption of the proof for the two-dimensional setup presented in \[12\]. In practice, slow progressions such as \( \mu_k = \mu_0 k^{2/3} \), have the highest potential with respect to reconstruction quality. Thus, if computation time is not critical, we suggest to use slow progressions. Otherwise, one may try a faster progression such as \( \mu_k = \mu_0 2^k \). Here we use the progression \( \mu_k = \mu_0 1.05^k \) which we found to be a fair tradeoff between quality of the results and computational costs in the present setup; see also \[42\].

### B. Implementation and runtimes

We have created two implementations of our method: a sequential version for the CPU and a parallel version for the GPU.

For the sequential version, we utilize Matlab with C++/MEX files for the time-critical operations; i.e., the univariate Potts solvers. The main advantage of this version is its flexible integration with the Matlab computing framework. It is, for example, possible to supply arbitrary proximal mapping operators as Matlab function handles without changes to the implementation of the algorithm. Furthermore, this implementation improves upon the computation time of our previously proposed Java-based implementation for the two-dimensional Potts problem (with \( A = id \)) \[43\] by a factor two.

The parallel version has been implemented for the GPU, based on C++ and the CUDA framework. The GPU significantly decreases the computation time provided that the proximal mapping can be evaluated efficiently on the GPU. In the present work, the imaging operator possess an efficient implementation based on the CUDA FFT library cuFFT. In our experiment (Table I), we processed 3D images of different sizes with our sequential CPU and our parallel GPU version. Benchmarking was performed using a workstation with 32GB RAM, a core i7-4770k processor with 3.5GHz and a GTX Titan Black GPU. The software used was Matlab 2015b, CUDA 7.0 and gcc 4.7.3. From Table I we observe that the parallel version leads to an eighteen-fold speedup. This potential is almost fully exploited even for moderate image dimensions of \( 200 \times 200 \times 50 \). Large volumes of dimension \( 400 \times 400 \times 50 \) can be processed in less than three minutes. The tradeoff is that the GPU implementation requires more effort to be adapted to alternate imaging setups; e.g. changing the proximal mapping requires implementation of the operator in CUDA and recompilation of the entire algorithm.

The core algorithm has two parts which benefit massively from GPU acceleration, i.e. solving the 1D Potts problems in parallel, and, to a lower degree, the calculation of the weighted mean solution from all 1D results. We implemented the solver for the 1D Potts problem as a custom kernel. The calculation of the weighted mean solution is implemented using cuBLAS, a GPU accelerated variant of the basic linear algebra library. This part does not require any custom kernels. The RAM requirements on the GPU scale with \( O(MS) \), where \( M \) is the total number of pixels of the image, and where \( S \) denotes the number of directions used for the discretization of the Potts problem

\[400 \times 400 \times 50 \] can be processed in less than three minutes. The tradeoff is that the GPU implementation requires more effort to be adapted to alternate imaging setups; e.g. changing the proximal mapping requires implementation of the operator in CUDA and recompilation of the entire algorithm.

The core algorithm has two parts which benefit massively from GPU acceleration, i.e. solving the 1D Potts problems in parallel, and, to a lower degree, the calculation of the weighted mean solution from all 1D results. We implemented the solver for the 1D Potts problem as a custom kernel. The calculation of the weighted mean solution is implemented using cuBLAS, a GPU accelerated variant of the basic linear algebra library. This part does not require any custom kernels. The RAM requirements on the GPU scale with \( O(MS) \), where \( M \) is the total number of pixels of the image, and where \( S \) denotes the number of directions used for the discretization of the Potts problem.

### C. Simulation study

We performed a simulation study for joint reconstruction and segmentation from blurred data. The experimental setup is as follows. As test data, we created two series of structurally different segmentation phantoms of size \( 200 \times 200 \times 50 \). The first series consists of 50 cell-type phantoms. We used the function \( \text{phantom3d} \) \[4] to create three overlapping ellipsoids to represent a cell body, a boundary of 11 pixels width to represent a cell boundary and complemented it with a few ellipsoids representing organelles. Cell body, boundary and organelles were each given distinct gray values. Two typical examples of the generated phantoms are shown in Figure 4. Each of the 50 phantoms has different randomly shifted ellipsoid centers. The second series of phantoms consists of 50 images of size \( 200 \times 200 \times 50 \) containing randomly rotated and shifted versions of geometric primitives. Two typical examples are depicted in Figure 5. All phantoms have a dynamic range in \([0, 100]\). We blurred the phantoms using a Gaussian point spread function. Then, we corrupted the resulting image by Poisson noise. We produced different image qualities (or noise levels) by simulating different photon counts. To this end, we scaled the blurred images by the factor \( 10^p \), for \( p = 0, \ldots, 4 \), before imposing the Poisson noise. Note that a higher scaling factor means higher data quality.

To measure the segmentation quality of the computed multilabel segmentations, we use the Rand index (RI) \[63\] \[64\].

<table>
<thead>
<tr>
<th>Image Dimension</th>
<th>Sequential (CPU)</th>
<th>Parallel (GPU)</th>
<th>Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>( 50 \times 50 \times 50 )</td>
<td>32.0 sec</td>
<td>7.4 sec</td>
<td>4.3</td>
</tr>
<tr>
<td>( 100 \times 100 \times 50 )</td>
<td>161.3 sec</td>
<td>15.8 sec</td>
<td>10.2</td>
</tr>
<tr>
<td>( 200 \times 200 \times 50 )</td>
<td>705.5 sec</td>
<td>49.9 sec</td>
<td>14.1</td>
</tr>
<tr>
<td>( 300 \times 300 \times 50 )</td>
<td>1656.8 sec</td>
<td>97.5 sec</td>
<td>17.0</td>
</tr>
<tr>
<td>( 400 \times 400 \times 50 )</td>
<td>3080.6 sec</td>
<td>169.2 sec</td>
<td>18.2</td>
</tr>
</tbody>
</table>

**TABLE I:** Comparison of CPU and GPU runtimes. For the largest image, the parallelized GPU implementation yields a speedup of around 18 times. This way, even large volumes can be processed in reasonable time.

\(^1\)Implementation of the method in \[43\] is available at \url{http://pottslab.de}

\(^2\)Available from the Matlab File Exchange.
Fig. 4: Qualitative results of the simulation study for the first type of phantoms: The first two and the last two columns show the segmentation results for the two example phantoms in (a). The first and the third column have a higher data quality (scaling factor $10^4$), the second and fourth column have a lower quality (scaling factor 10). The segmentations in (c) and (d) are encoded as piecewise constant gray-level images where the gray-values correspond to the estimated mean value on each segment. In (e), the results of (d) are shown in pseudocolors. We observe that the baseline method captures the cell boundary well but also produces fuzzy results in the body and the organelles. The proposed method better recovers the organelles; it has a moderate oversegmentation at the boundary. (See Table II for quantitative results.)
TABLE II: Quantitative results of the simulation study for the first class of phantoms. The table shows the average Rand index over all simulated 50 phantoms along with the standard deviation. The rows are indexed by the scaling of the data where higher scaling means higher data quality (corresponding to a lower noise level). The proposed method consistently yields a higher Rand index. (See Figure 5 for qualitative results.)

<table>
<thead>
<tr>
<th>Image scaling (higher number means higher data quality)</th>
<th>Baseline method</th>
<th>Proposed method</th>
</tr>
</thead>
<tbody>
<tr>
<td>$10^4$</td>
<td>0.9301 ± 0.0052</td>
<td>0.9645 ± 0.0031</td>
</tr>
<tr>
<td>$10^3$</td>
<td>0.9286 ± 0.0052</td>
<td>0.9642 ± 0.0036</td>
</tr>
<tr>
<td>$10^2$</td>
<td>0.9221 ± 0.0063</td>
<td>0.9610 ± 0.0036</td>
</tr>
<tr>
<td>$10^1$</td>
<td>0.9152 ± 0.0058</td>
<td>0.9500 ± 0.0040</td>
</tr>
<tr>
<td>$10^0$</td>
<td>0.8872 ± 0.0077</td>
<td>0.9265 ± 0.0049</td>
</tr>
</tbody>
</table>

D. Results on 3D widefield fluorescence microscopy data

Next we show experimental results on real 3D widefield fluorescence microscopy data. The data was acquired using an Olympus CellR widefield microscope. The point spread function (PSF) of the image acquisition setup is shown in Figure 6. It exhibits good localization in the $x_1$-$x_2$ plane but has a wide spread in the $x_3$ direction. The sampling steps are given by $\Delta_1 = \Delta_2 = 64.5$ nm and $\Delta_3 = 160$ nm. For the baseline approach we used 30 Richardson–Lucy iterations, and performed a k-means clustering where we chose the number of labels $K$ to match the expected number of labels in the image. (For further comparison, results for other deconvolution methods on the present data set can be found in [65–67].) For the proposed method, we used the initial coupling parameter $\mu_0 = 10^{-6}$ as proposed in [12]. The model parameter $\gamma$ was chosen depending on the experiment as indicated below.

1) Fluorescence bead: The first real data set of size $256^3$ is the image of a fluorescence bead with known diameter $2.5 \mu m$. We have manually chosen the $\gamma$-parameter to capture the topological structure of the hollow sphere. Figure 6 shows the segmentation of the bead from the linear measurements. We observe that the baseline approach overestimates the size of the bead with respect to the $x_3$ direction. The output of the proposed method matches significantly better with the true dimensions of the bead. We observe that it slightly oversegments the bead (two segments instead of one). However, such oversegmentation can typically be resolved relatively easy by region merging.

2) C-elegans embryo: The second real data set is the microscopic image of a C-elegans embryo using a DAPI

The outcome of the first series of phantoms is reported in Figure 4 and in Table II. Figure 4 exemplarily shows the outcome of the experiment for two different phantoms and two different noise levels. In Table II, we report the quantitative results averaged over all 50 cell-type phantoms. The analogous results for the geometric primitive phantoms are given in Figure 5 and in Table III. Qualitatively, we observe that the proposed method provides more contiguous segments than the baseline method. Quantitatively, we observe that the proposed method provides a higher Rand index than the baseline method consistently for all noise levels. This indicates an improved segmentation quality. Both methods tend to oversegmentation, i.e., the number of produced segments is larger than the number of true segments, which can be addressed by post-processing.
fluorescent stain. We downsampled the original image size by the factor two in each dimension to obtain an image of size $337 \times 357 \times 56$. We have manually chosen the $\gamma$-parameter such that the white organelles in the cells are captured as individual segments and that the main cell body is only moderately oversegmented. Figure 7 shows the result of this experiment. The baseline approach captures most nuclei sharply but it has some incomplete segments and has partially fuzzy boundaries in the $x_3$ direction. The proposed method has better localized boundaries. Again, we observe a slight oversegmentation in the cell body. The runtime amounts to around 43 seconds for the baseline method, to around 29 minutes for the sequential CPU implementation, and to around 291 seconds using the parallel GPU implementation.

V. CONCLUSION AND FUTURE RESEARCH

We have developed a fast algorithm for segmenting 3D images from linear measurements based on the Potts model; in particular, we addressed the specific challenges of the 3D setup. First, we proposed a new design for discretizations of the jump penalty to address the issue of non-cubic grids frequently appearing in 3D. Secondly, we presented a highly parallel implementation on the GPU which allowed us to process even large volumes in reasonable computation time. Thirdly, to take the non-negativity of many physical quantities into account, we incorporated corresponding constraints into the model and into the proposed algorithmic scheme. We have illustrated the potential of our method for joint image deconvolution and segmentation of 3D widefield microscopy images. We note that the method can be adapted to other imaging setups such as x-ray computed tomography and photoacoustic tomography. The feasibility condition is that we can compute the proximal mapping of the data term in reasonable time.

Future directions of research include the adaption to further imaging setups such as magnetic particle imaging and the adaption to the (piecewise smooth) Mumford-Shah functional as in [68].

ACKNOWLEDGEMENT

M. Storath and A. Weinmann acknowledge the support by the German Research Foundation (DFG grant STO1126/2-1, WE5886/4-1). M. Unser was supported by the European Research Council under the European Union’s Seventh Framework Programme (FP7/2007-2013) / ERC grant agreement no. 267439. The authors would like to thank Daniel Sage and Ferréol Soulez for valuable discussions on deconvolution of widefield microscopy images.

APPENDIX

A. Proof of Proposition 1

First let $x \in N$ be one of the coordinate directions, say $x = a_1 = (1, 0, 0)$. Then, the righthand side of (4) is given by $||x||_2 = 1$. For the lefthand side, we obtain

$$||x||_N = \sum_{s=1}^{11} \omega_s |x(a_s)| = \sum_{s=1}^{11} \omega_s |(a_s)_1| = \omega_1 + \omega_4 + \omega_5 + \omega_6 + \omega_7 + \omega_{10} + \omega_{11} + \omega_{12} + \omega_{13}$$
By symmetry, we have $\omega_1 = \ldots = \omega_3$, $\omega_4 = \ldots = \omega_8$, and $\omega_{10} = \ldots = \omega_{13}$. Plugging this into (4) leads to the condition

$$\omega_1 + 4\omega_4 + 4\omega_{10} = 1.$$  

Note that the other finite difference vectors in coordinate directions, $a_2$ and $a_3$, lead to the same equation. Similarly, the planar diagonals, such as $x = (1,1,0)$, lead to the equation

$$2\omega_1 + 6\omega_4 + 4\omega_{10} = \sqrt{2},$$

and the volume diagonals (e.g., $x = (1,1,1)$) yield

$$3\omega_1 + 6\omega_4 + 6\omega_{10} = \sqrt{3}.$$

Solving this linear system with respect to $\omega_1$, $\omega_4$ and $\omega_{10}$ gives us the weights claimed in (6).

B. Derivation of the ADMM iteration

The first step is to rewrite the augmented Lagrangian (10) in its scaled form; see (56). To this end, we rewrite

$$\langle \lambda_i, v - u_i \rangle + \frac{\mu}{2} ||v - u_i||_2^2 = \frac{\mu}{2} ||v - u_i + \frac{\lambda_i}{\mu}||_2^2 - \frac{1}{2\mu} ||\lambda_i||_2^2,$$

so that

$$L_p(\{u_i\}_{i=1}^S, v, \{\lambda_i\}_{i=1}^S)$$

$$= \sum_{s=1}^S \gamma \omega_s ||\nabla_{a_s} u_s||_0 + \frac{\mu}{2} ||v - u_s + \frac{\lambda_s}{\mu}||_2^2 - \frac{1}{2\mu} ||\lambda_s||_2^2 + ||Av - f||_2^2.$$  

(18)

Then, as the last two terms do not depend on $u_s$, minimization with respect to $u_s$ gives (11). To bring $\arg\min \ L_p(\{u_i\}_{i=1}^S, v, \{\lambda_i\}_{i=1}^S)$ into a convenient form we combine the second $S$ terms as

$$\sum_{s=1}^S \frac{\mu}{2} ||v - u_s + \frac{\lambda_s}{\mu}||_2^2 = \frac{\mu S}{2} ||v - \frac{1}{S} \sum_{s=1}^S (u_s - \frac{\lambda_s}{\mu})||_2^2 + C.$$  

with a constant $C \in \mathbb{R}$ that does not depend on $a$. Here, we used the fact that

$$\sum_i x_i(a - b_i)^2 = \left(\sum_i x_i\right)(a - \frac{\sum_i b_i x_i}{\sum_i x_i})^2 + C'$$

for $a, b_1, \ldots, b_N \in \mathbb{R}$ and $x_1, \ldots, x_N > 0$ and a constant $C' \in \mathbb{R}$ that does not depend on $a$, see e.g. (12). Minimizing with respect to $v$ yields (12).

REFERENCES

Fig. 7: Second real data experiment: Deconvolution and segmentation of the widefield microscopy image of a C-Elegans embryo. The data (a) was acquired using the same setup as in Figure 6. (b) The baseline approach captures parts of the nuclei well, but it has also incomplete segments and fuzzy boundaries in $x_3$ direction. (c,d) The proposed method produces more regular and complete segments. We observe a moderate over-segmentation in the cell body.


