Joint Reconstruction-Segmentation on Graphs

Budd, Jeremy M.; van Gennip, Yves; Latz, Jonas; Parisotto, Simone; Schonlieb, Carola Bibiane

DOI
10.1137/22M151546X

Publication date
2023

Document Version
Final published version

Published in
SIAM Journal on Imaging Sciences

SIAM J. IMAGING SCIENCES
Vol. 16, No. 2, pp. 911–947

Joint Reconstruction-Segmentation on Graphs*

Jeremy M. Budd†, Yves van Gennip‡, Jonas Latz§, Simone Parisotto¶, and Carola-Bibiane Schönlieb¶

Abstract. Practical image segmentation tasks concern images which must be reconstructed from noisy, distorted, and/or incomplete observations. A recent approach for solving such tasks is to perform this reconstruction jointly with the segmentation, using each to guide the other. However, this work has so far employed relatively simple segmentation methods, such as the Chan–Vese algorithm. In this paper, we present a method for joint reconstruction-segmentation using graph-based segmentation methods, which have been seeing increasing recent interest. Complications arise due to the large size of the matrices involved, and we show how these complications can be managed. We then analyze the convergence properties of our scheme. Finally, we apply this scheme to distorted versions of "two cows" images familiar from previous graph-based segmentation literature, first to a highly noised version and second to a blurred version, achieving highly accurate segmentations in both cases. We compare these results to those obtained by sequential reconstruction-segmentation approaches, finding that our method competes with, or even outperforms, those approaches in terms of reconstruction and segmentation accuracy.

Key words. image reconstruction, image segmentation, joint reconstruction-segmentation, graph-based learning, Ginzburg–Landau functional, Merriman–Bence–Osher scheme, total variation regularization

MSC codes. 05C99, 34B45, 35R02, 65F60, 94A08

DOI. 10.1137/22M151546X

1. Introduction.

Two tasks which lie at the heart of many applications in image processing are *image reconstruction*—the task of reconstructing an image from noisy, distorted, and/or incomplete observations—and *image segmentation*, the task of separating an image

*Received by the editors August 11, 2022; accepted for publication (in revised form) January 25, 2023; published electronically June 7, 2023.
https://doi.org/10.1137/22M151546X
Funding: The authors received support from the Philip Leverhulme Prize; the Royal Society Wolfson Fellowship; EPSRC advanced career fellowship EP/V029428/1; EPSRC grants EP/S026045/1, EP/T003553/1, EP/N014588/1, and EP/T017961/1; Wellcome Innovator Awards 215733/Z/19/Z and 221633/Z/20/Z; the European Union Horizon 2020 research and innovation programme under Marie Skłodowska-Curie grant agreement 777826 NoMADS; the Cantab Capital Institute for the Mathematics of Information; the Alan Turing Institute; and the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy - EXC-2047/1 - 390685813.
†Hausdorff Centre for Mathematics, Rheinische Friedrich-Wilhelms-Universität Bonn, Bonn, Germany (jeremy.budd@hcm.uni-bonn.de).
‡Delft Institute of Applied Mathematics, Technische Universiteit Delft, Delft, The Netherlands (y.vangennip@tudelft.nl).
§Maxwell Institute for Mathematical Sciences and School of Mathematical and Computer Sciences, Heriot-Watt University, Edinburgh, UK (j.latz@hw.ac.uk).
¶Department of Applied Mathematics and Theoretical Physics, University of Cambridge, Cambridge, UK (sp751@cam.ac.uk,cbs31@cam.ac.uk).

into its "important" parts. But in practice, the latter task is not independent of the former: when we seek to segment an image, we will not typically have access to the true image, but rather must reconstruct the image from imperfect observations. That is, in practice a segmentation task is often a *reconstruction*-segmentation task.

Over the last decade, the framework of "PDEs on graphs" has yielded highly effective techniques for image segmentation. In this paper, we will exhibit a technique for reconstruction-segmentation within this framework. In particular, we will incorporate into this framework the method of *joint reconstruction-segmentation*, which is an approach that performs the reconstruction and segmentation together, using each to guide the other, with the goal of improving the quality of the segmentation compared to performing the tasks in sequence. Previous implementations of this approach have employed relatively simple segmentation techniques. The key contribution of this paper will be to show how the more sophisticated graph-PDE-based segmentation techniques can be employed in joint reconstruction-segmentation.

1.1. Image reconstruction background. The general setting for image reconstruction is that one has some observations y of an image x^*, which are related via

$$y = \mathcal{T}(x^*) + e, \tag{1.1}$$

where $\mathcal{T}$ is the *forward model*, typically a linear map, and e is an error term (e.g., a Gaussian random variable). Solving (1.1) for x^*—given y, $\mathcal{T}$, and the distribution of e—is in general an ill-posed problem. A key approach to solving (1.1), pioneered by Tikhonov [46] and Phillips [40], has been to solve the variational problem

$$\operatorname*{argmin}_x \ \mathcal{R}(x) + D(\mathcal{T}(x), y), \tag{1.2}$$

where $\mathcal{R}$ is a *regularizer*, encoding a priori information about x^*, and D enforces fidelity to the observations and encodes information about e.

1.2. Image segmentation background. One of the most celebrated methods for image segmentation is that of Mumford and Shah [37]. This method segments an image $x : \Omega \to \mathbb{R}$ by constructing a piecewise smooth $\tilde{x} \approx x$ and a set of contours Γ (the boundaries of the segments) minimizing a given segmentation energy, namely the Mumford–Shah functional

$$\mathrm{MS}(\tilde{x}, \Gamma) := \int_{\Omega \setminus \Gamma} |\nabla \tilde{x}|^2 \, d\mu + \alpha \int_{\Omega} (x - \tilde{x})^2 \, d\mu + \beta |\Gamma|, \tag{1.3}$$

where μ is the Lebesgue measure. As MS is difficult to minimize in full generality, Chan and Vese [17] devised a method where $\tilde{x}$ is restricted to being piecewise constant.[1] This simplifies (1.3) to an energy which can be minimized via level-set methods. Some key drawbacks of these methods are that they can be computationally expensive, as one must solve a PDE; can be hard to initialize [23]; can perform poorly if the image has inhomogeneities [51]; and are constrained by the image geometry, and so are less able to detect large-scale nonlocal structures.

[1] Chan and Vese also add an extra energy term proportional to the area "inside" Γ.

These Mumford–Shah methods are related to Ginzburg–Landau methods (see, e.g., [21]), because the Ginzburg–Landau functional Γ-converges to total variation [36]. Because of this Γ-convergence (which also holds on graphs [47]), Bertozzi and Flenner [7] were inspired to develop a segmentation method based on minimizing the graph Ginzburg–Landau functional using the graph Allen–Cahn gradient flow. Soon after, Merkurjev, Kostić, and Bertozzi [34] introduced an alternative method using a graph Merriman–Bence–Osher (MBO) scheme. These "PDEs on graphs" methods have received considerable attention, both theoretical (see, e.g., [6, 33, 48]) and in applications (see, e.g., [14, 15, 24, 31, 35, 42]). In previous work, Budd and van Gennip [12] showed that the graph MBO scheme is a special case of a semidiscrete implicit Euler (SDIE) scheme for graph Allen–Cahn flow, and Budd, van Gennip, and Latz [13] investigated the use of this SDIE scheme for image segmentation and developed refinements to earlier methods that resulted in improved segmentation accuracy.

1.3. Joint reconstruction-segmentation background. Reconstruction-segmentation was traditionally approached *sequentially*: first reconstruct the image, then segment the reconstructed image. The key drawback of this method is that the reconstruction ignores any segmentation-relevant information. At the other extreme is the *end-to-end* approach: first collect training data $\{(y_n, u_n)\}$ of pairs of observations and corresponding segmentations, then use this data to learn (e.g., via deep learning) a map that sends y to u. However, this forgoes explicitly reconstructing x^* and can require a lot of training data, and the map can be a "black box" (i.e., it may be hard to explain its segmentation or prove theoretical guarantees).

Joint reconstruction-segmentation (a.k.a. simultaneous reconstruction and segmentation) lies between these extremes, seeking to perform the reconstruction and segmentation simultaneously, using each to guide the other. It was first proposed by Ramlau and Ring [43] for CT imaging, with related (but extremely varied) methods later developed for other medical imaging tasks (for an overview, see [19, section 2.4]). An extensive theoretical overview of *task-adapted reconstruction* was developed in Adler et al. [2], which found that joint reconstruction-segmentation produced more accurate segmentations than both the sequential and end-to-end approaches. These methods were enhanced in Corona et al. [19] using Bregman methods, and a number of theoretical guarantees were proved about this enhanced scheme. However, these approaches have mostly relied on Mumford–Shah or Chan–Vese methods for the segmentation.

1.4. Contributions and outline. The primary contribution of this work will be a joint reconstruction-segmentation method based around the joint minimization problem

$$\min_{x\in\mathbb{R}^{N\times\ell},u\in\mathcal{V}} \mathcal{R}(x) + \alpha\|\mathcal{T}(x) - y\|_F^2 + \beta \mathrm{GL}_{\varepsilon,\mu,f}(u, \Omega(\mathcal{F}(x), z_d)),$$

where x is the reconstruction, u is the segmentation, the first two terms describe a reconstruction energy as in (1.2), and the final term is a segmentation energy using the graph Ginzburg–Landau energy. The use of this energy is motivated by the success of the graph Ginzburg–Landau-based segmentation methods described in subsection 1.2. These objects, and other groundwork required for this paper, will all be defined in section 2. In particular, in this paper

i. we will present an iterative scheme for solving this minimization problem, which alternately updates the candidate reconstruction and the candidate segmentation (section 3);
ii. we will devise algorithms for computing the steps of this iterative scheme (sections 4 to 6); we compute the reconstruction update by linearizing the corresponding variational problem (section 4), and we compute the segmentation update via the SDIE scheme (section 5);
iii. we will demonstrate the convergence of this iterative scheme to critical points of the joint minimization problem (section 7);
iv. we will apply this scheme to highly noised and to blurred versions of the "two cows" image familiar from [7, 11, 13, 34] (section 8). Our scheme will exhibit very accurate segmentations which compete with or outperform sequential reconstruction-segmentation approaches.

2. Various groundwork.

2.1. Framework for analysis on graphs. We begin by giving a framework for analysis on graphs, abridging Budd [11, section 2], which itself is abridging van Gennip et al. [48].

Let (V, E, ω) be a finite, undirected, weighted, and connected graph with neither multi-edges nor self-loops. The finite set V is the *vertex set*, $E \subseteq V^2$ is the *edge set* (with $ij \in E$ if and only if $ji \in E$ for all $i, j \in V$), and $\{\omega_{ij}\}_{i,j\in V}$ are the *weights*, with $\omega_{ij} \geq 0$, $\omega_{ij} = \omega_{ji}$, and $\omega_{ii} = 0$, and $\omega_{ij} > 0$ if and only if $ij \in E$. We define function spaces

$$\mathcal{V} := \{u : V \to \mathbb{R}\}, \qquad \mathcal{V}_X := \{u : V \to X\}, \qquad \mathcal{E} := \{\varphi : E \to \mathbb{R}\}$$

if $X \subseteq \mathbb{R}$. For a parameter $r \in [0, 1]$, and writing $d_i := \sum_j \omega_{ij}$ for the *degree* of vertex $i \in V$, we define inner products on $\mathcal{V}$ and $\mathcal{E}$ (and hence inner product norms $\|\cdot\|_{\mathcal{V}}$ and $\|\cdot\|_{\mathcal{E}}$):

$$\langle u, v\rangle_{\mathcal{V}} := \sum_{i\in V} u_i v_i d_i^r, \qquad \langle \varphi, \phi\rangle_{\mathcal{E}} := \frac{1}{2}\sum_{i,j\in V} \varphi_{ij}\phi_{ij}\omega_{ij}.$$

Next, we introduce the graph variants of the gradient and Laplacian operators:

$$(\nabla u)_{ij} := \begin{cases} u_j - u_i, & ij \in E, \\ 0 & \text{otherwise,} \end{cases} \qquad \text{and} \qquad (\Delta u)_i := d_i^{-r} \sum_{j\in V} \omega_{ij}(u_i - u_j),$$

where the graph Laplacian Δ is positive semidefinite and self-adjoint with respect to $\mathcal{V}$. As shown in [48], these operators are related via $\langle u, \Delta v\rangle_{\mathcal{V}} = \langle \nabla u, \nabla v\rangle_{\mathcal{E}}$. We can interpret Δ as a matrix. Define $D := \operatorname{diag}(d)$ (i.e., $D_{ii} := d_i$, and $D_{ij} := 0$ otherwise) to be the *degree matrix*. Then writing ω for the matrix of weights ω_{ij} we get

$$\Delta := D^{-r}(D - \omega).$$

The choice of r is important. For $r = 0$, $\Delta = D - \omega$ is the standard *unnormalized (or combinatorial) Laplacian*. For $r = 1$, $\Delta = I - D^{-1}\omega$ (where I is the identity matrix) is the *random walk Laplacian*. There is also an important Laplacian not covered by this definition:

the *symmetric normalized Laplacian* $\Delta_s := I - D^{-1/2}\omega D^{-1/2}$. For image segmentation it is important to use a normalized Laplacian (see [7, section 2.3]), so we shall henceforth take $r = 1$.

2.2. The graph Ginzburg–Landau functional. In this paper, we shall use graph-based segmentation methods based on minimizing the graph Ginzburg–Landau functional. The basic form of this functional is

$$\mathrm{GL}_\varepsilon(u) := \frac{1}{2}\,||\nabla u||_{\mathcal{E}}^2 + \frac{1}{\varepsilon}\,\langle W \circ u, \mathbf{1}\rangle_{\mathcal{V}},$$

where W is a double-well potential and $\varepsilon > 0$ is a parameter. In particular, following Budd [11], we shall be taking W to be the *double-obstacle potential*:

$$W(s) := \begin{cases} \frac{1}{2}x(1-x) & \text{for } 0 \le x \le 1, \\ \infty & \text{otherwise.} \end{cases}$$

Furthermore, we define the *graph Ginzburg–Landau functional with fidelity* by

$$\mathrm{GL}_{\varepsilon,\mu,f}(u) := \frac{1}{2}\,||\nabla u||_{\mathcal{E}}^2 + \frac{1}{\varepsilon}\,\langle W \circ u, \mathbf{1}\rangle_{\mathcal{V}} + \frac{1}{2}\langle u - f, M(u-f)\rangle_{\mathcal{V}},$$

where $M := \mathrm{diag}(\mu)$ for $\mu \in \mathcal{V}_{[0,\infty)}$ the *fidelity parameter* and $f \in \mathcal{V}_{[0,1]}$ is the *reference*. We define $Z := \mathrm{supp}(\mu)$, which we call the *reference data*. Note that μ_i paramaterizes the strength of the fidelity to the reference at vertex i. We may assume without loss of generality that f is supported on Z.

It is worth briefly describing why minimizing $\mathrm{GL}_{\varepsilon,\mu,f}$ is a good way to segment an image. The first term penalizes the segmentation u if two vertices with a high edge weight are in different segments, encouraging the segmentation to group similar vertices together. The second term wants the segmentation to be binary. The third term penalizes u for disagreeing with an a priori segmentation, propagating those a priori labels to the rest of the vertices.

It will be useful for our joint reconstruction-segmentation scheme to redefine $\mathrm{GL}_{\varepsilon,\mu,f}$ as a function of both u and ω. A simple calculation gives that

$$\begin{aligned}\mathrm{GL}_{\varepsilon,\mu,f}(u,\omega) = {} & \frac{1}{2}\sum_{i,j\in V}\omega_{ij}(u_i-u_j)^2 + \frac{1}{2\varepsilon}\sum_{i,j\in V}\omega_{ij}(W(u_i)+W(u_j)) \\ & + \frac{1}{4}\sum_{i,j\in V}\omega_{ij}\left(\mu_i(u_i-f_i)^2 + \mu_j(u_j-f_j)^2\right).\end{aligned}$$

Note that this is linear in ω. We can therefore define $G_{\varepsilon,\mu,f} : \mathcal{V} \to \mathbb{R}^{V\times V}$ as

$$(G_{\varepsilon,\mu,f}(u))_{ij} = \frac{1}{2}(u_i-u_j)^2 + \frac{1}{2\varepsilon}(W(u_i)+W(u_j)) + \frac{1}{4}\left(\mu_i(u_i-f_i)^2 + \mu_j(u_j-f_j)^2\right)$$

such that

$$\mathrm{GL}_{\varepsilon,\mu,f}(u,\omega) = \mathrm{tr}(G_{\varepsilon,\mu,f}(u)^T\omega) =: \langle G_{\varepsilon,\mu,f}(u), \omega\rangle_F,$$

where $\langle\cdot,\cdot\rangle_F$ denotes the Frobenius inner product. Furthermore, note that if $v_i := \frac{1}{2}u_i^2 + \frac{1}{2\varepsilon}W(u_i) + \frac{1}{4}\mu_i(u_i-f_i)^2$, then $G_{\varepsilon,\mu,f}(u) = -uu^T + v\mathbf{1}^T + \mathbf{1}v^T$.

2.3. Turning an image into a graph. To represent an image as a graph, we first let our vertex set V be the set of pixels in the image and consider the image as a function $x : V \to \mathbb{R}^\ell$, where ℓ depends on whether the image is grayscale, RGB, or hyperspectral, etc. To build our graph, we construct *feature vectors* $\mathcal{F}(x) =: z : V \to \mathbb{R}^q$ where $\mathcal{F}$ is the *feature map* (which we shall assume to be linear). The philosophy behind these vectors is that vertices which are "similar" should have nearby feature vectors. What "similar" means is application-specific, e.g., [49] incorporates texture into the features, [7, 14] give other options, and there has been recent interest in deep learning methods for constructing features; see, e.g., [20, 35]. Next, we construct the weights on the graph by defining ω_{ij} to be given by some *similarity function* evaluated on z_i and z_j. There are a number of standard choices for the similarity function; see, e.g., [7, section 2.2]. For our choices for feature map and similarity measure see subsection 8.2 and (3.1), respectively.

2.4. The Nyström extension. A key practical challenge is that V is usually very large, and hence matrices such as the weight matrix $\omega \in \mathbb{R}^{V\times V}$ and Δ are much too large to store in memory. Instead, we shall compress these matrices using a technique called the Nyström extension, first introduced in Nyström [39] and developed for matrices in Fowlkes et al. [22]. Consider an $N \times N$ symmetric matrix A, written in block form

$$A = \begin{pmatrix} A_{XX} & A_{XX^c} \\ A_{X^cX} & A_{X^cX^c} \end{pmatrix},$$

where X is the *interpolation set*, with $|X| =: K \ll N$. Let $A_{XX} = U_X \Lambda U_X^T$, and let u_X^i be a column eigenvector from U_X with eigenvalue λ_i. The idea of the Nyström extension is to extend this eigenvector to a vector $((u_X^i)^T \ (u_{X^c}^i)^T)^T$ which is defined on all of V, using a quadrature rule. That is, $u_{X^c}^i$ is defined by $\lambda_i u_{X^c}^i = A_{X^cX} u_X^i$, which can be observed to resemble a quadrature rule for the eigenvalue problem $Au = \lambda_i u$. Let $U_{X^c} := \begin{pmatrix} u_{X^c}^1 & \cdots & u_{X^c}^K \end{pmatrix}$. Then $U_{X^c}\Lambda = A_{X^cX}U_X$ and so (assuming that A_{XX}^{-1} exists) $U_{X^c} = A_{X^cX}U_X\Lambda^{-1}$. Finally,

(2.1)
$$A \approx \begin{pmatrix} U_X \\ U_{X^c} \end{pmatrix} \Lambda \begin{pmatrix} U_X^T & U_{X^c}^T \end{pmatrix} = \begin{pmatrix} A_{XX} & A_{X^cX}^T \\ A_{X^cX} & A_{X^cX} A_{XX}^{-1} A_{X^cX}^T \end{pmatrix} = \begin{pmatrix} A_{XX} \\ A_{X^cX} \end{pmatrix} A_{XX}^{-1} \begin{pmatrix} A_{XX} & A_{X^cX}^T \end{pmatrix},$$

where in the first equality we used that $U_{X^c} = A_{X^cX}U_X\Lambda^{-1}$. The upshot of (2.1) is that we only need to store and calculate with A_{XX} and A_{X^cX}, which are much smaller than A. Also, (2.1) yields an efficient way to approximate matrix-vector products Av.

3. A joint-reconstruction-segmentation scheme on graphs.

3.1. Set-up. We will begin by formally stating our reconstruction-segmentation task.

Problem 3.1. *Let $x^* : Y \to \mathbb{R}^\ell$ be the image to be reconstructed and segmented. Let $y = \mathcal{T}(x^*) + e$ be observed data where the forward model $\mathcal{T}$ is differentiable and e is a random variable describing observation error. Let $x_d : Z \to \mathbb{R}^\ell$ be an already reconstructed and segmented reference image with a priori segmentation $f : Z \to \{0, 1\}$. Here Y and Z are disjoint finite sets. Given y, $\mathcal{T}$, x_d, and f, reconstruct $x \approx x^*$ and find $u : Y \cup Z \to \{0, 1\}$ such that $u|_Y$ segments x and $u|_Z$ is close to f.*

Next, we must incorporate this into a graph framework, following subsection 2.3. Let $V := Y \cup Z$ be the vertex set of our graph, and let the edge set E be given by $E := \{ij \mid i, j \in V, i \neq j\}$. Let $N := |Y|$ and $N_d := |Z|$. To encode the candidate reconstruction $x : Y \to \mathbb{R}^\ell$ and the reference image x_d in the weights on this graph, we define feature maps $\mathcal{F}$ and $\mathcal{F}_d$ and feature vectors $z : Y \to \mathbb{R}^q$ and $z_d : Z \to \mathbb{R}^q$ by $z := \mathcal{F}(x)$ and $z_d := \mathcal{F}_d(x_d)$. Since x_d and $\mathcal{F}_d$ are given, we hereafter treat z_d as given. We then define the edge weights via $\omega = \Omega(z, z_d)$, where $\Omega(z, z_d)$ is given by (for $\mathbf{z} := (z, z_d)$)

$$\Omega_{ij}(z, z_d) := e^{-\frac{\|\mathbf{z}_i - \mathbf{z}_j\|_F^2}{q\sigma^2}} \tag{3.1}$$

with $\|\cdot\|_F$ denoting the Frobenius norm.[2] The q in the denominator averages over the q components of $\mathbf{z}$ so that parameter choices for σ generalize better.

Note 3.2. The feature vectors z and z_d are defined so that z does not depend on x_d and z_d does not depend on x. This is a simplification, since x and x_d might be different parts of the same image and hence one might want z to partially depend on x_d. However, this simplification greatly aids in the following analysis, and in computation, as it means that the edge weights between vertices of Z can be considered fixed and given.

3.2. The joint reconstruction-segmentation scheme. To solve Problem 3.1, we will employ a variational approach. We will consider our candidate reconstructions x and segmentations u to be candidate solutions to the following joint minimization problem:

$$\min_{x \in \mathbb{R}^{N \times \ell}, u \in \mathcal{V}} \mathcal{R}(x) + \alpha \|\mathcal{T}(x) - y\|_F^2 + \beta \mathrm{GL}_{\varepsilon,\mu,f}(u, \Omega(\mathcal{F}(x), z_d)), \tag{3.2}$$

where $\mathcal{R}$ is a convex regularizer, which following Appendix B we shall assume can be written as $\mathcal{R}(x) = R(\mathcal{K}(x))$ for $\mathcal{K}$ a linear map and R convex and lower semicontinuous (l.s.c.) with convex conjugate R^*[3] proper, convex, l.s.c., and nonnegative. The first two terms in the objective functional are a standard Tikhonov reconstruction energy as in (1.2), and the final Ginzburg–Landau term is the segmentation energy. As this problem (and related variational problems considered in this paper) are nonconvex, by "solving" we will mean finding adequate local minimizers.

To avoid needing to solve the difficult problem (3.2) directly, we will use the following alternating iterative scheme[4] to approach solutions (where $\alpha, \beta, \eta_n, \nu_n$ are parameters):

$$x_{n+1} = \operatorname*{argmin}_{x \in \mathbb{R}^{N \times \ell}} \mathcal{R}(x) + \alpha \|\mathcal{T}(x) - y\|_F^2 + \beta \mathrm{GL}_{\varepsilon,\mu,f}(u_n, \Omega(\mathcal{F}(x), z_d)) + \eta_n \|x - x_n\|_F^2, \tag{3.3a}$$

$$u_{n+1} = \operatorname*{argmin}_{u \in \mathcal{V}} \beta \mathrm{GL}_{\varepsilon,\mu,f}(u, \Omega(\mathcal{F}(x_{n+1}), z_d)) + \nu_n \|u|_Y - u_n|_Y\|_{\mathcal{V}}^2. \tag{3.3b}$$

We can understand this scheme intuitively as iterating the following steps:

I. Given the current segmentation, update the reconstruction using the segmentation energy as an extra regularizer and the previous reconstruction as a momentum term.

II. Given the current reconstruction, update the segmentation using the previous segmentation of the image to be reconstructed as a momentum term.

[2] This choice of edge weight function is not arbitrary. The fact that it has a particularly well-structured derivative will be used in Appendix A and the fact that it is analytic will be used in section 7.

[3] That is, $R^*(p) := \sup_{x'} \langle p, x' \rangle - R(x')$.

[4] Note that the $\|\cdot\|_{\mathcal{V}}$ term depends on the degrees given by the weights $\Omega(\mathcal{F}(x_{n+1}), z_d)$.

3.3. Initialization. The simplest initial reconstruction x_0 would be $x_0 := \mathcal{T}^+(y)$, where $\mathcal{T}^+$ is the (Moore–Penrose) pseudoinverse of $\mathcal{T}$ (see [27, section 5.5.2]). However, in practice $\mathcal{T}^+(y)$ can be too poorly structured to give a good initial segmentation. Also, the pseudoinverse can be highly unstable [27, section 5.5.3] and does not exist for nonlinear $\mathcal{T}$. Thus, we favor initializing by applying a cheap and better behaved reconstruction method to y. The initial segmentation u_0 is constructed by segmenting x_0 via the SDIE methods to be described in section 5.

4. Solving (3.3a).

We now describe how we compute (approximate) solutions to (3.3a). This minimization problem is highly computationally challenging, so we will simplify it by linearizing (3.3a). This reduces the problem to one which can be solved by standard methods.

4.1. Linearizing (3.3a). The challenging term in (3.3a) is the Ginzburg–Landau energy term. Recall from subsection 2.2 that this can be written

$$\beta\langle G_n, \Omega(\mathcal{F}(x), z_d)\rangle_F \simeq \underbrace{\beta\langle (G_n)_{YY}, \Omega_{YY}(\mathcal{F}(x))\rangle_F}_{=:\mathscr{F}_1(\mathcal{F}(x))} + \underbrace{2\beta\langle (G_n)_{YZ}, \Omega_{YZ}(\mathcal{F}(x), z_d)\rangle_F}_{=:\mathscr{F}_2(\mathcal{F}(x))},$$

where for a matrix A, $A_{YZ} := (A_{ij})_{i\in Y, j\in Z}$ and likewise for A_{YY}, and where

$$G_n := G_{\varepsilon,\mu,f}(u_n) = -u_n u_n^T + v_n \mathbf{1}^T + \mathbf{1} v_n^T, \tag{4.1}$$

for v_n defined by $(v_n)_i := \frac{1}{2}(u_n)_i^2 + \frac{1}{2\varepsilon} W((u_n)_i) + \frac{1}{4}\mu_i((u_n)_i - f_i)^2$. Let us assume that our candidate minimizer for (3.3a) is close to x_n (this assumption will become more reasonable the higher the value of η_n is). Then we can make the following approximation:

$$\begin{aligned}
&\mathscr{F}_1(\mathcal{F}(x)) + \mathscr{F}_2(\mathcal{F}(x)) \\
&\approx \mathscr{F}_1(\mathcal{F}(x_n)) + \mathscr{F}_2(\mathcal{F}(x_n)) + \langle x - x_n, \nabla_x \mathscr{F}_1(\mathcal{F}(x_n)) + \nabla_x \mathscr{F}_2(\mathcal{F}(x_n))\rangle \\
&= \langle x, g_n\rangle + \text{constant terms (in } x),
\end{aligned}$$

where $g_n := \nabla_x \mathscr{F}_1(\mathcal{F}(x_n)) + \nabla_x \mathscr{F}_2(\mathcal{F}(x_n))$. We will describe how to compute g_n in Appendix A. Using this approximation, we can approximate (3.3a) by solving

$$\operatorname*{argmin}_{x\in\mathbb{R}^{N\times\ell}} \mathcal{R}(x) + \langle x, g_n\rangle_F + \alpha\|\mathcal{T}(x) - y\|_F^2 + \eta_n\|x - x_n\|_F^2. \tag{4.2}$$

Defining $\tilde{x}_n := x_n - \frac{1}{2}\eta_n^{-1} g_n$, (4.2) is equivalent to

$$\operatorname*{argmin}_{x\in\mathbb{R}^{N\times\ell}} \mathcal{R}(x) + \alpha\|\mathcal{T}(x) - y\|_F^2 + \eta_n\|x - \tilde{x}_n\|_F^2. \tag{4.3}$$

This is of the form of a standard variational image reconstruction problem (1.2). To solve (4.3), we shall be employing an algorithm of Chambolle and Pock [16]; see Appendix B.

Note 4.1. Due to difficulties in employing the algorithms of [16] for nonlinear $\mathcal{T}$, we will henceforth take $\mathcal{T}$ to be linear (except in section 7). The framework we describe in this paper is however applicable for general $\mathcal{T}$, so long as one is able to efficiently solve (4.3) for that $\mathcal{T}$.

4.2. The algorithm for (3.3a). We use Algorithm 4.1 to approximately solve (4.3), thereby approximately solving (3.3a).

Algorithm 4.1 Algorithm for solving the linearized (3.3a).

function ReconUpdate($x_n, u_n, y, \mathcal{T}, z_d, f, V, Y, Z, \mathcal{F}, q, \sigma, R, \mathcal{K}, W, \alpha, \beta, \eta_n, K$)
  ▷ Computes x_{n+1} solving (4.3). Note: assumes that $\mathcal{T}$ is linear
  $v_n = \frac{1}{2}(u_n)^2 + \frac{1}{2\varepsilon}W(u_n) + \frac{1}{4}\mu \odot (u_n - f)^2$ ▷ Squaring elementwise
  $z_n = \mathcal{F}(x_n)$
  $w_1 = \texttt{CProd}(z_n, (z_n, z_d), u_n, v_n, \sigma, V, Y, Z, K)$ ▷ See Algorithm A.1
  $w_2 = \texttt{CProd}(z_n, \mathbf{1}_V, u_n, v_n, \sigma, V, Y, Z, K)$
  $g_n = \frac{4\beta}{q\sigma^2}\mathcal{F}^*(w_1 - w_2 \odot z_n)$
  $\tilde{x}_n = x_n - \frac{1}{2}\eta_n^{-1} g_n$
  $\texttt{proxG} : (x, \delta t) \mapsto ((\delta t^{-1} + 2\eta_n)I + 2\alpha\mathcal{T}^*\mathcal{T})^{-1}(2\alpha\mathcal{T}^*(y) + 2\eta_n\tilde{x}_n + x/\delta t)$ ▷ See (B.3)
  $\texttt{proxRS} : (x, \delta t) \mapsto \mathrm{prox}_{\delta t R^*}(x)$ ▷ Recall that $\mathcal{R}(x) = R(\mathcal{K}x)$
  $x_{n+1} = \texttt{PrimalDual}(x_n, 2\eta_n, \mathcal{K}, \mathcal{K}^*, \texttt{proxRS}, \texttt{proxG})$ ▷ See Algorithm B.1
  return x_{n+1}
end function

5. Solving (3.3b). After rescaling by β^{-1}, we rewrite the objective function in (3.3b) as

$$\begin{aligned}
&\mathrm{GL}_\varepsilon(u, \Omega(\mathcal{F}(x_{n+1}), z_d)) + \frac{1}{2}\sum_{i\in V} d_i\mu_i(u_i - f_i)^2 + \frac{1}{2}\frac{2\nu_n}{\beta}\|u|_Y - u_n|_Y\|_{\mathcal{V}}^2 \\
&= \mathrm{GL}_\varepsilon(u, \Omega(\mathcal{F}(x_{n+1}), z_d)) + \frac{1}{2}\sum_{i\in V} d_i\mu_i'(u_i - f_i')^2 \\
&= \mathrm{GL}_{\varepsilon,\mu',f'}(u, \Omega(\mathcal{F}(x_{n+1}), z_d)),
\end{aligned}$$

where $\mu' := \mu + 2\nu_n\beta^{-1}\chi_Y$ and $f' := f + u_n \odot \chi_Y$ (where χ_Y denotes the indicator function of the set Y). We have used that $\mu|_Y = f|_Y = \mathbf{0}$.

5.1. The SDIE scheme for minimizing $\mathrm{GL}_{\varepsilon,\mu',\tilde{f}'}$. Following [11, 13], in order to minimize $\mathrm{GL}_{\varepsilon,\mu',\tilde{f}'}$ we shall consider its *Allen–Cahn gradient flow*:

$$\varepsilon\frac{du}{dt}(t) + \varepsilon\Delta u(t) + \varepsilon M'(u(t) - f') + \frac{1}{2}\mathbf{1} - u(t) = \beta(t), \tag{5.1}$$

where $M' := \mathrm{diag}(\mu')$ and $\beta(t)$ arises from the subdifferential of W; see [11, section 3.3.3] for details. Following [11, section 4], we compute solutions to (5.1) via an SDIE scheme, defined by

$$\left(1 - \frac{\tau}{\varepsilon}\right)u_{m+1} - \mathcal{S}_\tau u_m + \frac{\tau}{2\varepsilon}\mathbf{1} = \frac{\tau}{\varepsilon}\beta_{m+1}, \tag{5.2}$$

for $\tau > 0$ a time step, β_{m+1} a subdifferential term (see [11, Definition 4.1.1]), and $\mathcal{S}_\tau$ the solution operator for fidelity-forced graph diffusion, described by the following theorem.

Theorem 5.1 (see [11, Theorem 3.2.6]). *The fidelity-forced graph diffusion of $u_0 \in \mathcal{V}$ is*

$$\frac{du}{dt}(t) = -\Delta u(t) - M'(u(t) - f'), \qquad u(0) = u_0. \tag{5.3}$$

Algorithm 5.1 Algorithm for solving (3.3b) using the SDIE scheme.

function SegUpdate($u_n, x_{n+1}, z_d, \mathcal{F}, q, \sigma, V, Y, Z, K, \delta, \beta, \nu_n, \tau, \varepsilon, \mu, f, k_s$) ▷ Computes the minimizer of (3.3b) by computing an SDIE sequence as in Appendix C
  $\mu' = \mu + 2\nu_n\beta^{-1}\chi_Y$
  $f' = f + u_n \odot \chi_Y$
  $\omega : ij \mapsto \Omega_{ij}(z, z_d, q, \sigma)$ ▷ Defined as in (3.1)
  $[U_1, \Lambda, U_2] =$ `NyströmQR`$(\omega, V, Z, K/2, K/2)$ ▷ See Algorithm C.1
  $F : x \mapsto (-U_1\Lambda(U_2^T x) - \mu' \odot (x - f'))$
  $\hat{v} =$ `ode_solver`$(F, [0,\tau], \mathbf{0})$ ▷ Solves $\hat{v}'(t) = F(\hat{v})$ on $[0,\tau]$, $\hat{v}(0) = \mathbf{0}$, e.g., as in [34]
  $b = \hat{v}(\tau)$ ▷ $b := F_\tau(\Delta + M')M'f'$
  $\Sigma = I_K - \Lambda$
  $(a_1, a_2) = (\exp(-\tau/k_s(\mu' + \mathbf{1})), \exp(\tau/k_s \mathrm{diag}(\Sigma)) - \mathbf{1}_K)$ ▷ See (C.1)
  $a_3 = \mathrm{sqrt}(a_1)$ ▷ See (C.1)
  $u^0 = \frac{1}{2}\chi_Y + f$ ▷ As an example initial condition
  $m = 0$
  while $\|u^m - u^{m-1}\|_2^2 / \|u^m\|_2^2 \geq \delta$ **do**
    $v = u^m$
    for $r = 1$ to k_s **do**
      $v = a_1 \odot v + a_3 \odot (U_1(a_2 \odot (U_2^T(a_3 \odot v))))$ ▷ Strang formula iteration (C.1)
    end for
    $v = v + b$ ▷ Approximates $v = \mathcal{S}_\tau u^m$
    $V_1 = \{i \in V \mid v_i \in [\tau/2\varepsilon, 1 - \tau/2\varepsilon)\}$
    $V_2 = \{i \in V \mid v_i \geq 1 - \tau/2\varepsilon\}$
    $u^{m+1} = (1 - \tau/\varepsilon)^{-1}(v - \tau/2\varepsilon\mathbf{1}) \odot \chi_{V_1} + \chi_{V_2}$ ▷ Applies the (5.4) thresholding
    $m = m + 1$
  end while
  return u^m
end function

For $t, x \in \mathbb{R}$, let $F_t(x) := (1 - e^{-tx})/x$, and extend F_t to (real) matrix inputs via its Taylor series. Then, for any given $u_0 \in \mathcal{V}$, (5.3) has a unique solution, given by the map

$$u(t) = \mathcal{S}_t u_0 := e^{-t(\Delta + M')}u_0 + F_t(\Delta + M')M'f'.$$

The solution to (5.2) is then given by the following theorem.

Theorem 5.2 (see [11, Theorem 4.2.1]). *For $\tau \in [0, \varepsilon)$, (5.2) has unique solution*

$$(u_{m+1})_i = \begin{cases} 0 & \text{if } (\mathcal{S}_\tau u_m)_i < \frac{\tau}{2\varepsilon}, \\ \frac{1}{2} + \frac{(\mathcal{S}_\tau u_m)_i - 1/2}{1 - \frac{\tau}{\varepsilon}} & \text{if } \frac{\tau}{2\varepsilon} \leq (\mathcal{S}_\tau u_m)_i < 1 - \frac{\tau}{2\varepsilon}, \\ 1 & \text{if } (\mathcal{S}_\tau u_m)_i \geq 1 - \frac{\tau}{2\varepsilon}. \end{cases} \tag{5.4a}$$

Algorithm 6.1 Graph-based joint reconstruction-segmentation algorithm using (3.3).

function JOINTRECSEG($y, \mathcal{T}, z_d, f, V, Y, Z, \mathcal{F}, q, \sigma, R, \mathcal{K}, W, \alpha, \beta, \delta, \eta_n, \nu_n, \tau, \varepsilon, \mu, K, N, k_s$)
  ▷ Computes the first N iterates of (3.3)
  $x_0 = \texttt{cheap\_reconstruction}(y, \mathcal{T})$ ▷ Initial cheap reconstruction
  $u_0 = \texttt{SegUpdate}(f, x_0, z_d, \mathcal{F}, q, \sigma, V, Y, Z, K, \delta, 1, 0, \tau, \varepsilon, \mu, f, k_s)$
    ▷ Initial SDIE segmentation; see Algorithm 5.1
  for $n = 0$ to $N-1$ **do** ▷ The iterations of (3.3)
    $x_{n+1} = \texttt{ReconUpdate}(x_n, u_n, y, \mathcal{T}, z_d, f, V, Y, Z, \mathcal{F}, q, \sigma, R, \mathcal{K}, W, \alpha, \beta, \eta_n, K)$
      ▷ Solves (3.3a); see Algorithm 4.1
    $u_{n+1} = \texttt{SegUpdate}(u_n, x_{n+1}, z_d, \mathcal{F}, q, \sigma, V, Y, Z, K, \delta, \beta, \nu_n, \tau, \varepsilon, \mu, f, k_s)$
      ▷ Solves (3.3b)
  end for
  return $\{x_n\}_{n=0}^N, \{u_n\}_{n=0}^N$
end function

For $\tau = \varepsilon$, (5.2) has solutions

$$(u_{m+1})_i \in \begin{cases} \{1\}, & (\mathcal{S}_\tau u_m)_i > 1/2, \\ [0,1], & (\mathcal{S}_\tau u_m)_i = 1/2, \\ \{0\}, & (\mathcal{S}_\tau u_m)_i < 1/2. \end{cases} \tag{5.4b}$$

Note 5.3. The $\tau = \varepsilon$ special case described by (5.4b) is the *graph MBO scheme*, which has seen widespread use in image segmentation, pioneered by Merkurjev, Kostić, and Bertozzi [34].

We describe how to compute this SDIE scheme in Appendix C.

5.2. The algorithm for (3.3b). We summarize the above as Algorithm 5.1.

6. The full pipeline. We summarize the full pipeline as Algorithm 6.1.

7. Convergence analysis. In this section, we will show that (3.3) converges to critical points of (3.2), using the theory from Attouch et al. [5]. From Note 4.1 we recall that in this section we do not require $\mathcal{T}$ to be linear. We first rewrite (3.2) abstractly:

$$\min_{x \in \mathcal{X}, u \in \mathcal{V}} \mathscr{F}(x) + \mathscr{G}(u, x) + \mathscr{H}(u) =: \mathcal{J}(u, x),$$

where $\mathcal{X} := \mathbb{R}^{N \times \ell}$, $\mathscr{F}(x) := \mathcal{R}(x) + \alpha \|\mathcal{T}(x) - y\|_F^2$,

$$\mathscr{G}(u, x) := \beta \langle -uu^T + \mathbf{1}v^T + v\mathbf{1}^T, \Omega(\mathcal{F}(x), z_d) \rangle_F,$$

$v_i := \frac{1}{2}u_i^2 + \frac{1}{4\varepsilon}u_i(1-u_i) + \frac{1}{4}\mu_i(u_i - f_i)^2$, and $\mathscr{H}(u) := 0$ if $u \in \mathcal{V}_{[0,1]}$ and $\mathscr{H}(u) := \infty$ otherwise.

Note 7.1. Both $\mathscr{F}$ and $\mathscr{H}$ are proper and l.s.c., and $\mathscr{G}$ is C^∞ (indeed, analytic), so [5, Assumption ($\mathscr{H}$)] is satisfied. Furthermore, $\mathscr{G}(u, x) + \mathscr{H}(u) = \mathrm{GL}_{\varepsilon, \mu, f}(u, \Omega(\mathcal{F}(x), z_d))$.

Then (3.3) can be written as

$$x_{n+1} = \operatorname*{argmin}_{x\in\mathcal{X}} \mathscr{F}(x) + \mathscr{G}(u_n, x) + \eta_n \|x - x_n\|_F^2, \tag{7.1a}$$

$$u_{n+1} = \operatorname*{argmin}_{u\in\mathcal{V}} \mathscr{G}(u, x_{n+1}) + \mathscr{H}(u) + \nu_n \|u|_Y - u_n|_Y\|_{\mathcal{V}}^2, \tag{7.1b}$$

and our partially linearized iterative scheme can be written as

$$\tilde{x}_n = x_n - \frac{1}{2}\eta_n^{-1}\nabla_x \mathscr{G}(u_n, x_n), \tag{7.2a}$$

$$x_{n+1} = \operatorname*{argmin}_{x\in\mathcal{X}} \mathscr{F}(x) + \eta_n \|x - \tilde{x}_n\|_F^2, \tag{7.2b}$$

$$u_{n+1} = \operatorname*{argmin}_{u\in\mathcal{V}} \mathscr{G}(u, x_{n+1}) + \mathscr{H}(u) + \nu_n \|u|_Y - u_n|_Y\|_{\mathcal{V}}^2. \tag{7.2c}$$

However, the presence of the *semi*norm $\|\cdot|_Y\|_{\mathcal{V}}$ in (7.1b) is an obstacle to the deployment of the theory from [5]. Hence, we will make an assumption that for all n, $u_n|_Z = f$. In practice, we have observed that this approximately holds, and furthermore the larger the value of μ the more closely this will hold. Thus, defining $\mathcal{V}^f := \{u \in \mathcal{V} \mid u|_Z = f\}$ and $\mathcal{V}^f_{[0,1]} := \{u \in \mathcal{V}_{[0,1]} \mid u|_Z = f\}$, under this assumption (7.1) becomes

$$x_{n+1} = \operatorname*{argmin}_{x\in\mathcal{X}} \mathscr{F}(x) + \mathscr{G}(u_n, x) + \eta_n \|x - x_n\|_F^2, \tag{7.3a}$$

$$u_{n+1} = \operatorname*{argmin}_{u\in\mathcal{V}} \mathscr{G}(u, x_{n+1}) + \mathscr{H}^f(u) + \nu_n \|u - u_n\|_{\mathcal{V}}^2, \tag{7.3b}$$

where $\mathscr{H}^f(u) := 0$ if $u \in \mathcal{V}^f_{[0,1]}$ and $\mathscr{H}^f(u) := \infty$ otherwise. Let $\mathcal{J}^f(u,x) := \mathscr{F}(x) + \mathscr{G}(u,x) + \mathscr{H}^f(u)$, which is equal to $\mathcal{J}(u,x)$ if $u \in \mathcal{V}^f$. We begin by making some key definitions.[5]

Definition 7.2 (Kurdyka–Łojasiewicz property). *A proper l.s.c. function $g : \mathbb{R}^n \to (-\infty, \infty]$ has the Kurdyka–Łojasiewicz property at $\bar{z} \in \operatorname{dom} \partial g$[6] if there exist $\eta \in (0, \infty]$, a neighborhood U of $\bar{z}$, and a continuous concave function $\varphi : [0, \eta) \to [0, \infty)$ such that*

- *φ is C^1 with $\varphi(0) = 0$ and $\varphi' > 0$ on $(0, \eta)$, and*
- *for all $z \in U$ such that $g(\bar{z}) < g(z) < g(\bar{z}) + \eta$, the Kurdyka–Łojasiewicz inequality holds:*

$$\varphi'(g(z) - g(\bar{z}))\operatorname{dist}(\mathbf{0}, \partial g(z)) \geq 1.$$

If $\varphi(s) := cs^{1-\theta}$ is a valid concave function for the above with $c > 0$ and $\theta \in [0, 1)$, then we will say that g has the Kurdyka–Łojasiewicz property with exponent θ at $\bar{z}$.

Definition 7.3 (semianalyticity and subanalyticity). *Following, e.g., Łojasiewicz [32], we define $A \subseteq \mathbb{R}^n$ to be a semianalytic set if for all $z^* \in \mathbb{R}^n$ there exists a neighborhood U containing z^* and a finite collection of analytic functions (a_{ij}, b_{ij}) such that*

$$A \cap U = \bigcup_i \bigcap_j \{z \in U \mid a_{ij}(z) = 0 \text{ and } b_{ij}(z) > 0\}.$$

[5] With respect to the question of what these definitions are for, we ask the reader to bear with us, with the hope that within a page or two their purpose will become clearer.

[6] We denote by $\operatorname{dom} g$ the set of z such that $g(z) < \infty$ and by $\operatorname{dom} \partial g$ the set of $z \in \operatorname{dom} g$ such that the (limiting) subdifferential of g at z, $\partial g(z)$ (defined in [5, Definition 2.1]), is nonempty.

Following Hironaka [28], we define A to be a subanalytic set if for all $z^* \in \mathbb{R}^n$ there exists a neighborhood U containing z^*, $m \in \mathbb{N}$, and a bounded semianalytic set $B \subset \mathbb{R}^{n+m}$ such that

$$A \cap U = \{z \in \mathbb{R}^n \mid \exists y \in \mathbb{R}^m \text{ such that } (z,y) \in B\}.$$

We define $g : \mathbb{R}^n \to (-\infty, \infty]$ to be a semianalytic function if its graph $\operatorname{Gr} g := \{(z,z') \in \mathbb{R}^n \times \mathbb{R} \mid z' = g(z)\}$ is a semianalytic set and define g to be a subanalytic function if its graph is a subanalytic set. Both of these collections of sets are closed under elementary set operations.

We collect some key results regarding these definitions in the following lemma.

Lemma 7.4.

i. *If g is proper and l.s.c., and $\bar{z} \in \operatorname{dom} g$ with $\mathbf{0} \notin \partial g(\bar{z})$, then for all $\theta \in [0,1)$, g has the Kurdyka–Łojasiewicz property with exponent θ at $\bar{z}$.*
ii. *If g is proper and subanalytic, $\operatorname{dom} g$ is closed, and g is continuous on its domain, then for all $\bar{z} \in \operatorname{dom} g$ with $\mathbf{0} \in \partial g(\bar{z})$, there exists $\theta \in [0,1)$ such that g has the Kurdyka–Łojasiewicz property with exponent θ at $\bar{z}$.*
iii. *If $g : \mathbb{R}^n \to (-\infty, \infty]$ is subanalytic and $h : \mathbb{R}^n \to \mathbb{R}$ is analytic, then $g+h$ is subanalytic.*
iv. *If $g : \mathcal{V}^f \times \mathcal{X} \to (-\infty, \infty]$ is subanalytic, then $h : (u,x) \mapsto g(u,x) + \mathscr{H}^f(u)$ is subanalytic.*

Proof.

i. Proved in Li and Pong [30, Lemma 2.1].
ii. Proved in Bolte, Daniilidis, and Lewis [8, Theorem 3.1].
iii. Fix $(z^*, w^*) \in \mathbb{R}^n \times \mathbb{R}$. Since g is subanalytic, there exists U containing $(z^*, w^* - h(z^*))$ and a bounded semianalytic set $B \subset \mathbb{R}^{n+1+m}$ such that

$$\operatorname{Gr} g \cap U = \{(z,w) \in \mathbb{R}^n \times \mathbb{R} \mid \exists y \in \mathbb{R}^m \text{ such that } (z,w,y) \in B\}.$$

Since h is continuous, there exists a neighborhood V containing (z^*, w^*) such that for all $(z,w) \in \operatorname{Gr}(g+h) \cap V$, $(z, w - h(z)) \in \operatorname{Gr} g \cap U$, and hence $(z, w - h(z), y) \in B$ for some $y \in \mathbb{R}^m$. Let $B' := \{(z, w + h(z), y) \mid (z,w,y) \in B\}$. Since h is analytic, B' is a bounded semianalytic set, and for all $(z,w) \in \operatorname{Gr}(g+h) \cap V$, there exists $y \in \mathbb{R}^m$ such that $(z,w,y) \in B'$. It follows that $g+h$ is subanalytic.
iv. $\operatorname{Gr} h = \{(u,x,y) \mid u \in \mathcal{V}^f_{[0,1]} \text{ and } (u,x,y) \in \operatorname{Gr} g\} = (\mathcal{V}^f_{[0,1]} \times \mathcal{X} \times \mathbb{R}) \cap \operatorname{Gr} g$. It is simple to check that $\mathcal{V}^f_{[0,1]}$, $\mathcal{X}$, and $\mathbb{R}$ are semianalytic, and hence subanalytic, and therefore if $\operatorname{Gr} g$ is subanalytic, then so is $\operatorname{Gr} h$.

■

Assumption 7.5. Suppose that $\mathcal{R}$ is subanalytic, continuous on its domain, and bounded below, and $\operatorname{dom} \mathcal{R}$ is closed. Suppose also that $\mathcal{T}$ is analytic and that $\mathscr{F}(x) \to \infty$ as $\|x\|_F \to \infty$.

Note 7.6. Examples of $\mathcal{R}$ satisfying this assumption are $\mathcal{R}(x) := \|Ax\|_1$ (commonly used in compressed sensing; see Adcock and Hansen [1]), where A is any matrix, and $\mathcal{R}$ given by a feedforward neural network with a ReLU activation function (commonly used as regularizers; see, e.g., Arridge et al. [4]); see Theorem D.1 for proofs. Examples of $\mathscr{F}$ satisfying the assumption are when $\mathcal{T}$ is an invertible linear map or $\mathcal{R}$ is coercive.

Theorem 7.7 (see [5, Lemma 3.1]). *If Assumption 7.5 holds and, for some $0 < a < b$ and all n, $\eta_n, \nu_n \in (a,b)$, then $(u_n, x_n)_{n \in \mathbb{N}}$ solving (7.3) are well-defined, and furthermore,*

i. $\mathcal{J}^f(u_{n+1},x_{n+1}) \le \mathcal{J}^f(u_n,x_n)$ *with equality if and only if* $x_{n+1}=x_n$ *and* $u_{n+1}=u_n$;
ii. $\lim_{n\to\infty} \|x_{n+1}-x_n\|_F+\|u_{n+1}-u_n\|_{\mathcal{V}}=0$ *(indeed, the differences are square-summable);*
iii. *for all bounded subsequences* $(u_{n'},x_{n'})$, $\operatorname{dist}(\mathbf{0},\partial\mathcal{J}^f(u_{n'},x_{n'}))\to 0$.

Proof. Note that, by Assumption 7.5, $\mathcal{J}^f(u,x)$ is bounded below and that for all $x\in\mathcal{X}$, $\mathcal{J}^f(\cdot,x)$ is proper. Hence given the assumption on η_n and ν_n, [5, Assumption $(\mathscr{H}_1)$] is satisfied, and therefore the result follows by [5, Lemma 3.1]. ∎

Lemma 7.8. *Let Assumption 7.5 hold. Then for all* $\bar{u}\in\mathcal{V}^f_{[0,1]}$ *and for all* $\bar{x}\in\operatorname{dom}\mathcal{R}$, *there exists* $\theta\in[0,1)$ *such that* $\mathcal{J}^f$ *has the Kurdyka–Łojasiewicz property with exponent* θ *at* $(\bar{u},\bar{x})$.

Proof. Note first that $\operatorname{dom}\mathcal{J}^f=\mathcal{V}^f_{[0,1]}\times\operatorname{dom}\mathcal{R}$, which is therefore closed by the assumption. Next, note that by the assumption and the continuity of $\mathscr{G}$ and $\mathscr{H}^f$ (in the latter case, on its domain), $\mathcal{J}^f$ is continuous on its domain. Finally, since $\mathscr{G}$ is analytic, $\|\mathcal{T}(x)-y\|_F^2$ is analytic, and $\mathcal{R}$ is subanalytic, it follows by Lemma 7.4(iii)–(iv) that $\mathcal{J}^f$ is subanalytic.

Let $\bar{u}\in\mathcal{V}^f_{[0,1]}$ and $\bar{x}\in\operatorname{dom}\mathcal{R}$. If $\mathbf{0}\in\partial\mathcal{J}^f(\bar{u},\bar{x})$, then by the above Lemma 7.4(ii) applies. Thus there exists $\theta\in[0,1)$ such that $\mathcal{J}^f$ has the Kurdyka–Łojasiewicz property with exponent θ at $(\bar{u},\bar{x})$. If instead $\mathbf{0}\notin\partial\mathcal{J}^f(\bar{u},\bar{x})$, then by Lemma 7.4(i)—as $\mathcal{J}^f$ is proper and l.s.c.— for all $\theta\in[0,1)$, $\mathcal{J}^f$ has the Kurdyka–Łojasiewicz property with exponent θ at $(\bar{u},\bar{x})$. ∎

Lemma 7.9. *Suppose that, for some* $0<a<b$ *and all* n, $\eta_n,\nu_n\in(a,b)$, *and that Assumption 7.5 holds. Then for all* $u_0\in\mathcal{V}^f_{[0,1]}$ *and for all* $x_0\in\operatorname{dom}\mathcal{R}$, *if* $(u_n,x_n)_{n\in\mathbb{N}}$ *is defined from* (u_0,x_0) *by (7.3), then* $\|(u_n,x_n)\|:=\|u_n\|_{\mathcal{V}}+\|x_n\|_F$ *is bounded.*

Proof. As $u_n\in\mathcal{V}^f_{[0,1]}$ for all n, it suffices to show that $\|x_n\|_F$ is bounded. For all $u\in\mathcal{V}$ and for all $x\in\mathcal{X}$, $\mathscr{G}(u,x)+\mathscr{H}^f(u)\ge 0$, and, by Theorem 7.7(i), $\mathcal{J}^f(u_n,x_n)\le\mathcal{J}^f(u_0,x_0)$ for all n. By Assumption 7.5, $\mathscr{F}(x)\to\infty$ as $\|x\|_F\to\infty$; hence $\|x_n\|_F$ is bounded. ∎

Theorem 7.10 (see [5, Theorems 3.2, 3.3, and 3.4]). *Suppose that for some* $0<a<b$ *and all* n, $\eta_n,\nu_n\in(a,b)$; $(u_n,x_n)_{n\in\mathbb{N}}$ *is defined from* (u_0,x_0) *by (7.3), and Assumption 7.5 holds. Then for all* $u_0\in\mathcal{V}^f_{[0,1]}$ *and all* $x_0\in\operatorname{dom}\mathcal{R}$, (u_n,x_n) *converges to a critical point of* $\mathcal{J}^f$.

Furthermore, suppose that $\bar{u}\in\mathcal{V}^f_{[0,1]}$, $\bar{x}\in\operatorname{dom}\mathcal{R}$, *and* $(\bar{u},\bar{x})$ *is a global minimum of* $\mathcal{J}^f$. *Then there exists a neighborhood* U *containing* $(\bar{u},\bar{x})$ *and* $\eta>0$ *such that if* $(u_0,x_0)\in U$ *and* $\min\mathcal{J}^f<\mathcal{J}^f(u_0,x_0)<\min\mathcal{J}^f+\eta$, *then* $(u_n,x_n)\to(u^*,x^*)$, *a global minimizer of* $\mathcal{J}^f$.

Finally, let $(u_n,x_n)\to(u_\infty,x_\infty)$ *where* $u_\infty\in\mathcal{V}^f_{[0,1]}$ *and* $x_\infty\in\operatorname{dom}\mathcal{R}$. *If* $\theta\in[0,1)$ *is the Kurdyka–Łojasiewicz exponent of* $\mathcal{J}^f$ *at* (u_∞,x_∞) *(which exists by Lemma 7.8), then*

i. *if* $\theta=0$, *then* (u_n,x_n) *converges in finitely many steps;*
ii. *if* $\theta\in(0,\frac{1}{2}]$, *then there exist* $c>0$ *and* $\zeta\in[0,1)$ *such that* $\|(u_n,x_n)-(u_\infty,x_\infty)\|\le c\zeta^n$;
iii. *if* $\theta\in(\frac{1}{2},1)$, *then there exists* $c>0$ *such that* $\|(u_n,x_n)-(u_\infty,x_\infty)\|\le cn^{-(1-\theta)/(2\theta-1)}$.

Proof. Given the suppositions [5, Assumptions $(\mathscr{H})$ and $(\mathscr{H}_1)$] hold, by Lemma 7.8, for all $\bar{u}\in\mathcal{V}^f_{[0,1]}$ and $\bar{x}\in\operatorname{dom}\mathcal{R}$ there exists $\theta\in[0,1)$ such that $\mathcal{J}^f$ has the Kurdyka–Łojasiewicz property with exponent θ at $(\bar{u},\bar{x})$, and by Lemma 7.9, $\|(u_n,x_n)\|$ does not tend to infinity. Therefore the results follow immediately from [5, Theorems 3.2, 3.3, and 3.4], respectively. ∎

Note 7.11. The above convergence results concerned (7.3), in which everything is solved without linearization. In Bolte, Sabach, and Teboulle [9], similar convergence results as in [5]

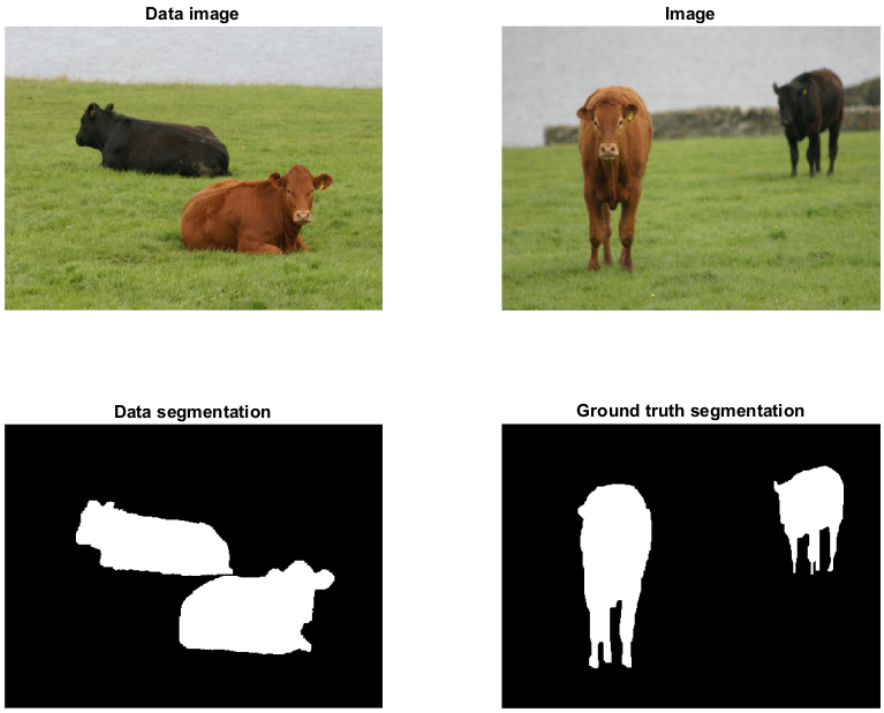

Figure 1. *Two cows: the reference data image, the image to be segmented, the reference f (which is a segmentation of the reference data image), and the ground truth segmentation associated to Example 8.1. Both segmentations were drawn by hand by the authors.*

were proved for fully linearized alternating schemes for energies with the Kurdyka–Łojasiewicz property. It is beyond the scope of this work to extend such results to (7.2), which is only partially linearized, but it is the authors' belief that such an extension is likely to be straightforward.

8. Applications.

We will test our scheme on distorted versions of the "two cows" images familiar from [7, 11, 13, 34].[7]

Example 8.1 (two cows). We take the image in the top left of Figure 1 as the reference data Z and the segmentation in the bottom left as the reference labels f, which separate the cows from the background. The image x^* to be segmented is the top-right image of Figure 1, with the ground truth segmentation in the bottom right. Both images are RGB images of size 480×640 pixels. Figure 2 shows the result of segmenting x^* via Algorithm 5.1, as in [11, section 5.3.3].

8.1. Computational set-up.

All programming was done in MATLAB R2020b with the relevant toolboxes Computer Vision Toolbox Version 10.1, Image Processing Toolbox Version 11.4, Signal Processing Toolbox Version 8.7, and Deep Learning Toolbox Version 14.3. In this section, all functions in `typewriter font` will refer to built-in MATLAB functions.

[7] These images are taken from the Microsoft Research Cambridge Object Recognition Image Database, available at https://www.microsoft.com/en-us/research/project/image-understanding/, accessed 4 March 2022.

Figure 2. *Image masked with MBO segmentation for Example 8.1, reproduced from* [11, Figure 5.9(d)]. *The segmentation accuracy is* 98.4622%.

Timings were taken with implementations executed serially on a machine with an Intel Core i7-9800X @ 3.80 GHz (16 cores) CPU and 32 GB RAM.

8.2. The feature map and its adjoint. In the following applications, we will define $\mathcal{F}$ (and likewise $\mathcal{F}_d$ *mutatis mutandis*) as follows. Recall that $x : Y \to \mathbb{R}^\ell$. For each pixel $i \in Y$, suppose we have a map $\mathcal{N}_i : \{1,\dots,k\} \to Y$ which defines the k "image-neighbors" of i in Y and we likewise have a kernel $\mathscr{K} : \{1,\dots,k\} \to \mathbb{R}$. Then for each channel $s \in \{1,\dots,\ell\}$ of x, $i \in Y$, and $p \in \{1,\dots,k\}$, we define $(\mathcal{Z}(x^s))_{ip} := \mathscr{K}(p)x^s_{\mathcal{N}_i(p)}$, and we define $z = \mathcal{F}(x) \in \mathbb{R}^{N\times k\ell}$ by $z := \begin{pmatrix}\mathcal{Z}(x^1) & \mathcal{Z}(x^2) & \dots & \mathcal{Z}(x^\ell)\end{pmatrix}$. We now derive $\mathcal{F}^* : \mathbb{R}^{N\times k\ell} \to \mathbb{R}^{N\times\ell}$, the adjoint of $\mathcal{F}$ with respect to $\langle\cdot,\cdot\rangle_F$. Write $w = \begin{pmatrix}w^1 & w^2 & \dots & w^\ell\end{pmatrix}$, where $w^s \in \mathbb{R}^{N\times k}$. Then since

$$\langle \mathcal{Z}(x^s), w^s\rangle_F = \sum_{i\in Y}\sum_{p=1}^{k} x^s_{\mathcal{N}_i(p)}\mathscr{K}(p)w^s_{ip} = \sum_{j\in Y} x^s_j \left(\sum_{\{(i,p)|\mathcal{N}_i(p)=j\}} \mathscr{K}(p)w^s_{ip}\right),$$

it follows that $\mathcal{Z}$ has adjoint $\mathcal{Z}^* : \mathbb{R}^{N\times k} \to \mathbb{R}^N$ given by

$$(\mathcal{Z}^*(w^s))_j = \sum_{\{(i,p)|\mathcal{N}_i(p)=j\}} \mathscr{K}(p)w^s_{ip}.$$

Finally, by construction, $\mathcal{F}^*(w) = \begin{pmatrix}\mathcal{Z}^*(w^1) & \mathcal{Z}^*(w^2) & \dots & \mathcal{Z}^*(w^\ell)\end{pmatrix}$.

For this section we will take $k = 9$, the image-neighbors of pixel i to be the 3×3 square centered on i (with replication padding at the boundaries), and $\mathscr{K}$ to be 9 multiplied by a 3×3 Gaussian kernel with standard deviation 1, centered on the center of that square, computed via `fspecial('gaussian')`. As the images are RGB, $\ell = 3$, and $z \in \mathbb{R}^{N\times 27}$.

8.3. Measures of reconstruction and segmentation accuracy. In the following examples, we will measure the accuracy of a reconstruction x relative to a ground truth of x^* by the peak signal-to-noise ratio (PSNR) (defined as in [1, (2.6)]).

Figure 3. *Typical observed data y for Example 8.2.*

We will measure the accuracy of a segmentation u relative to a ground truth u^* by its Dice score, defined as $2TP/(2TP+FP+FN)$, where TP is the number of pixels which are true positives ("positives" will here be pixels identified as "cow"), FP of false positives, and FN of false negatives. That is (since "positive" pixels will be given label "1"):

$$\text{Dice score of } u \text{ relative to } u^* := \frac{2u \cdot u^*}{2u \cdot u^* + u \cdot (1-u^*) + (1-u) \cdot u^*}.$$

8.4. Denoising the "two cows" image. As our first application, we will test our method on a highly noised version of Example 8.1.

Example 8.2 (noised two cows). Let Z, f, and x^* (the true image that is to be reconstructed and segmented) be as in Example 8.1. Let the observed data y (see Figure 3) be x^* plus Gaussian noise with mean 0 and standard deviation 1, created via `imnoise`. Thus, $\mathcal{T}$ is the identity. This is a very high noise level, with a typical PSNR of y relative to x^* of 6.2 dB.

8.4.1. Parameters and initialization. Unless otherwise stated, all parameter values were chosen through manual trial and error. We let $\alpha = 0.75$, $\beta = 10^{-5}$, $\eta_n = 0.1$, $\nu_n = 10^{-6}$, and $\sigma = 3$. For the SDIE scheme for (3.3b), we choose $\tau = \varepsilon = 0.00285$, $\mu = 50\chi_Z$, $k_s = 5$, and stopping condition parameter $\delta = 10^{-10}$. For the Nyström-QR scheme (see Appendix C) we take $K = 100$. As regularizer $\mathcal{R}$ we use Huber-TV [29]:[8]

$$\mathcal{R}(x) = R(\mathcal{K}x) = 10 \sum_{i \in Y} \begin{cases} \|(\nabla x)_i\|_2 - 0.005 & \text{if } \|(\nabla x)_i\|_2 > 0.01, \\ \|(\nabla x)_i\|_2^2 / 0.02 & \text{if } \|(\nabla x)_i\|_2 \le 0.01, \end{cases}$$

[8] This choice of regularizer from among the various commonly used candidates is somewhat arbitrary; we have not yet explored the impact of the choice of regularizer on the behavior of this scheme.

where $\mathcal{K}x := \nabla x$, with $(\nabla x)_i = (x_{i\downarrow} - x_i, x_{i\rightarrow} - x_i)^T$, where $x_{i\downarrow}$ is the x-value at the pixel directly below pixel i in the image and $x_{i\rightarrow}$ the x-value at the pixel directly to the right of pixel i (with replication padding at boundaries).

The initial reconstruction x_0 was computed via a standard TV-based (i.e., Rudin–Osher–Fatemi [44]) denoising, with fidelity term 1.05. That is,

$$x_0 = \underset{x\in\mathbb{R}^{N\times\ell}}{\operatorname{argmin}} \operatorname{TV}(x) + 1.05\|x - y\|_F^2,$$

where $\operatorname{TV}(x) := \sum_{i\in Y} \|(\nabla x)_i\|_2$; this is solved using the split Bregman method from Getreuer [25], using code from https://getreuer.info/posts/tvreg/index.html (accessed 10 August 2022), with 50 split Bregman iterations and a tolerance of 10^{-5}. The initial segmentation u_0 of x_0 was computed via the SDIE scheme with the above parameters and initial state $u^0 = 0.47\chi_Y + f$.

8.4.2. Example results. Before discussing the choice of parameters, timings, and accuracy in more detail, we present an example run of the reconstruction-segmentation method for the noisy data and set-up described in Example 8.2 and subsection 8.4.1, respectively.

We show the results of our denoising-segmentation in Figure 4. In the figure we see that the reconstruction is not particularly good by itself (as one would expect given the very high noise level), but the segmentation is very good (compared to the baseline achieved by other methods; see subsection 8.4.4). Importantly, the reconstruction increases the contrast between cows and background, which aids the segmentation.

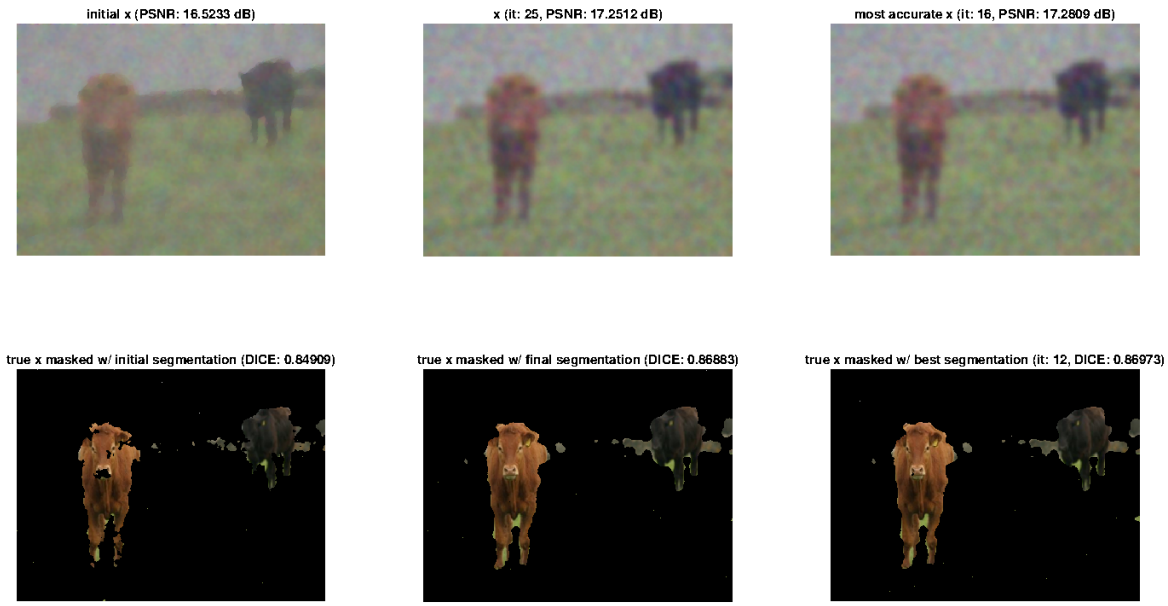

Figure 4. *Reconstruction and segmentation of Example 8.2 from a single run of the reconstruction-segmentation algorithm using the parameters from subsection 8.4.1. Each column contains the reconstruction (top) and segmentation (bottom). We show (left to right) initialization, the final iteration (= 25), and the output with the best scores over the run.*

8.4.3. Parameters, accuracy, and timings. We now study the denoising-segmentation of Example 8.2 more quantitatively. We consider timings of the total runs but also of the most important steps, as well as reconstruction and segmentation accuracy. In order to understand the dependence on the algorithm's parameters we consider four settings: the setting from subsection 8.4.1, a change in the segmentation parameters compared to subsection 8.4.1 ($K = 70$ (decrease), $\varepsilon = \tau = 0.003$ (increase)) (as used in [13]), a doubling of the segmentation weight ($\beta = 2 \cdot 10^{-5}$), and a decrease in the reconstruction weight ($\alpha = 0.5$). We list the results of these settings in Table 1 and present them in Figures 5 to 9.

Table 1

Results averaged over 50 runs for different parameter settings for Example 8.2. Values are given as "mean(standard deviation)." Total timing is for 25 iterations of the algorithm. Details of the changes are described in the main text.

	Changes to subsection 8.4.1	$\emptyset$	K, ε, τ	β	α
	Figure	6	7	8	9
Accuracy	Dice score [%]	85.84(1.68)	85.46(1.28)	87.14(1.45)	86.51(1.59)
	PSNR [dB]	17.19(0.03)	17.22(0.03)	17.74(0.05)	17.54(0.04)
Timings [s]	total	150.19(1.52)	128.86(1.23)	154.00(1.81)	153.80(1.25)
	initialization	1.49(0.03)	1.52(0.03)	1.51(0.08)	1.49(0.03)
	reconstruction	3.43(0.08)	3.31(0.07)	3.57(0.08)	3.56(0.07)
	segmentation	2.52(0.03)	1.79(0.03)	2.53(0.04)	2.53(0.03)

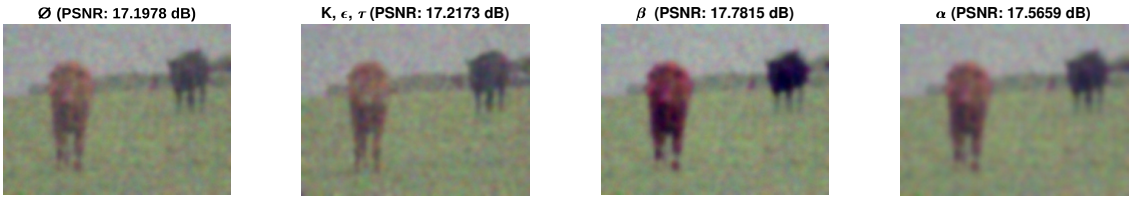

Figure 5. *Example reconstruction results obtained from the denoising-reconstruction method for the different parameter settings for Example 8.2 described in subsection 8.4.3.*

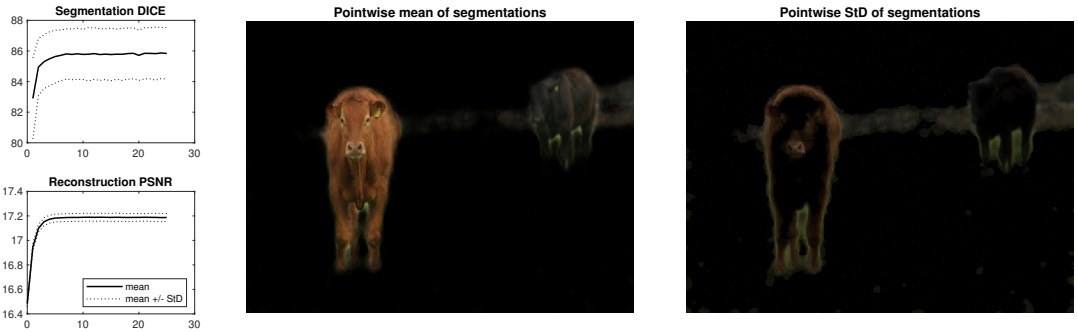

Figure 6. *Results averaged over 50 runs of the reconstruction-segmentation scheme in the denoising case. The line plots (left) show the evolution of the Dice score (%) and PSNR (dB) over the 25 iterations of the algorithm. The images (center and right) show mean and standard deviation of the segmentation at $t = 25$ which are used as weights for the ground truth images. The setting is that of subsection 8.4.1.*

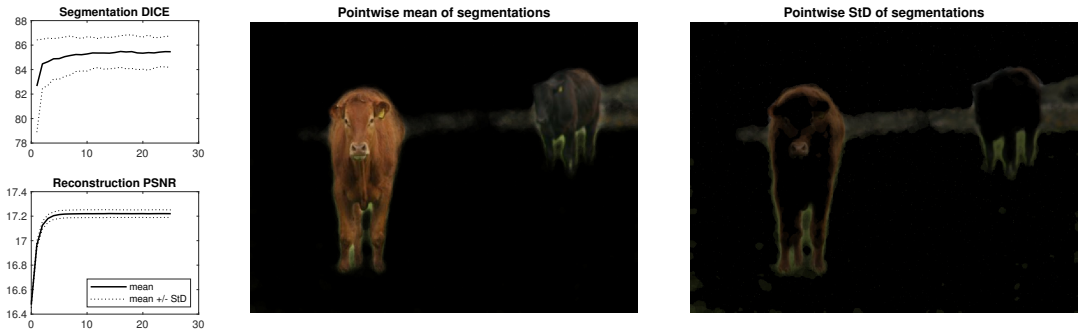

Figure 7. *Results averaged over* 50 *runs of the reconstruction-segmentation scheme in the denoising case. The line plots (left) show the evolution of the Dice score (%) and PSNR (dB) over the* 25 *iterations of the algorithm. The images (center and right) show mean and standard deviation of the segmentation at* $t = 25$ *which are used as weights for the ground truth images. The setting is that of subsection* 8.4.1*, subject to the changes* $K = 70$*,* $\tau = \varepsilon = 0.003$*.*

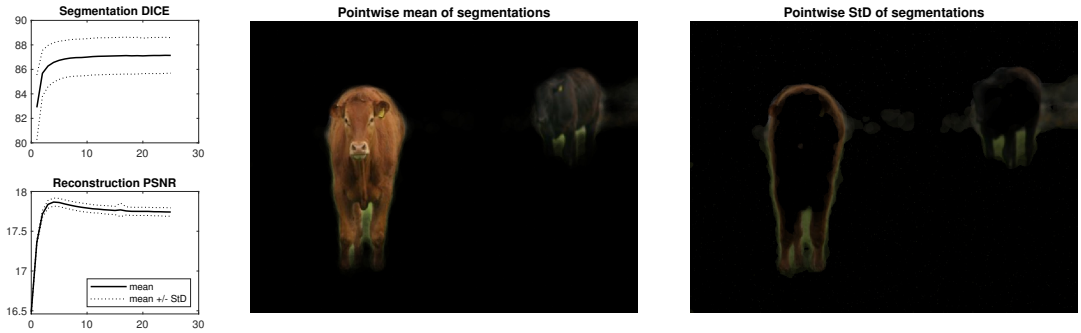

Figure 8. *Results averaged over* 50 *runs of the reconstruction-segmentation scheme in the denoising case. The line plots (left) show the evolution of the Dice score (%) and PSNR (dB) over the* 25 *iterations of the algorithm. The images (center and right) show mean and standard deviation of the segmentation at* $t = 25$ *which are used as weights for the ground truth images. The setting is that of subsection* (8.4.1)*, subject to the change* $\beta = 2 \cdot 10^{-5}$*.*

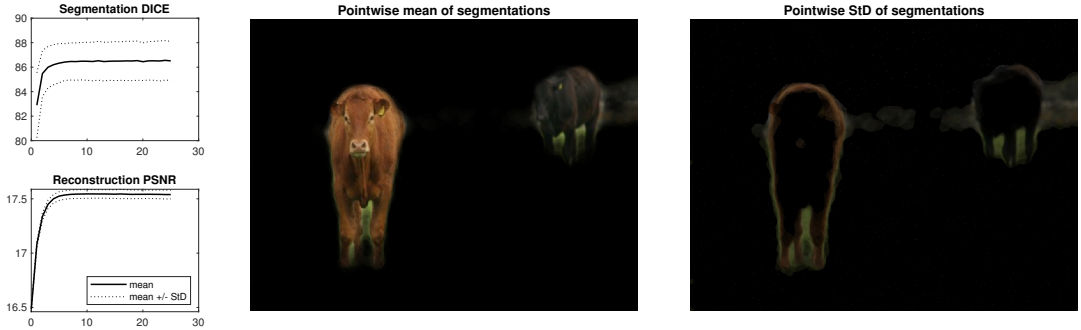

Figure 9. *Results averaged over* 50 *runs of the reconstruction-segmentation scheme in the denoising case. The line plots (left) show the evolution of the Dice score (%) and PSNR (dB) over the iterations of the algorithm. The images (center and right) show mean and standard deviation of the segmentation at* $t = 25$ *which are used as weights for the ground truth images. The setting is that of subsection* 8.4.1*, subject to the change* $\alpha = 0.5$*.*

In all of the settings, we obtain a quick and accurate segmentation. The PSNR of the reconstruction is similar in all settings, suggesting that this low accuracy is intrinsic to this noise level. The initial segmentation is very fast, but in all settings is not as accurate as the joint reconstruction-segmentation. Of the results, some observations are particularly remarkable:

- For the pure MBO segmentation discussed in [13], the Nyström rank K and $\tau = \varepsilon$ were chosen as in the second setting presented here (Figure 7). While this choice was optimal for pure segmentation, and the smaller rank leads to a smaller computational cost, the obtained Dice scores are worse than all the other settings.
- The larger β value (Figure 8; see also Figure 5, third from left) puts more weight on the Ginzburg–Landau energy, leading to the best Dice average, but (unexpectedly) also the highest PSNR average.
- The Dice values have relatively large standard deviations. This is likely caused by the very high noise level, but also by the inherent randomness of the Nyström extension.
- We attain near peak Dice and PSNR values in fewer than 10 iterations in all cases.

8.4.4. Comparison to sequential reconstruction-segmentation. Finally, we compare the accuracy of our joint reconstruction-segmentation approach to the more traditional sequential approach. That is, we will first denoise the image of cows, and then segment it. Segmentations will be performed using the graph MBO scheme with the same set-up as in subsection 8.4.1.

One example of this sequential approach is our initialization process (i.e., TV denoising followed by MBO segmentation), which we observed gave worse PSNR and Dice scores than the joint reconstruction-segmentation output. However, this is perhaps unfair because that initialization was specifically chosen because it is quick (around $1.5s$; see Table 1), while the whole joint reconstruction-segmentation scheme takes about 2.5 minutes to run (although as was mentioned above, the scheme achieves near peak accuracy in closer to 1 minute).

For a potentially fairer comparison, we consider three more sophisticated denoisers: denoising with total generalised variation (TGV) regularization [10] through code by Condat [18] and downloaded from https://lcondat.github.io/software.html (accessed 21 May 2022); the block-matching and 3D filtering (BM3D) algorithm [38] via code by Mäkinen, Azzari, and Foi and downloaded from https://webpages.tuni.fi/foi/GCF-BM3D/ (accessed 18 May 2022); and the built-in MATLAB pretrained denoising convolutional neural network (CNN) (defined in Zhang et al. [52]) loaded via `denoisingNetwork('DnCNN')` and implemented via `denoiseImage`. Remarkably, however, we observe that the BM3D denoiser barely outperforms TV, the TGV denoiser performs slightly worse than TV, and the CNN denoiser performs much worse;[9] see Figure 10. Comparing with the PSNR scores in Table 1, we note that the TV, TGV, BM3D, and CNN denoisers are all outperformed by the reconstruction from our scheme.

The mean Dice score $\pm$ standard deviation (over 50 trials) for MBO segmentations of the TGV denoised image is 0.7138 ± 0.0578, of the BM3D denoised image is 0.8180 ± 0.0130, and of the CNN denoised image is 0.5167 ± 0.0094. Typical segmentations are shown in Figure 11. Similarly to the reconstructions, all of the segmentations are worse or considerably worse than the segmentations obtained from our scheme. However, the sequential denoising-segmentations are notably faster than our scheme; we report the timings in Table 2.

[9] We expect this poor performance to be because the network is pretrained; this performance is somewhat concerning and may not be representative of the general capabilities of CNNs for this denoising task.

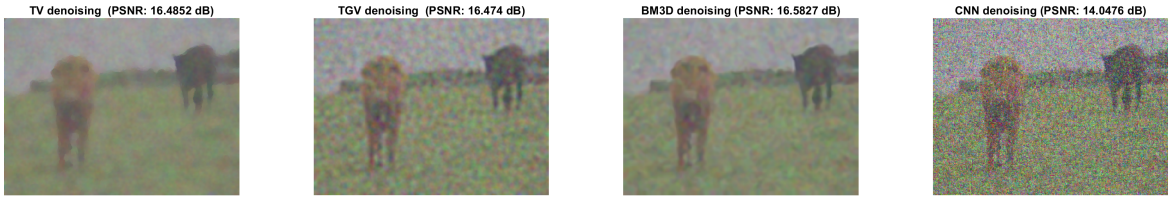

Figure 10. *Typical denoised output for the TV-based, TGV-based, BM3D, and CNN denoisers.*

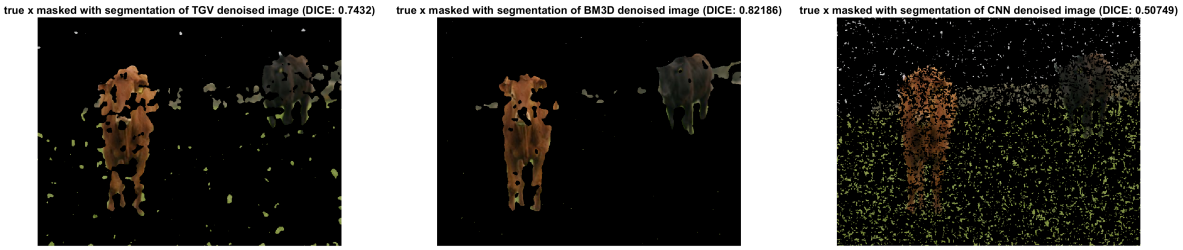

Figure 11. *The ground truth image masked with typical MBO segmentations of the TGV, BM3D, and CNN denoised images.*

Table 2

Timings averaged over 50 *runs for the three sequential denoising-segmentation methods. Values are given as "mean(standard deviation)."*

	TGV	BM3D	CNN
Reconstruction time [s]	8.51(0.05)	14.67(0.14)	0.19(0.14)
Segmentation time [s]	5.99(0.89)	5.84(0.04)	5.87(0.03)
Total time [s]	14.50(0.89)	20.51(0.14)	6.06(0.13)

8.5. Deblurring the "two cows". For our next example, now with $\mathcal{T}$ not equal to the identity map, we consider a blurred version of Example 8.1.

Example 8.3 (blurred two cows). Let Z, f, and the true image x^* (that is to be reconstructed and segmented) be as in Example 8.1. Let the observed data y (see Figure 12) be a horizontal motion blurring of x^* of distance 75 pixels (with symmetric padding at the boundary) created via `imfilter`, plus Gaussian noise with mean 0 and standard deviation 10^{-1} created via `imnoise`. This y has a typical PSNR relative to x^* of around 17.9 dB.

8.5.1. $\mathcal{T}$, its adjoint, and (B.3). In Example 8.3, the forward model $\mathcal{T}$ works by convolving x with a motion blur filter $\mathcal{M}$ (computed using `fspecial('motion')`). The adjoint $\mathcal{T}^*$ therefore corresponds to convolution with a filter $\mathcal{M}'$ defined by reflecting $\mathcal{M}$ in both axes. In the case of motion blur, $\mathcal{M} = \mathcal{M}'$, so $\mathcal{T}$ is self-adjoint. Furthermore, $\mathcal{M}$ has nonnegative values, and so $\mathcal{T}$ is represented by a nonnegative (symmetric) matrix.

In order to solve (4.3), recall that we need to compute (B.3). That is, we need to be able to compute solutions to equations of the form $((2\eta_n + \delta t^{-1})I + 2\alpha\mathcal{T}^2)x = z$. We will do this via a fixed-point iteration. That is, we let $x^0 := z$ and iterate

Figure 12. *Typical observed data y for Example 8.3.*

$$x^{m+1} := \frac{1}{2\eta_n + \delta t^{-1}} z - \frac{2\alpha}{2\eta_n + \delta t^{-1}} \mathcal{T}^2 x^m. \tag{8.1}$$

By the Banach fixed-point theorem, if $\zeta := 2\alpha(2\eta_n + \delta t^{-1})^{-1}\|\mathcal{T}^2\| < 1$, then $\|x - x^m\|_F = \mathcal{O}(\zeta^m)$ as $m \to \infty$. As $\|\mathcal{T}\| = 1$,[10] it suffices to take $\alpha < \eta_n + \frac{1}{2}\delta t^{-1}$ for convergence of (8.1).

8.5.2. Parameters and initialization. We take $\alpha = \eta_n = 2$, $\tau = \varepsilon = 0.002$, $K = 200$, and parameters β, ν_n, σ, μ, k_s, and δ as in subsection 8.4.1. We take $\mathcal{R}$ as in subsection 8.4.1 except that we change the multiplicative factor from 10 to 1. Another change is the number of iterations of the algorithm: while we iterate for 25 steps in the denoising problems, in preliminary runs (not reported) we noticed in the deblurring case that 15 steps are sufficient.

The initial reconstruction x_0 is computed via TV deblurring with fidelity term 45, i.e.,

$$x_0 = \operatorname{argmin}_{x \in \mathbb{R}^{N \times \ell}} \mathrm{TV}(x) + 45\|\mathcal{T}(x) - y\|_F^2.$$

This is solved by the split Bregman method of Getreuer [26], using code from https://getreuer.info/posts/tvreg/index.html (accessed 10 August 2022), with 50 split Bregman iterations and a tolerance of 10^{-5}. The initial segmentation u_0 of x_0 is computed via the SDIE scheme with the above parameters and initial state $u^0 = 0.47\chi_Y + f$.

8.5.3. Example results. Before discussing the choice of parameters, timings, and accuracy in more detail, we present an example run of the reconstruction-segmentation method for the noisy data and set-up discussed in the beginning of this section. Here, we use the parameter setting from subsection 8.5.2. We show the results of this example run in Figure 13.

[10] As $\mathcal{T}$ is self-adjoint, $\|\mathcal{T}\|$ is the modulus of the largest eigenvalue of $\mathcal{T}$. Let $\mathcal{T}x = \lambda x$, where $\lambda \in \mathbb{R}$ since $\mathcal{T}$ is real and symmetric. Since $\mathcal{T}(\mathbf{1}) = \mathbf{1}$, $\mathcal{T}$ is represented by a nonnegative matrix, and $-\|x\|_\infty \mathbf{1} \le x \le \|x\|_\infty \mathbf{1}$ elementwise, by applying $\mathcal{T}$ to the latter we get $-\|x\|_\infty \mathbf{1} \le \lambda x \le \|x\|_\infty \mathbf{1}$ elementwise, and so $|\lambda| \le 1$.

Figure 13. *Reconstruction and segmentation of Example 8.3 from a single run of the reconstruction-segmentation algorithm using the parameters from subsection 8.5.2. Each column contains the reconstruction (top) and segmentation (bottom). We show (left to right) initialization, final iteration (= 15), and the output with the best scores over the run.*

The effect of an increasing contrast between cows and background in the reconstruction, which we already observed for the denoising results in subsection 8.4.2, is even more visible in this deblurring reconstruction. In contrast to the denoising setting, here this even leads to reconstructions with PSNRs that deteriorate over the run time of the algorithm.

8.5.4. Parameters, accuracy, and timings. As in subsection 8.4.3, we now study the deblurring-segmentation of Example 8.3 more quantitatively. We again consider timings of the total runs and of the key steps, as well as reconstruction and segmentation accuracy. Again, we look at four parameter settings: the setting from subsection 8.5.2, a change in the segmentation parameters compared to subsection 8.5.2 ($K = 100$ (decrease), $\varepsilon = \tau = 0.00285$ (increase), $u^0 = 0.45\chi_Y + f$ (lower constant on Y)), an increase in the segmentation weight ($\beta = 1.52 \cdot 10^{-5}$), and an increase in the reconstruction parameters ($\alpha = \eta_n = 10$). We list the results of these settings in Table 3 and present them in Figures 14 to 18.

We now comment on those simulation results. As mentioned before, we see that in most settings, the reconstruction PSNR is reduced over the course of the algorithm. The joint reconstruction-segmentation algorithm enhances the constrast between segments to a point where the reconstruction quality suffers (see Figure 14). Unlike in most settings we used in the denoising problem, here the segmentation accuracy is not always monotonically increasing.

It is interesting to note that (as shown in Figure 16) ε, τ should be chosen smaller for the deblurring problem than for Example 8.2 or the noise-free problem (see [13]). In the Allen–Cahn equation, a smaller ε leads to a smaller interface and, thus, a harder thresholding.

Table 3

Results averaged over 50 *runs for different parameter settings for Example* 8.3. *Values are given as "mean(standard deviation)." Total timing is for* 15 *iterations of the algorithm.*

	Changes to subsection 8.5.2	$\emptyset$	$K, \varepsilon, \tau, u^0$	β	α, η_n
	Figure	15	16	17	18
Accuracy	Dice score [%]	86.99(0.63)	77.23(12.96)	85.11(0.98)	85.79(0.82)
	PSNR [dB]	24.16(0.15)	22.30(3.52)	18.80(0.23)	27.32(0.06)
Timings [s]	total	179.73(0.97)	124.21(4.73)	179.60(0.58)	177.53(0.39)
	initialization	3.31(0.09)	3.93(0.18)	3.30(0.09)	3.29(0.06)
	reconstruction	5.91(0.11)	5.44(0.77)	5.97(0.09)	5.87(0.06)
	segmentation	5.85(0.08)	2.58(0.24)	5.78(0.06)	5.75(0.06)

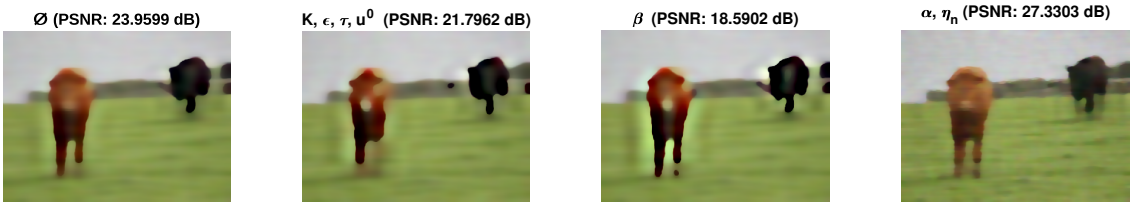

Figure 14. *Example reconstruction results obtained from the deblurring-reconstruction method for the different parameter settings for Example* 8.3 *described in subsection* 8.5.4.

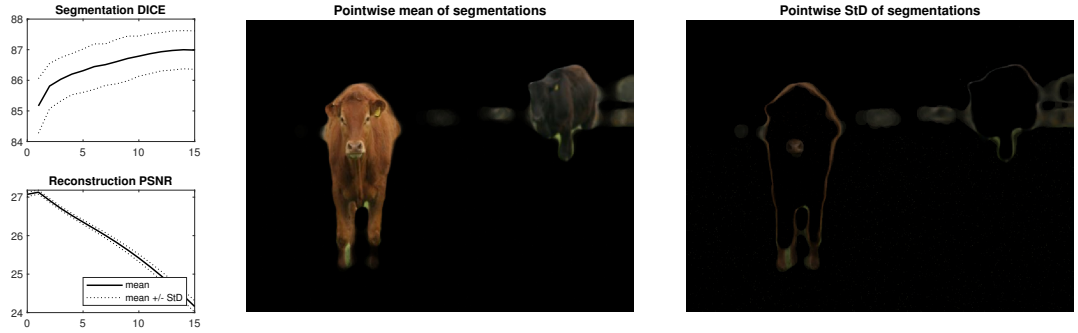

Figure 15. *Results averaged over* 50 *runs of the reconstruction-segmentation scheme in the deblurring case. The line plots (left) show the evolution of the Dice score (%) and PSNR (dB) over the* 15 *iterations of the algorithm. The images (center and right) show mean and standard deviation of the segmentation at* $t = 15$ *which are used as weights for the ground truth images. The setting is that of subsection* 8.5.2.

A harder thresholding should lead to a stronger regularization which aids the deblurring. Indeed, the softer thresholding in Figure 16 leads to more parts of the background being incorrectly identified as cow. Moreover, an even larger Nyström rank K is required. If $K = 100$, the results have a large variance and barely improve the initial segmentation on average.

When increasing β, we see a slight increase in the Dice standard deviation, which might imply that the method becomes more unstable when increasing the influence of the Ginzburg–Landau energy. When increasing α, as expected, we see an increased reconstruction accuracy.

In the case where $\beta = 1.5 \cdot 10^{-5}$ (Figure 17), we see a certain long-term instability: the Dice score reaches its maximum at iteration step 9 but is considerably lower at the end of the

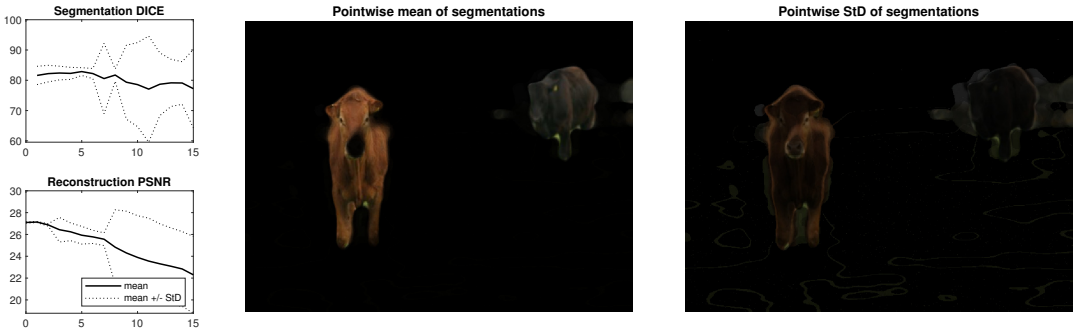

Figure 16. *Results averaged over* 50 *runs of the reconstruction-segmentation scheme in the deblurring case. The line plots (left) show the evolution of the Dice score (%) and PSNR (dB) over the* 15 *algorithm iterations. The images (center and right) show mean and standard deviation of the segmentation at* $t = 15$ *which are used as weights for the ground truth images. Setting is as in subsection* 8.5.2*, with changes* $\varepsilon = \tau = 0.00285$*,* $K = 100$*,* $u^0 = 0.45\chi_Y + f$*.*

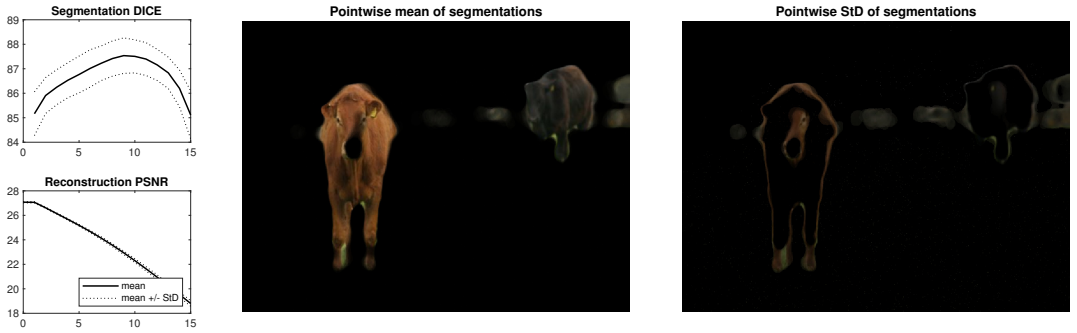

Figure 17. *Results averaged over* 50 *runs of the reconstruction-segmentation scheme in the deblurring case. The line plots (left) show the evolution of the Dice score (%) and PSNR (dB) over the* 15 *iterations of the algorithm. The images (center and right) show mean and standard deviation of the segmentation at* $t = 15$ *which are used as weights for the ground truth images. The setting is that of subsection* 8.5.2*, subject to the change* $\beta = 1.5 \cdot 10^{-5}$*.*

Figure 18. *Results averaged over* 50 *runs of the reconstruction-segmentation scheme in the deblurring case. The line plots (left) show the evolution of the Dice score (%) and PSNR (dB) over the* 15 *iterations of the algorithm. The images (center and right) show mean and standard deviation of the segmentation at* $t = 15$ *which are used as weights for the ground truth images. The setting is that of subsection* 8.5.2*, subject to the change* $\alpha = \eta_n = 10$*.*

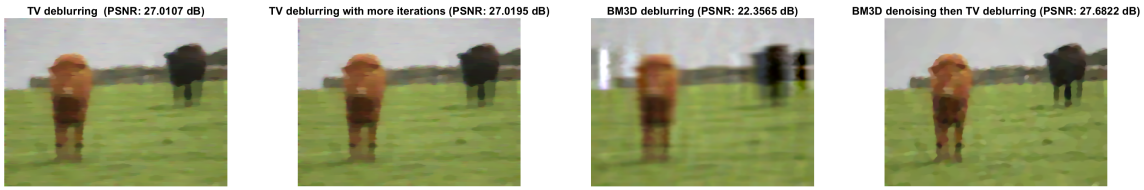

Figure 19. *Typical deblurred output for the TV-based deblurrings, BM3D deblurring, and BM3D denoising followed by TV deblurring.*

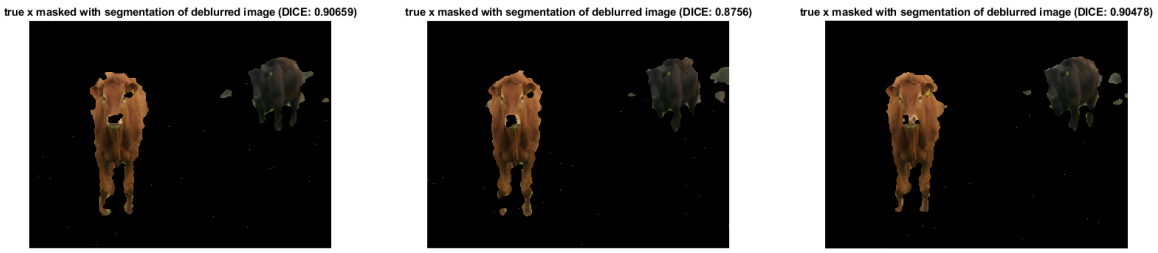

Figure 20. *The true image masked with three typical MBO segmentations of the BM3D denoised and TV deblurred image.*

algorithm. In iteration step 9, we obtain an average Dice of 0.8754 ($\pm$0.0073), beating all of the results reported in Table 3. Hence, the number of iterations is also an important tuning parameter as the system can experience metastability.

8.5.5. Comparison to sequential reconstruction-segmentation. As in the denoising case, we observe that (except in the setting of Figure 16) our joint scheme outperforms the sequential TV-based initialization in terms of Dice score (and in the setting of Figure 18, also in PSNR).

For a fairer comparison, we seek to compare the performance of our scheme to that of a sequential method using a more sophisticated deblurrer. We consider three alternative deblurrers:[11] the TV-based deblurrer but with 500 split Bregman iterations and tolerance 10^{-10}; the BM3D deblurrer from Mäkinen, Azzari, and Foi [38] (i.e., `BM3DDEB` in the associated software); and a BM3D denoising followed by a TV deblurring (with fidelity term 150, 100 split Bregman iterations, and tolerance 10^{-7}). Typical deblurrings via these methods are shown in Figure 19.

We segment only the latter of these (via MBO segmentation), as it is the only one with a perceptible improvement over the TV deblurring. The mean Dice score $\pm$ standard deviation (over 50 trials) for MBO segmentations of this deblurred image (with $\tau = \varepsilon = 0.00285$, $K = 200$, and initial state $0.45\chi_Y + f$; cf. Figure 16) is 0.8737 ± 0.0143.[12] Segmentations from the first three of these runs are shown in Figure 20. Visually, these segmentations appear slightly patchier than those we obtain with our joint method (cf. Figure 13), but they have higher Dice scores (with the exception of the metastable optimal segmentation observed in

[11] We also tested the built-in MATLAB deblurrers, but these did much worse than the TV deblurring.

[12] Taking $\tau = \varepsilon = 0.002$ and initial state $0.47\chi_Y + f$ (as in subsection 8.5.2) performs slightly worse, with mean Dice score $\pm$ standard deviation (over 50 runs) equal to 0.8679 ± 0.0095.

Figure 17, which has a slightly higher mean Dice score). However, the sequential deblurring-segmentations are much faster than the joint method; over 50 runs, their mean ($\pm$ standard deviation) reconstruction time is $23.45(\pm 0.16)s$, segmentation time $5.85(\pm 0.05)s$, and total time $29.30(\pm 0.17)s$.

9. Conclusions and directions for future work. In this paper, we have developed a joint reconstruction-segmentation scheme which incorporates the highly effective graph-based segmentation techniques that have been developed over the past decade. There are numerous challenges which arise in the efficient implementation of this scheme, but we have shown how these obstacles can be navigated. Furthermore, we have shown how the Kurdyka–Łojasiewicz-based theory of [5, 9] can be applied to show the convergence of our scheme.

Finally, we have tested our scheme on highly noised and blurry counterparts of the "two cows" image familiar from the literature. In the denoising case, we observed that our scheme gives very accurate segmentations despite the very high noise level and gives reasonably accurate reconstructions, with a run time of about 2.5 minutes. Moreover, our joint scheme substantially outperforms sequential denoising-segmention methods, in both segmentation and reconstruction accuracy, even when much more sophisticated denoisers are employed, albeit at the cost of a much longer run time.

In the deblurring case, again our scheme gives highly accurate segmentations (with a run time of about 3 minutes), but in the reconstructions it introduces an artificial level of contrast between the "cows" and the "background." This aids the segmentation accuracy at the cost of the reconstruction accuracy deteriorating over the course of the iterations. Increasing the reconstruction weighting prevents this effect, at the expense of a lower segmentation accuracy. Increasing the segmentation weighting has the curious effect of producing a very accurate but metastable (w.r.t. a change in the number of iterations) segmentation after nine iterations. Compared to sequential deblurring-segmentation, our scheme produces worse reconstructions and on the whole slightly worse segmentations (with the exception of the metastable segmentation, which is slightly better) and runs considerably slower. However, it should be noted that the deblurring method which was used is more sophisticated than the one used within our scheme and that our joint scheme does give substantially more accurate segmentations than sequential deblurring-segmentation using only a TV-based deblurrer (i.e., our initialization process).

There are three major directions for future work. First, the scheme in its current form has many parameters which must be tuned by hand. Future work will seek to develop techniques for tuning these parameters in a more principled way, so that this scheme can be applied to a large image set without the need for constant manual retuning.

Second, we have in this work applied our scheme only to artificially noised/blurred images, and in the comparison to sequential methods we did not exhaust the state of the art. Future work will seek to test our scheme on real observations, with potentially unknown ground truths and/or forward maps and compare our scheme to other state-of-the-art methods (including other joint reconstruction-segmentation methods such as those in Corona et al. [19]).

Finally, there are a number of potential ways to make our scheme more accurate. One is to use a different regularizer. Candidates of particular interest are implicit regularization with a "Plug-and-Play" denoiser (as in Venkatakrishnan, Bouman, and Wohlberg [50]) and learned regularization (see, e.g., Arridge et al. [4, section 4]). Another potential improvement

is the use of more sophisticated feature maps (e.g., the CNN-VAE maps used in Miller et al. [35]), though in this choice one is constrained by the need to compute both the map and its adjoint very efficiently. Moreover, the theory in this paper assumes a linear feature map. A third opportunity for improvement lies in solving (3.3a) without resorting to linearization.

Appendix A. Computing g_n. In this appendix we explain how we compute g_n, which is defined in section 4. We recall from subsection 3.1 that $z := \mathcal{F}(x)$, and likewise we define $z_n := \mathcal{F}(x_n)$. Then

$$g_n = \mathcal{F}^*(\nabla_z \mathscr{F}_1(z_n) + \nabla_z \mathscr{F}_2(z_n)).$$

We now compute each term in turn. To compute $\nabla_z \mathscr{F}_1(z)$, note that

$$\begin{aligned}\mathscr{F}_1(z+\delta z) &= \beta\langle (G_n)_{YY}, \Omega_{YY}(z+\delta z)\rangle_F \\ &= \beta\left\langle (G_n)_{YY}, \Omega_{YY}(z) + [\langle \nabla_z \Omega_{ij}(z), \delta z\rangle_F]_{i,j\in Y}\right\rangle_F + o(\delta z) \\ &= \mathscr{F}_1(z) + \beta \sum_{i,j\in Y} (G_n)_{ij} \sum_{l\in Y}\sum_{r=1}^{q} (\nabla_z \Omega_{ij}(z))_{lr}\delta z_{lr} + o(\delta z)\end{aligned}$$

and thus for all $l \in Y$ and $r \in \{1,\dots,q\}$

$$(\nabla_z \mathscr{F}_1(z))_{lr} = \beta \sum_{i,j\in Y} (G_n)_{ij} (\nabla_z \Omega_{ij}(z))_{lr}.$$

Now, since $(\Omega_{YY})_{ij}(z) = e^{-\|z_i - z_j\|_2^2/q\sigma^2}$ for $i \neq j$ and $(\Omega_{YY})_{ii}(z) = 0$, we have

$$\nabla_{z_{lr}} (\Omega_{YY})_{ij}(z) = \frac{2}{q\sigma^2} \begin{Bmatrix} 0, & l \notin \{i,j\} \\ (\Omega_{YY})_{il}(z)(z_{ir} - z_{lr}), & j = l \\ (\Omega_{YY})_{lj}(z)(z_{lr} - z_{jr}), & i = l \end{Bmatrix} = \frac{2}{q\sigma^2} (\Omega_{YY})_{ij}(z)(z_{ir} - z_{jr})(\delta_{jl} - \delta_{il}),$$

where δ denotes the Kronecker delta. Therefore,

$$(\nabla_z \mathscr{F}_1(z))_{lr} = \frac{2\beta}{q\sigma^2} \sum_{i,j\in Y} (G_n)_{ij} \Omega_{ij}(z)(z_{ir} - z_{jr})(\delta_{jl} - \delta_{il}).$$

Hence, letting $\mathcal{A}(z) := (G_n)_{YY} \odot \Omega_{YY}(z)$ (where $\odot$ denotes the Hadamard product),

$$(\nabla_z \mathscr{F}_1(z))_{lr} = \frac{2\beta}{q\sigma^2} \sum_{i,j\in Y} \mathcal{A}_{ij}(z)(z_{ir} - z_{jr})(\delta_{jl} - \delta_{il}) = \frac{4\beta}{q\sigma^2} \left(\sum_{j\in Y} \mathcal{A}_{lj}(z) z_{jr} - z_{lr} \sum_{j\in Y} \mathcal{A}_{lj}(z) \right)$$

since $\mathcal{A}(z)$ is symmetric, and therefore

$$\nabla_z \mathscr{F}_1(z) = \frac{4\beta}{q\sigma^2} \left(\mathcal{A}(z) z - (\mathcal{A}(z)\mathbf{1}_N) \odot z \right).$$

To compute $\nabla_z \mathscr{F}_2(z)$, note that by a similar argument as the above,

$$(\nabla_z \mathscr{F}_2(z))_{lr} = -\frac{4\beta}{q\sigma^2} \sum_{i\in Y, j\in Z} (G_n)_{ij} (\Omega_{YZ})_{ij}(z, z_d)(z_{ir} - (z_d)_{jr})\delta_{il}$$

for all $l \in Y$ and $r \in \{1, \dots, q\}$. Hence, letting $\mathcal{B}(z) := (G_n)_{YZ} \odot \Omega_{YZ}(z, z_d)$, we get

$$(\nabla_z \mathscr{F}_2(z))_{lr} = \frac{4\beta}{q\sigma^2} \sum_{i \in Y, j \in Z} \mathcal{B}_{ij}(z)((z_d)_{jr} - z_{ir})\delta_{il} = \frac{4\beta}{q\sigma^2} \left(\sum_{j \in Z} \mathcal{B}_{lj}(z)(z_d)_{jr} - z_{lr} \sum_{j \in Z} \mathcal{B}_{lj}(z) \right)$$

and therefore we arrive at a similar formula as for $\nabla_z \mathscr{F}_1(z)$:

$$\nabla_z \mathscr{F}_2(z) = \frac{4\beta}{q\sigma^2} \left(\mathcal{B}(z) z_d - (\mathcal{B}(z) \mathbf{1}_{N_d}) \odot z \right).$$

Tying this all together, we get

$$\begin{aligned} g_n &= \frac{4\beta}{q\sigma^2} \mathcal{F}^* (\mathcal{A}(z_n) z_n + \mathcal{B}(z_n) z_d - (\mathcal{A}(z_n) \mathbf{1}_N + \mathcal{B}(z_n) \mathbf{1}_{N_d}) \odot z_n) \\ &= \frac{4\beta}{q\sigma^2} \mathcal{F}^* \left(\mathcal{C}_n \begin{pmatrix} z_n \\ z_d \end{pmatrix} - (\mathcal{C}_n \mathbf{1}_{N+N_d}) \odot z_n \right), \end{aligned} \tag{A.1}$$

where $\mathcal{C}_n := (G_n)_{YV} \odot \Omega_{YV}(z_n, z_d)$. To compute (A.1), we need to compute matrix-vector products of the form $\mathcal{C}_n v$. Recalling (4.1), it follows that

$$(G_n)_{YV} = -u_n|_Y u_n^T + v_n|_Y \mathbf{1}_V^T + \mathbf{1}_Y v_n^T.$$

Next, we observe a neat linear algebra result,[13]

$$\left((-u_n|_Y u_n^T + v_n|_Y \mathbf{1}_V^T + \mathbf{1}_Y v_n^T) \odot A \right) v = -u_n|_Y \odot (A(u_n \odot v)) + v_n|_Y \odot (Av) + A(v_n \odot v),$$

where in this case $A = \Omega_{YV}(z_n, z_d)$. Hence it suffices to be able to compute terms of the form $\Omega_{YV}(z_n, z_d)v$. Via the Nyström extension (2.1) we have

$$\Omega_{YV}(\mathcal{F}(x_n), z_d)v \approx \left(\Omega_{VX}(\mathcal{F}(x_n), z_d) \Omega_{XX}^{-1}(\mathcal{F}(x_n), z_d) \Omega_{XV}(\mathcal{F}(x_n), z_d) v \right)|_Y, \tag{A.2}$$

where $X \subseteq V$ is some interpolation set, so we can compute such products quickly.

These considerations lead us to Algorithm A.1 to compute $\mathcal{C}_n v$ for $v \in \mathcal{V}$.

Appendix B. Primal-dual optimization methods for (4.3). To solve (4.3), we shall be employing an algorithm of Chambolle and Pock [16]. Following [16], we rewrite (4.3) as

$$\min_{x \in \mathbb{R}^{N \times \ell}} R(\mathcal{K}x) + G(x), \tag{B.1}$$

where $G(x) := \alpha \|\mathcal{T}(x) - y\|_F^2 + \eta_n \|x - \tilde{x}_n\|_F^2$. Then the primal problem (B.1) can be reformulated as the primal-dual saddle point problem (where we recall the definition of R^* from footnote 3)

$$\min_{x \in \mathbb{R}^{N \times \ell}} \max_{p \in \mathcal{K}(\mathbb{R}^{N \times \ell})} \langle \mathcal{K}x, p \rangle + G(x) - R^*(p). \tag{B.2}$$

[13] To see this, observe that in suffix notation, for $i \in Y$ and $j \in V$, the left-hand side is $(-(u_n)_i (u_n)_j + (v_n)_i + (v_n)_j) A_{ij} v_j$ and the right-hand side is $-(u_n)_i A_{ij}((u_n)_j v_j) + (v_n)_i A_{ij} v_j + A_{ij}((v_n)_j v_j)$.

Algorithm A.1 Definition of the `CProd` function to be used in Algorithm 4.1.

function CPROD($z_n, v, u, v_n, \sigma, V, Y, Z, K$) ▷ Approximates $\mathcal{C}_n v$ as above
  $B : w \mapsto (\texttt{OmegaProd}(w, z_n, z_d, q, \sigma, V, Y, Z, K))|_Y$ ▷ See below
  return $-u|_Y \odot B(u \odot v) + v_n|_Y \odot B(v) + B(v_n \odot v)$
end function
function OMEGAPROD($v, z, z_d, q, \sigma, V, Y, Z, K$) ▷ Approximates $\Omega(z, z_d)v$ via the Nyström extension as in (A.2)
  $\omega : ij \mapsto \Omega_{ij}(z, z_d, q, \sigma)$ ▷ Defined as in (3.1)
  $X = \texttt{random\_subset}(Y, K/2) \cup \texttt{random\_subset}(Z, K/2)$
  $(\omega_{XX}, \omega_{VX}) = (\omega(X, X), \omega(V, X))$
  $\omega_{XX} v' = \omega_{VX}^T v$ ▷ Solving the linear system for v'
  return $\omega_{VX} v'$
end function

Algorithm B.1 Algorithm for solving (B.1), using [16, Algorithm 2].

function PRIMALDUAL($x_0, \gamma, \mathcal{K}, \mathcal{K}^*, \texttt{proxRS}, \texttt{proxG}$) ▷ γ must satisfy [16, (35)]
  $p_0 = \mathcal{K}(x_0)$
  $\bar{x}_0 = x_0$
  $(\delta t_{0,1}, \delta t_{0,2}) = (1/\|\mathcal{K}\|, 0.99/\|\mathcal{K}\|)$ ▷ We must have $\delta t_{0,1} \delta t_{0,2} \|\mathcal{K}\|^2 \le 1$
  $n = 0$
  while stopping condition not met **do**
    $p_{n+1} = \texttt{proxRS}(p_n + \delta t_{n,1} \mathcal{K}(\bar{x}_n), \delta t_{n,1})$ ▷ $\texttt{proxRS}(p, \delta t) := \text{prox}_{\delta t R^*}(p)$
    $x_{n+1} = \texttt{proxG}(x_n - \delta t_{n,2} \mathcal{K}^*(p_{n+1}), \delta t_{n,2})$ ▷ $\texttt{proxG}(x, \delta t) := \text{prox}_{\delta t G}(x)$
    $\theta_n = (1 + 2\gamma \delta t_{n,2})^{-\frac{1}{2}}$
    $(\delta t_{n+1,1}, \delta t_{n+1,2}) = (\delta t_{n,1} \theta_n^{-1}, \delta t_{n,2} \theta_n)$
    $\bar{x}_{n+1} = x_{n+1} + \theta_n (x_{n+1} - x_n)$
    $n = n + 1$
  end while
  return x_n
end function

Reformulating (B.1) as (B.2) suggests approaching the minimizer of (B.1) by alternately updating a sequence of x_n and p_n. The algorithm we shall be using, i.e., [16, Algorithm 2], is a sophistication of this basic idea and is summarized as Algorithm B.1. For details and a convergence analysis, see [16].

We shall assume that for any $\delta t > 0$ we can efficiently compute the proximal operator of $\delta t R^*$, $\text{prox}_{\delta t R^*}$.[14] So to employ Algorithm B.1, we need two ingredients: a method for

[14] For a proper, l.s.c., and convex function f, the *proximal* (a.k.a. *resolvent*) *operator of* f is defined by $\text{prox}_f(x') := \text{argmin}_x f(x) + \frac{1}{2}\|x - x'\|^2$. As mentioned in [16], Moreau's identity allows $\text{prox}_{\delta t R^*}$ to be computed via $\text{prox}_{\delta t^{-1} R}$. Since computing $\text{prox}_{\delta t^{-1} R}$ is equivalent to performing a denoising regularized by R, this is a place where "Plug-and-Play" denoising methods (see Venkatakrishnan, Bouman, and Wohlberg [50]) could be employed instead of the explicit regularization methods we will use in this paper.

computing $\text{prox}_{\delta t G}$, and a γ obeying [16, (35)], i.e., such that for all $x, x' \in \mathbb{R}^{N\times\ell}$,

$$G(x') \geq G(x) + \langle \nabla_x G(x), x' - x\rangle_F + \frac{1}{2}\gamma\|x - x'\|_F^2.$$

First, assume that $\mathcal{T}$ is linear. Then it follows that $\nabla_x G(x) = 2\alpha\mathcal{T}^*(\mathcal{T}(x)-y)+2\eta_n(x-\tilde{x}_n)$ and thus we require that

$$\begin{aligned}
&\frac{1}{2}\gamma\|x - x'\|_F^2 \\
&\quad\leq \alpha\langle \mathcal{T}(x) + \mathcal{T}(x') - 2y, \mathcal{T}(x') - \mathcal{T}(x)\rangle_F + \eta_n\langle x + x' - 2\tilde{x}_n, x' - x\rangle_F \\
&\qquad - \langle 2\alpha\mathcal{T}^*(\mathcal{T}(x) - y) + 2\eta_n(x - \tilde{x}_n), x' - x\rangle_F \\
&\quad= \alpha\left(\langle \mathcal{T}(x) + \mathcal{T}(x') - 2y, \mathcal{T}(x' - x)\rangle_F - 2\langle \mathcal{T}(x) - y, \mathcal{T}(x' - x)\rangle_F\right) + \eta_n\|x - x'\|_F^2 \\
&\quad= \alpha\|\mathcal{T}(x' - x)\|_F^2 + \eta_n\|x - x'\|_F^2,
\end{aligned}$$

and hence it suffices to take $\gamma = 2\eta_n$. Finally, recall that $\text{prox}_{\delta t G}$ is defined by

$$\text{prox}_{\delta t G}(x) = \text{argmin}_{x'\in\mathbb{R}^{N\times\ell}} G(x') + \frac{\|x' - x\|_F^2}{2\delta t},$$

which has unique minimizer x' solving $\nabla_x G(x') + \frac{1}{\delta t}(x' - x) = 0$. Solving for x', we get

$$\text{prox}_{\delta t G}(x) = \left((\delta t^{-1} + 2\eta_n)I + 2\alpha\mathcal{T}^*\mathcal{T}\right)^{-1}\left(2\alpha\mathcal{T}^*(y) + 2\eta_n\tilde{x}_n + \frac{x}{\delta t}\right). \tag{B.3}$$

If $\mathcal{T}$ is nonlinear, things are more difficult. There may not exist a valid γ, and if so one must use a method such as [16, Algorithm 1] to solve (4.3), which has slower convergence. We must still compute $\text{prox}_{\delta t G}(x)$. Since $\mathcal{T}$ is assumed to be differentiable, for all x there exists a linear map $D\mathcal{T}_x$ such that $\mathcal{T}(x + \delta x) = \mathcal{T}(x) + D\mathcal{T}_x(\delta x) + o(\delta x)$. Then it follows that

$$\nabla_x G(x) = 2\alpha(D\mathcal{T}_x)^*(\mathcal{T}(x) - y) + 2\eta_n(x - \tilde{x}_n)$$

and hence $x' := \text{prox}_{\delta t G}(x)$ solves

$$(1 + 2\delta t\eta_n)x' + 2\alpha\delta t(D\mathcal{T}_{x'})^*(\mathcal{T}(x') - y) = x + 2\delta t\eta_n\tilde{x}_n. \tag{B.4}$$

Finally, (B.4) can be solved by numerical root-finding methods (see, e.g., [41, section 9]).

Appendix C. Computing the SDIE scheme. In this appendix we describe how we compute the SDIE scheme that is described in subsection 5.1. By Theorem 5.2, an SDIE update has two steps: a fidelity-forced diffusion and a piecewise linear thresholding. The thresholding is trivial to compute, but the diffusion is nontrivial. Our method for computing fidelity-forced diffusion was described in detail in [11, section 5.2.6], so here we only reproduce the key details.

By Theorem 5.1, given u_n and the parameter $\tau > 0$, we seek to compute

$$\mathcal{S}_\tau u_n = e^{-\tau(\Delta + M')}u_n + b,$$

where $b := F_\tau(\Delta + M')M'f'$. To compute $e^{-\tau(\Delta+M')}u_n$, we use the Strang formula [45]:

Algorithm C.1 Nyström-QR method for computing an approximate SVD of Δ or Δ_s.

function NYSTRÖMQR($ij \mapsto \omega_{ij}, V, Z, K_1, K_2$) ▷ Computes U_1, Λ, and U_2, where $\Delta \approx U_1 \Lambda U_2^T$ is an approximate SVD of rank $K := K_1 + K_2$
  $X = \texttt{random\_subset}(V \setminus Z, K_1) \cup \texttt{random\_subset}(Z, K_2)$
  $(\omega_{XX}, \omega_{VX}) = (\omega(X,X), \omega(V,X))$
  $\hat{d} = \omega_{VX}(\omega_{XX}^{-1}(\omega_{VX}^T \mathbf{1}))$ ▷ Uses (2.1) to approximate $d = \omega\mathbf{1}$
  $\tilde{\omega}_{VX} = \hat{d}^{-1/2} \odot \omega_{VX}$ ▷ Applying $\odot$ columnwise, i.e., $(\tilde{\omega}_{VX})_{ij} = \hat{d}_i^{-1/2}(\omega_{VX})_{ij}$
  $[Q, R] = \texttt{thin\_qr}(\tilde{\omega}_{VX})$ ▷ Computes thin QR decomposition $\tilde{\omega}_{VX} = QR$
  $S = R\omega_{XX}^{-1}R^T$ ▷ N.B. $S \in \mathbb{R}^{K \times K}$
  $S = (S + S^T)/2$ ▷ Corrects symmetry-breaking computational errors
  $[\Phi, \Sigma] = \texttt{eig}(S)$ ▷ Computes eigendecomposition $S = \Phi\Sigma\Phi^T$
  $\Lambda = I_K - \Sigma$
  $U_s = Q\Phi$ ▷ $\Delta_s \approx U_s \Lambda U_s^T$, so to return the decomposition of Δ_s terminate here
  $U_1 = \hat{d}^{-1/2} \odot U_s$ ▷ i.e., $(U_1)_{ij} = \hat{d}_i^{-1/2}(U_s)_{ij}$
  $U_2 = \hat{d}^{1/2} \odot U_s$ ▷ i.e., $(U_2)_{ij} = \hat{d}_i^{1/2}(U_s)_{ij}$
  return U_1, Λ, U_2
end function

$$e^{-\tau(\Delta+M')} = \left(e^{-\frac{\tau}{2m}M'} e^{-\frac{\tau}{m}\Delta} e^{-\frac{\tau}{2m}M'}\right)^m + \mathcal{O}\left(m^{-2}\right) = \left(e^{-\frac{1}{2}\delta t M'} e^{-\delta t \Delta} e^{-\frac{1}{2}\delta t M'}\right)^m + \mathcal{O}\left(\delta t^2\right),$$

where $m \in \mathbb{N}$ and $\delta t := \tau/m$. Computing matrix-vector products with $e^{-\frac{1}{2}\delta t M'}$ is straightforward, since M' is a diagonal matrix. To compute matrix-vector products with $e^{-\delta t \Delta}$, we will compute an approximate eigendecomposition of Δ using the Nyström-QR method (recommended by Alfke et al. [3]) described in Algorithm C.1. For details on this method, see [11, section 5.2.3].

Note C.1. Algorithm C.1 really computes an approximate decomposition $U_s \Sigma U_s^T$ of $\tilde{\omega} := D^{-1/2}\omega D^{-1/2}$ and then makes a further approximation $\Delta_s = I - \tilde{\omega} \approx U_s(I_K - \Sigma)U_s^T = U_s \Lambda U_s^T$, where I_K is the $K \times K$ identity matrix, and so on for the random walk Laplacian.

In [13, 11], this Nyström-QR approximate decomposition was used to approximate the matrix exponential via $e^{-\delta t \Delta} \approx I + U_1(e^{-\delta t \Lambda} - I_K)U_2^T$. But by Note C.1, $\Delta \approx I - U_1 \Sigma U_2^T$ is a slightly more accurate approximation, and so we have the improved approximation:

$$e^{-\delta t \Delta} \approx e^{-\delta t}\left(I + U_1(e^{\delta t \Sigma} - I_K)U_2^T\right).$$

Therefore, we compute $v_m \approx e^{-\tau(\Delta+M')}u_n$ by defining $v_0 := u_n$, and v_r for $r \in \{1, \ldots, m\}$ by

$$\begin{aligned} v_{r+1} &= e^{-\delta t} e^{-\delta t M'} v_r + e^{-\delta t} e^{-\frac{1}{2}\delta t M'} U_1(e^{\delta t \Sigma} - I_K)U_2^T e^{-\frac{1}{2}\delta t M'} v_r \\ &= a_1(\delta t) \odot v_r + a_3(\delta t) \odot \left(U_1\left(a_2(\delta t) \odot \left(U_2^T\left(a_3(\delta t) \odot v_r\right)\right)\right)\right), \end{aligned} \tag{C.1}$$

where $a_1(\delta t) := \exp(-\delta t(\mu' + \mathbf{1}))$, $a_2(\delta t) := \exp(\delta t \operatorname{diag}(\Sigma)) - \mathbf{1}_K$, and $a_3(\delta t)$ is the elementwise square root of $a_1(\delta t)$ (where exp is applied elementwise, and $\mathbf{1}_K$ is the vector of K ones).

Finally we note that, by Theorem 5.1, b is the fidelity-forced diffusion with initial condition $u_0 = \mathbf{0}$ at time τ. We compute b via the semi-implicit Euler scheme used for fidelity-forced diffusion in [34]. To compute this, again we use the Nyström-QR decomposition of Δ.

Note C.2. We do not use the scheme from [34] for all of the fidelity-forced diffusions because the Strang formula is more accurate for the $e^{-\tau(\Delta+M')}u_n$ term, see [11, section 5.2.5–5.2.7] for details.

Appendix D. Examples of subanalytic regularizers.

The following theorem shows that the examples of regularizers $\mathcal{R}$ that are given in Note 7.6 do indeed satisfy the conditions required by Assumption 7.5.

THEOREM D.1. *The following functions are semianalytic, bounded below, and continuous on their domain (which is $\mathbb{R}^n$):*

i. $f : x \mapsto \|Ax\|_1$ *for* $A \in \mathbb{R}^{m\times n}$.

ii. *A feedforward neural network* $\mathcal{NN} := f_L \circ \cdots \circ f_1$ *where* $f_j(x') := \rho(A^{(j)}x' + b^{(j)})$, $A^{(j)} \in \mathbb{R}^{n_j \times n_{j-1}}$, $b^{(j)} \in \mathbb{R}^{n_j}$, $n_0 = n$, $n_L = 1$, *and* $(\rho(x'))_i := \max\{0, x_i'\}$ *(the ReLU function).*

Proof.

i. $\mathrm{Gr} f = \{(x, \|Ax\|_1) \mid x \in \mathbb{R}^n\}$ can be written as

$$\bigcup_{e\in\{-1,1\}^m} \bigcap_{j=1}^{m} \left\{ (z_1, z_2) \in \mathbb{R}^{n+1} \;\middle|\; z_2 - \sum_{i=1}^{m} e_i (Az_1)_i = 0 \text{ and } e_j (Az_1)_j \geq 0 \right\}$$

and thus $\mathrm{Gr} f$ is a semianalytic set. The other properties are trivial.

ii. $\mathcal{NN}$ is a composition of piecewise linear continuous functions and is hence piecewise linear and continuous. It follows that it is semianalytic. It is bounded below due to ρ applying the ReLU function.

■

REFERENCES

[1] B. ADCOCK AND A. C. HANSEN, *Compressive Imaging: Structure, Sampling, Learning*, Cambridge University Press, Cambridge, UK, 2021, https://doi.org/10.1017/9781108377447.

[2] J. ADLER, S. LUNZ, O. VERDIER, C.-B. SCHÖNLIEB, AND O. ÖKTEM, *Task adapted reconstruction for inverse problems*, Inverse Problems, 38 (2022), 075006, https://doi.org/10.1088/1361-6420/ac28ec.

[3] D. ALFKE, D. POTTS, M. STOLL, AND T. VOLKMER, *NFFT meets Krylov methods: Fast matrix-Vector products for the graph laplacian of fully connected networks*, Front. Appl. Math. Stat., 4 (2018), 61, https://doi.org/10.3389/fams.2018.00061.

[4] S. ARRIDGE, P. MAASS, O. ÖKTEM, AND C.-B. SCHÖNLIEB, *Solving inverse problems using data-driven models*, Acta Numer., 28 (2019), pp. 1–174, https://doi.org/10.1017/S0962492919000059.

[5] H. ATTOUCH, J. BOLTE, P. REDONT, AND A. SOUBEYRAN, *Proximal alternating minimization and projection methods for nonconvex problems: An approach based on the Kurdyka-Łojasiewicz inequality*, Math. Oper. Res., 35 (2010), pp. 438–457.

[6] E. BAE AND E. MERKURJEV, *Convex variational methods on graphs for multiclass segmentation of high-dimensional data and point clouds*, J. Math. Imaging Vision, 58 (2017), pp. 468–493.

[7] A. BERTOZZI AND A. FLENNER, *Diffuse interface models on graphs for classification of high dimensional data*, Multiscale Model. Simul., 10 (2012), pp. 1090–1118, https://doi.org/10.1137/11083109X.

[8] J. Bolte, A. Daniilidis, and A. Lewis, *The Łojasiewicz inequality for nonsmooth subanalytic functions with applications to subgradient dynamical systems*, SIAM J. Optim., 17 (2007), pp. 1205–1223, https://doi.org/10.1137/050644641.

[9] J. Bolte, S. Sabach, and M. Teboulle, *Proximal alternating linearized minimization for nonconvex and nonsmooth problems*, Math. Program., 146 (2014), pp. 459–494, https://doi.org/10.1007/s10107-013-0701-9.

[10] K. Bredies, K. Kunisch, and T. Pock, *Total generalized variation*, SIAM J. Imaging Sci., 3 (2010), pp. 492–526, https://doi.org/10.1137/090769521.

[11] J. Budd, *Theory and Applications of Differential Equation Methods for Graph-based Learning*, Ph.D. thesis, Technische Universiteit Delft, 2022, https://doi.org/10.4233/uuid:b8e0648c-d38e-4f95-bcd7-b99a943cb2d1.

[12] J. Budd and Y. van Gennip, *Graph Merriman-Bence–Osher as a semidiscrete implicit Euler scheme for graph Allen-Cahn flow*, SIAM J. Math. Anal., 52 (2020), pp. 4101–4139, https://doi.org/10.1137/19M1277394.

[13] J. Budd, Y. van Gennip, and J. Latz, *Classification and image processing with a semi-discrete scheme for fidelity forced Allen-Cahn on graphs*, GAMM Mitt., 44 (2021), pp. 1–43, https://doi.org/10.1002/gamm.202100004.

[14] L. Calatroni, Y. Gennip, C.-B. Schönlieb, H. Rowland, and A. Flenner, *Graph clustering, variational image segmentation methods and hough transform scale detection for object measurement in images*, J. Math. Imaging Vision, 57 (2017), pp. 269–291, https://doi.org/10.1007/s10851-016-0678-0.

[15] J. Calder, B. Cook, M. Thorpe, and D. Slepcev, *Poisson learning: Graph based semi-supervised learning at very low label rates*, in Proceedings of the International Conference on Machine Learning, 2020, pp. 8588–8598.

[16] A. Chambolle and T. Pock, *A first-Order primal-Dual algorithm for convex problems with applications to imaging*, J. Math. Imaging Vision, 40 (2011), https://doi.org/10.1007/s10851-010-0251-1.

[17] T. Chan and L. Vese, *Active contours without edges*, IEEE Trans. Image Process., 10 (2001), pp. 266–277, https://doi.org/10.1109/83.902291.

[18] L. Condat, *A primal–dual splitting method for convex optimization involving lipschitzian, proximable and linear composite terms*, J. Optim. Theory Appl., 158 (2013), pp. 460–479, https://doi.org/10.1007/s10957-012-0245-9.

[19] V. Corona, M. Benning, M. J. Ehrhardt, L. F. Gladden, R. Mair, A. Reci, A. J. Sederman, S. Reichelt, and C.-B. Schönlieb, *Enhancing joint reconstruction and segmentation with non-convex Bregman iteration*, Inverse Problems, 35 (2019), 055001, https://doi.org/10.1088/1361-6420/ab0b77.

[20] M. de Vriendt, P. Sellars, and A. I. Aviles-Rivero, *The GraphNet Zoo: An all-in-one graph based deep semi-supervised framework for medical image classification*, in Uncertainty for Safe Utilization of Machine Learning in Medical Imaging, and Graphs in Biomedical Image Analysis, Springer, New York, 2020, pp. 187–197.

[21] S. Esedoḡlu and Y.-H. R. Tsai, *Threshold dynamics for the piecewise constant Mumford-Shah functional*, J. Comput. Phys., 211 (2006), pp. 367–384, https://doi.org/10.1016/j.jcp.2005.05.027.

[22] C. Fowlkes, S. Belongie, F. Chung, and J. Malik, *Spectral grouping using the Nyström method*, IEEE Trans. Pattern Anal. Mach. Intell., 26 (2004), pp. 214–225, https://doi.org/10.1109/TPAMI.2004.1262185.

[23] S. Gao and T. D. Bui, *Image segmentation and selective smoothing by using Mumford-Shah model*, IEEE Trans. Image Process., 14 (2005), pp. 1537–1549.

[24] C. Garcia-Cardona, E. Merkurjev, A. L. Bertozzi, A. Flenner, and A. G. Percus, *Multiclass data segmentation using diffuse interface methods on graphs*, IEEE Trans. Pattern Anal. Mach. Intell., 36 (2014), pp. 1600–1613, https://doi.org/10.1109/TPAMI.2014.2300478.

[25] P. Getreuer, *Rudin-Osher–Fatemi total variation denoising using split Bregman*, IPOL J. Image Process. Online, 2 (2012), pp. 74–95, https://doi.org/10.5201/ipol.2012.g-tvd.

[26] P. Getreuer, *Total variation deconvolution using split Bregman*, IPOL J. Image Process. Online, 2 (2012), pp. 158–174, https://doi.org/10.5201/ipol.2012.g-tvdc.

[27] G. H. Golub and C. F. van Loan, *Matrix Computations*, 4th ed., Johns Hopkins University Press, Baltimore, 2013, http://www.cs.cornell.edu/cv/GVL4/golubandvanloan.htm.

[28] H. Hironaka, *Subanalytic sets, number theory*, in Algebraic Geometry and Commutative Algebra, Kinokuniya, Tokyo, pp. 453–493.
[29] P. J. Huber, *Robust Statistics,* Wiley Ser. Probab. Stat., John Wiley & Sons, New York, 1981, https://doi.org/10.1002/0471725250.
[30] G. Li and T. K. Pong, *Calculus of the exponent of Kurdyka–Łojasiewicz inequality and its applications to linear convergence of first-order methods*, Found. Comput. Math., 18 (2017), pp. 1199–1232, https://doi.org/10.1007/s10208-017-9366-8.
[31] H. Li, H. Chen, M. Haberland, A. L. Bertozzi, and P. J. Brantingham, *PDEs on graphs for semi-supervised learning applied to first-person activity recognition in body-worn video*, Discrete Contin. Dyn. Syst., 41 (2021), pp. 4351–4373.
[32] S. Łojasiewicz, *Triangulation of semi-analytic sets*, Ann. Scuola Norm. Super. Pisa Cl. Sci., 18 (1964), pp. 449–474.
[33] X. Luo and A. Bertozzi, *Convergence of the graph Allen-Cahn scheme*, J. Stat. Phys., 167 (2017), pp. 934–958, https://doi.org/10.1007/s10955-017-1772-4.
[34] E. Merkurjev, T. Kostić, and A. Bertozzi, *An MBO scheme on graphs for classification and image processing*, SIAM J. Imaging Sci., 6 (2013), pp. 1903–1910, https://doi.org/10.1137/120886935.
[35] K. Miller, J. Mauro, J. Setiadi, X. Baca, Z. Shi, J. Calder, and A. L. Bertozzi, *Graph-based Active Learning for Semi-Supervised Classification of SAR Data*, https://arxiv.org/abs/2204.00005, 2022.
[36] L. Modica and S. Mortola, *Un esempio di Γ^--convergenza, Boll. Unione Mat. Ital.* Ser. V B, 14 (1977), pp. 285–299.
[37] D. Mumford and J. Shah, *Optimal approximations by piecewise smooth functions and associated variational problems*, Comm. Pure Appl. Math., 42 (1989), pp. 577–685, https://doi.org/10.1002/cpa.3160420503.
[38] Y. Mäkinen, L. Azzari, and A. Foi, *Collaborative filtering of correlated noise: Exact transform-domain variance for improved shrinkage and patch matching*, IEEE Trans. Image Process., 29 (2020), pp. 8339–8354, https://doi.org/10.1109/TIP.2020.3014721.
[39] E. J. Nyström, *Über die praktische auflösung von Integralgleichungen mit anwendungen auf randwertaufgaben*, Acta Math., 54 (1930), pp. 185–204, https://doi.org/10.1007/BF02547521.
[40] D. L. Phillips, *A technique for the numerical solution of certain integral equations of the first kind*, J. ACM, 9 (1962), pp. 84–97.
[41] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes: The Art of Scientific Computing*, 3rd ed., Cambridge University Press, New York, 2007.
[42] Y. Qiao, C. Shi, C. Wang, H. Li, M. Haberland, X. Luo, A. M. Stuart, and A. L. Bertozzi, *Uncertainty quantification for semi-supervised multi-class classification in image processing and ego-motion analysis of body-worn videos*, Electron. Imaging, 2019 (2019), pp. 264-1–264-7, https://doi.org/10.2352/ISSN.2470-1173.2019.11.IPAS-264.
[43] R. Ramlau and W. Ring, *A Mumford-Shah level-set approach for the inversion and segmentation of X-ray tomography data*, J. Comput. Phys., 221 (2007), pp. 539–557.
[44] L. I. Rudin, S. Osher, and E. Fatemi, *Nonlinear total variation based noise removal algorithms*, Phys. D, 60 (1992), pp. 259–268, https://doi.org/10.1016/0167-2789(92)90242-F.
[45] G. Strang, *On the construction and comparison of difference schemes*, SIAM J. Numer. Anal., 5 (1968), pp. 506–517, http://www.jstor.org/stable/2949700.
[46] A. N. Tikhonov, *Solution of incorrectly formulated problems and the regularization method*, Soviet Math. Dokl., 4 (1963), pp. 1035–1038.
[47] Y. van Gennip and A. L. Bertozzi, *Gamma-Convergence of Graph Ginzburg–Landau Functionals*, https://arxiv.org/abs/1204.5220, 2018.
[48] Y. van Gennip, N. Guillen, B. Osting, and A. Bertozzi, *Mean Curvature, Threshold Dynamics, and Phase Field Theory on Finite Graphs*, Milan J. Math., 82 (2014), pp. 3–65.
[49] M. Varma and A. Zisserman, *A statistical approach to texture classification from single images*, Int. J. Comput. Vis., 62 (2005), pp. 61–81, https://doi.org/10.1007/s11263-005-4635-4.
[50] S. V. Venkatakrishnan, C. A. Bouman, and B. Wohlberg, *Plug-and-Play priors for model based reconstruction*, in Proceedings of the IEEE Global Conference on Signal and Information Processing, 2013, pp. 945–948, https://doi.org/10.1109/GlobalSIP.2013.6737048.

[51] K. Zhang, H. Song, and L. Zhang, *Active contours driven by local image fitting energy*, Pattern Recognit., 43 (2010), pp. 1199–1206.

[52] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang, *Beyond a gaussian denoiser: Residual learning of deep CNN for image denoising*, IEEE Trans. Image Process., 26 (2017), pp. 3142–3155, https://doi.org/10.1109/TIP.2017.2662206.