Published in Image Processing On Line on 2026–04–10.
Submitted on 2024–09–16, accepted on 2025–12–26.
ISSN 2105–1232 © 2026 IPOL & the authors CC–BY–NC–SA
This article is available online with supplementary materials,
software, datasets and online demo at
https://doi.org/10.5201/ipol.2026.578

An Implementation of Two-Phase Image Segmentation Using the Split Bregman Method

Olakunle Abawonse[1], Günay Doğan[2]

[1]African Institute of Mathematical Sciences (AIMS), Rwanda (olakunle.abawonse@aims.ac.rw)
[2]National Institute of Standards and Technology, Gaithersburg, MD, USA (gunay.dogan@nist.gov)

Communicated by Enric Meinhardt-Llopis, Axel Davy, and Marina Gardella *Demo edited by* Marina Gardella

Abstract

In this paper, we describe an implementation of the two-phase image segmentation algorithm proposed by Goldstein, Bresson and Osher in [Geometric Applications of the Split Bregman Method: Segmentation and Surface Reconstruction, Journal of Scientific Computing, 2010]. This algorithm partitions the domain of a given 2D image into foreground and background regions, and each pixel of the image is assigned membership to one of these two regions. The underlying assumption for the segmentation model is that the pixel values of the input image can be summarized by two distinct average values, and that the region boundaries are smooth. Accordingly, the model is formulated as an energy functional whose variable is a region membership function that assigns pixels to either region, as originally proposed by Chan and Vese in [Active Contours Without Edges, IEEE Transactions on Image Processing, 2001]. This energy is the sum of image data terms in the regions and a length penalty for region boundaries. Goldstein, Bresson and Osher modify the energy of Chan-Vese so that their new energy can be minimized efficiently using the split Bregman method to produce an equivalent two-phase segmentation. We provide a detailed implementation of this method, and document its performance with several images over a range of algorithm parameters.

Source Code

The reviewed source code and documentation for this algorithm are available at the IPOL web page of this article[1]. Compilation and usage instructions are included in the `README.txt` file of the archive.

Keywords: image segmentation; Bregman iterations; convex optimization

[1]https://doi.org/10.5201/ipol.2026.578

1 The Segmentation Model

Image segmentation is the problem of partitioning a given image into distinct regions, each of which is homogeneous with respect to image pixel values or image features. This problem is intrinsically hard as the image pixels often include noise or other local variability due to texture or other factors. A seminal mathematical model for achieving segmentation in such situations is the Mumford-Shah model proposed in [13]. Given a gray-scale image $f : \Omega \to \mathbb{R}$ defined on the image domain $\Omega \subset \mathbb{R}^2$, a segmentation and smoothing are computed by minimizing the following model

$$E_{MS}(\Sigma, u) = \lambda_\Sigma |\Sigma| + \int_\Omega (u(x) - f(x))^2 dx + \mu \int_{\Omega - \Sigma} |\nabla u|^2 dx, \quad \mu, \lambda_\Sigma > 0, \tag{1}$$

with respect to the set of discontinuities or edges Σ in the image, and a piecewise smooth approximation $u(x)$ to the image function $f(x)$. The minimizers Σ^* and u^* of (1) are a segmentation and a smoothing/denoising of f respectively. The edge length penalty $|\Sigma|$ in (1) encourages smooth curves of discontinuity in the solution, and the integral of $|\nabla u|^2$ penalizes variations in u. The model (1), however, is hard to minimize computationally.

A more tractable model can be obtained by assuming that the image f can be approximated by a piecewise constant function instead of a piecewise smooth function, as observed in many images. This assumption is the basis of the Chan and Vese segmentation model [4], which aims to segment the domain $\Omega \subset \mathbb{R}^2$ of the image f into a foreground region Ω_1 and a background region $\Omega_2 = \Omega \setminus \Omega_1$. The image values in Ω_1, Ω_2 can be approximated by averages c_1, c_2 respectively. Note, however, that more regions, or other image value models, such as piecewise affine linear functions, can also be assumed in (1) [11]. The so-called two-phase segmentation problem of Chan and Vese [4, 5] can be formulated as the minimization of the energy

$$E_2(\Omega_1, \Omega_2, c_1, c_2) := \mathrm{Per}(\Omega_1) + \lambda \int_{\Omega_1} (f(x) - c_1)^2 dx + \lambda \int_{\Omega_2} (f(x) - c_2)^2 dx, \ \lambda > 0, \tag{2}$$

where $\mathrm{Per}(\Omega_1)$ is the perimeter of Ω_1, i.e. the length of the boundary between Ω_1 and Ω_2. It is included in the energy (2) as a regularization term to encourage smoothness of the region boundaries. The second and third terms in (2) are the data fidelity terms to ensure that the regions Ω_1, Ω_2 match the perceived regions of the image, each approximated by the corresponding averages c_1, c_2 of pixel values. In the absence of the regularization term $\mathrm{Per}(\Omega_1)$, the segmentation will be noisy, as is typically the case when the image pixels are thresholded into two sets, for example, by using the Otsu thresholding method for segmentation [15, 1].

The optimality of (2) with respect to c_1, c_2 implies $c_1 = \frac{1}{|\Omega_1|} \int_{\Omega_1} f(x)dx, c_2 = \frac{1}{|\Omega_2|} \int_{\Omega_2} f(x)dx$. Thus the energy (2) can be reduced to an energy in terms of Ω_1, Ω_2 only. This results in a region optimization problem. Chan and Vese [4] encode the regions in a level set function $\phi(x) : \Omega \to \mathbb{R}$:

$$\Omega_1 = \{x : \phi(x) < 0\}, \qquad \Omega_2 = \{x : \phi(x) > 0\},$$

and rewrite the energy in terms of the level set function

$$E_{CV}(\phi) = \int_\Omega |\nabla H_\epsilon(\phi)| dx + \lambda \int_\Omega H_\epsilon(\phi)(f(x) - c_1)^2 + (1 - H_\epsilon(\phi))(f(x) - c_2)^2 dx, \tag{3}$$

in which H_ϵ is a regularized Heaviside function. The first integral in (3) approximates the perimeter term in (2), and the second integral approximates the data fidelity terms in (2). A minimizer of (3) is computed by starting from a given initial level set function $\phi_0(x)$ and using the following time-dependent partial differential equation (PDE) to evolve $\phi(x)$,

$$\phi_t = H'_\epsilon(\phi) \left(\mathrm{div}(\frac{\nabla \phi}{|\nabla \phi|}) - \lambda((f(x) - c_1)^2 - (f(x) - c_2)^2) \right). \tag{4}$$

Equation (4) is a gradient descent evolution for the energy (3). A stable and efficient numerical discretization and implementation of the PDE (4) requires care, and computing the solution can be slow.

Chan, Esedoglu and Nikolova note in [3] that the following evolution equation

$$\frac{\partial u}{\partial t} = \operatorname{div}(\frac{\nabla u}{|\nabla u|}) - \lambda((f(x) - c_1)^2 - (f(x) - c_2)^2), \tag{5}$$

has the same stationary points as Equation (4), and it can be used to compute a solution $\Omega_1 = \{x : u(x) < \mu\}$, $\Omega_2 = \{x : u(x) > \mu\}$ ($\mu > 0$ is some threshold) to the two-phase segmentation problem. Equation (5) is the gradient descent evolution for the following energy

$$E_0(u) = \|u\|_{TV(\Omega)} + \lambda \int_\Omega u(x)(f(x) - c_1)^2 + (1 - u(x))(f(x) - c_2)^2 dx, \quad \lambda > 0, \tag{6}$$

where $u(x) \in BV(\Omega)$ acts as a region indicator function, and $\|u\|_{TV(\Omega)}$, the total variation of u, is a regularization term, analogous to $\mathrm{Per}(\Omega_1)$ in (2), to attain regions with smooth boundaries. The seminorm $\|u\|_{TV(\Omega)}$ is defined as

$$\|u\|_{TV(\Omega)} := \int_\Omega |Du| := \sup \left\{ \int_\Omega u \operatorname{div} p \, dx : p \in C_c^1(\Omega, \mathbb{R}^2), |p(x)| \leq 1 \right\}. \tag{7}$$

The total variation of a function u was introduced in image processing by Rudin, Osher and Fatemi in [16] to denoise images. For this, they evolved a noisy image f using the total variation PDE, thereby reducing the total variation and noise content of f. In [2], Chambolle proposed a first order minimization algorithm for total variation minimization through nonlinear projections to convex sets in dual space. In [10], Goldstein et al. proposed to use Bregman iterations to solve total variation image denoising. The implementation of the algorithm from [10] was described by Getreuer in [6].

The following theorem connects the minimizer u of the energy (6) with the solution of the two-phase segmentation problem and the energy (2). Note that $\int_\Omega (f(x) - c_2)^2 dx$ in (6) is constant for fixed c_1, c_2, and can be disregarded for this result.

Theorem 1. *[3] For any given fixed $c_1, c_2 \in \mathbb{R}$, a global minimizer for energy (2) is obtained by solving the convex minimization*

$$\min_{0 \leq u \leq 1} \int_\Omega |\nabla u| dx + \lambda \int_\Omega \left\{ (f(x) - c_1)^2 - (f(x) - c_2)^2 \right\} u(x) dx, \tag{8}$$

and then setting $\Omega_1 = \{x : u(x) \geq \mu\}$ for a.e. $\mu \in [0, 1]$.

The above theorem suggests that we can take Equation (6) (excluding the constant-valued $\int_\Omega (f(x) - c_2)^2 dx$ term) as the energy to be minimized and restrict our solution space to $\{u \in BV(\Omega) : 0 \leq u(x) \leq 1\}$.

Chan, Esedoglu and Nikolova observe in [3] that using a weighted total variation (TV_g) seminorm instead of $\|u\|_{TV(\Omega)}$ in (7) yields better segmentations as the TV_g seminorm suppresses diffusion across region boundaries better. The weighted total variation seminorm $\|u\|_{TV_g(\Omega)}$ of u is defined as

$$\|u\|_{TV_g(\Omega)} := \int_\Omega |Du|_g := \sup \left\{ \int_\Omega u \operatorname{div} p \, dx : p \in C_c^1(\Omega, \mathbb{R}^2), |p(x)| \leq g(x) \right\}, \tag{9}$$

where $g(x)$ is a weight function defined using the image function $f(x)$. It is chosen so that it has a small value close to region boundaries or edges in the image, and a value close to 1 away from the edges. The following is a commonly used choice for $g(x)$, which we used in our implementation

$$g(x) = 1 \,/\, (1 + |(\nabla G_\sigma * f)(x)|^2 / \rho^2), \ \rho > 0, \tag{10}$$

where $G_\sigma = \frac{1}{2\pi\sigma^2}\exp\left(-\frac{|x|^2}{2\sigma^2}\right)$ is the Gaussian convolution kernel with standard deviation σ, used to compute smoother derivatives of $f(x)$ to mark the edges in the image. Including weighted total variation (9) as a regularization in (6) and assuming known averages c_1, c_2 leads to the following energy that we minimize to obtain u

$$E_g(u) = \int_\Omega g(x)|\nabla u|dx + \lambda \int_\Omega ((f(x)-c_1)^2 - (f(x)-c_2)^2)u(x)dx. \tag{11}$$

Given the energy (11), we can consider an alternating optimization approach for the more general case of unknown region averages c_1, c_2. We first estimate c_1, c_2 given u, then minimize (11) with respect to u, and continue to alternate between these two steps until convergence. In [7], Goldstein et al. proposed the split Bregman method to solve the minimization problem for (11), which we review in the following sections. Moreno et al. used the same approach for a four-phase segmentation in [12].

2 Minimization with Bregman Iterations

The minimizer u of (11) can be computed using Bregman iterations [7]. Thus, in this section, we give a brief introduction to the Bregman method. A detailed explanation can be found in [14]. Suppose J and H are (possibly non-differentiable) convex functionals defined on a Hilbert space X. The Bregman iterations can be used to solve a convex minimization problem of the form

$$\min_{u\in B} J(u) + \lambda H(u),\ \lambda > 0. \tag{12}$$

In order to state the iterative minimization algorithm for (12), we will need to introduce the notions of subgradient and Bregman distance.

Definition 1. *Suppose $J : X \to \mathbb{R}$ is a convex function and $u \in X$. An element $p \in X^*$ is called a subgradient of J at v if for all $u \in X$*

$$J(u) - J(v) - \langle p, u - v\rangle \geq 0.$$

The set of all subgradients of J at v is called the subdifferential of J at v, and it is denoted by $\partial J(v)$.

Definition 2. *Suppose $J : X \to \mathbb{R}$ is a convex function, $u, v \in X$ and $p \in \partial J(v)$. Then the Bregman distance between points u and v is defined by*

$$D_J^p(u, v) := J(u) - J(v) - \langle p, u - v\rangle.$$

We have the following basic distance-like properties of the Bregman distance:

(i) $D_J^p(v, v) = 0$,

(ii) $D_J^p(u, v) \geq 0$,

(iii) $D_J^p(u, v) \geq D_J^p(w, v)$ for w on the line segment from u to v.

Property (iii) can be shown by setting $w = \alpha u + (1-\alpha)v$ on the line segment between u and v.

$$D_J^p(w, v) = J(w) - J(v) - \langle p, \alpha u + (1-\alpha)v - v\rangle = J(w) - J(v) - \alpha\langle p, (u - v)\rangle,$$

$$\begin{aligned}\text{so that}\quad D_J^p(u, v) - D_J^p(w, v) &= J(u) - J(w) - \langle p, u - v\rangle + \alpha\langle p, u - v\rangle \\ &\geq J(u) - (\alpha J(u) + (1-\alpha)J(v)) - (1-\alpha)\langle p, u - v\rangle \\ &= (1-\alpha)(J(u) - J(v) - \langle p, u - v\rangle) = (1-\alpha)D_J^p(u, v) \geq 0.\end{aligned}$$

The Bregman distance can be used as part of Bregman iterations to solve the minimization problem (12). Osher et al. defined the Bregman iterations as follows in [14]

$$u^{k+1} = \underset{u}{\operatorname{argmin}}\, D_J^{p^k}(u, u^k) + \lambda H(u),\ p^k \in \partial J(u^k). \tag{13}$$

When H is differentiable, for u^{k+1} minimizing $D_J^{p^k}(u, u^k) + \lambda H(u)$, we have $p^k - \lambda \nabla H(u^{k+1}) \in \partial J(u^{k+1})$, so we can take $p^{k+1} = p^k - \lambda \nabla H(u^{k+1})$. Thus, Bregman iterations for differentiable H are summarized in Algorithm 1.

Algorithm 1: Bregman iterations for differentiable H

input : u^0 – initial estimate
output: (u^k, p^k) – Bregman iterates
Initialize $p^0 \in \partial J(u^0)$
for $k = 0, 1, 2, \ldots$ **do**
  $u^{k+1} = \operatorname{argmin}_u\ D_J^{p^k}(u, u^k) + \lambda H(u)$
  $p^{k+1} = p^k - \lambda \nabla H(u^{k+1})$

Theorem 2. *[14] Assume J and H are convex functionals defined on $BV(\Omega)$, and H is differentiable. Then we have $H(u_{k+1}) \le H(u_k)$. Moreover, assume there exists a minimizer $u^* \in BV(\Omega)$ of $H(\cdot)$ with $J(u^*) < \infty$. Then*

$$D_J^{p_k}(u^*, u_k) \le D_J^{p_{k-1}}(u^*, u_{k-1}) \ \text{ and } \ H(u_k) \le H(u^*) + \frac{1}{k} J(u^*).$$

In particular, u_k is a minimizing sequence of Algorithm 1 under the above assumptions.

For the segmentation problem (11) that we will consider, we are interested in Bregman iterations in the case of linear constraints $Au = z$, corresponding to the case of $H(u) = \frac{1}{2}\|Au - z\|^2$, namely,

$$\min_u J(u) + \frac{\lambda}{2}\|Au - z\|^2,$$

which can be solved using Algorithm 1 as follows

$$u^{k+1} = \underset{u}{\operatorname{argmin}}\, D_J^{p^k}(u, u^k) + \frac{\lambda}{2}\|Au - z\|^2 = \underset{u}{\operatorname{argmin}}\, J(u) - \langle p^k, u - u^k\rangle + \frac{\lambda}{2}\|Au - z\|^2,$$
$$p^{k+1} = p^k - \lambda A^T(Au^{k+1} - z).$$

Existence of b_k from $p_k = \lambda A^* b_k$ allows us to write a more compact form of update iterations. The above iterations were shown in [14] to be equivalent to Algorithm 2 when A is linear.

Now we describe how we build on Bregman iterations to minimize the segmentation energy (11). This approach was proposed in [7]. We first introduce an auxiliary variable $d := \nabla u$ resulting in the constrained minimization problem

$$\tilde{E}(u, d) = \int_\Omega g(x)|d|dx + \lambda \int_\Omega ((f(x) - c_1)^2 - (f(x) - c_2)^2)u(x)dx \qquad \text{s.t.} \quad d = \nabla u.$$

Then we write the equivalent augmented Lagrangian

$$\int_\Omega g(x)|d|dx + \lambda \int_\Omega ((f(x) - c_1)^2 - (f(x) - c_2)^2)u(x)dx + \frac{\gamma}{2}\|d - \nabla u - b\|^2,\ \gamma > 0, \tag{14}$$

which can be minimized with the customized Bregman iterations in Algorithm 3.

We will describe how to solve the minimization subproblem in Algorithm 3 in the next section.

Algorithm 2: Bregman iterations for $H(u) = \frac{1}{2}\|Au - z\|^2$

input : z – given data, λ – regularization parameter
output: (u^k, b^k) – Bregman iterates
Initialize u^0, $b^0 = 0$
for $k = 0, 1, 2, \ldots$ **do**
  $u^{k+1} = \operatorname{argmin}_u \; J(u) + \frac{\lambda}{2}\|Au - z - b^k\|^2$
  $b^{k+1} = b^k + z - Au^k$

Algorithm 3: Bregman iterations for energy (14)

input : f, g, c_1, c_2, λ, γ
output: (u^k, d^k, b^k) – Bregman iterates
Initialize $u^0 = 0$, $d^0 = 0$, $b^0 = 0$
for $k = 0, 1, 2, \ldots$ **do**
  $(u^{k+1}, d^{k+1}) = \underset{0\le u(x)\le 1,\, d}{\operatorname{argmin}} \int_\Omega g(x)|d| + \lambda \int_\Omega \big((f(x)-c_1)^2 - (f(x)-c_2)^2\big)u(x) + \frac{\gamma}{2}\|d - \nabla u - b^k\|^2$
  $b^{k+1} = b^k + \nabla u^{k+1} - d^{k+1}$

3 Discrete Model and Minimization

Although the image processing models and the Bregman minimization algorithm to solve these models are expressed in function spaces, images are usually given as 2D arrays of pixels. This necessitates working with discretized versions of these models. Thus we start this section with a discretization of the differential operators. Then we describe the discretized Bregman algorithms for two-phase image segmentation.

3.1 Discrete Derivatives

In this section, we will define the discrete analogues of gradient, divergence and Laplacian operators on uniformly-spaced image grid $(u_{i,j}), (f_{i,j}), (g_{i,j}), (\vec{w}_{i,j}), i = 0, \ldots, n_x, j = 0, \ldots, n_y$, of bounded functions $u, f, g, \vec{w}$. The discrete first derivatives of $f_{i,j}$ are defined as

$$\partial_x f_{i,j} := \begin{cases} f_{i+1,j} - f_{i,j}, & i = 0, \ldots, n_x - 1, \\ 0, & i = n_x, \end{cases} \qquad \partial_y f_{i,j} := \begin{cases} f_{i,j+1} - f_{i,j}, & j = 0, \ldots, n_y - 1, \\ 0, & j = n_y. \end{cases}$$

The discrete gradient of $u_{i,j}$ is defined as $\nabla u_{i,j} := \begin{pmatrix} \partial_x u_{i,j} \\ \partial_y u_{i,j} \end{pmatrix}$. The discrete adjoint gradient of $g_{i,j}$ is $-\nabla^* g_{i,j} := \begin{pmatrix} -\partial_x^* g_{i,j} \\ -\partial_y^* g_{i,j} \end{pmatrix}$, where

$$-\partial_x^* g_{i,j} = \begin{cases} g_{0,j}, & i = 0, \\ g_{i,j} - g_{i-1,j}, & 1 \le i \le n-1, \\ -g_{n-1,j}, & i = n, \end{cases}$$

and $-\partial_y^* g_{i,j}$ is defined similarly. For $\vec{w}_{i,j} = (w^x_{i,j}, w^y_{i,j})^T$, we have $\operatorname{div} \vec{w}_{i,j} = -\partial_x^* w^x_{i,j} - \partial_y^* w^y_{i,j}$. Finally, we define the discrete Laplacian as $\Delta u_{i,j} = -\partial_x^* \partial_x u_{i,j} - \partial_y^* \partial_y u_{i,j}$.

3.2 The Discretized Segmentation Model

In this section, we introduce the discretized version of the segmentation energy (11) for the given image data $\{f_{ij}\}$, and we describe the minimization algorithm for the discretized energy. We discretize

the segmentation energy (11) as follows

$$E(u) = \sum_{i,j} g_{i,j}|\nabla u_{i,j}|dx + \lambda \sum_{i,j}((f_{i,j} - c_1)^2 - (f_{i,j} - c_2)^2)u_{i,j}. \tag{15}$$

As described in Section 3, we introduce the auxiliary variable $d_{i,j} \approx \nabla u_{i,j}$, and consider the equivalent augmented Lagrangian. Then the key step in the Bregman iterations is the following minimization problem

$$\underset{0\leq u\leq 1,d}{\operatorname{argmin}} \sum_{i,j} g_{i,j}|d_{i,j}| + \lambda \sum_{i,j}((f_{i,j} - c_1)^2 - (f_{i,j} - c_2)^2)u_{i,j} + \frac{\gamma}{2}\sum_{i,j}(d_{i,j} - \nabla u_{i,j} - b_{i,j})^2,\ \gamma > 0. \tag{16}$$

To minimize the discretized augmented Lagrangian (16), we use the alternating direction method, namely, we minimize first over d, then over u while keeping the other variable fixed. We iterate in this manner until convergence.

The d subproblem: With u fixed, the d subproblem is

$$\underset{d}{\operatorname{argmin}} \sum_{i,j} g_{i,j}|d_{i,j}| + \frac{\gamma}{2}\sum_{i,j}(d_{i,j} - \nabla u_{i,j} - b_{i,j})^2. \tag{17}$$

The solution to the d subproblem obtained in [7] is given by

$$d_{i,j} = \frac{\nabla u_{i,j} + b_{i,j}}{|\nabla u_{i,j} + b_{i,j}|}\max\{|\nabla u_{i,j} + b_{i,j}| - \frac{g_{i,j}}{\gamma}, 0\}. \tag{18}$$

The u subproblem: With d fixed, the u subproblem is

$$\underset{0\leq u\leq 1}{\operatorname{argmin}}\, \lambda \sum_{i,j}((f_{i,j} - c_1)^2 - (f_{i,j} - c_2)^2)u_{i,j} + \frac{\gamma}{2}\sum_{i,j}(\nabla u_{i,j} - d_{i,j} + b_{i,j})^2. \tag{19}$$

The optimal u satisfies

$$\lambda r_{i,j} + \gamma\nabla^*(\nabla u_{i,j} - d_{i,j} + b_{i,j}) = 0, \tag{20}$$

where $r_{i,j} = (f_{i,j} - c_1)^2 - (f_{i,j} - c_2)^2$. We rewrite (20) as the following elliptic PDE

$$\Delta u_{i,j} = \frac{\lambda}{\gamma} r_{i,j} + \operatorname{div}(d_{i,j} - b_{i,j}), \tag{21}$$

which can be solved with a variety of PDE solvers, including Jacobi iterations. We do not, however, take Jacobi iterations to fully solve the PDE at each Bregman iteration. Instead, we execute an approximate solve with a single Jacobi iteration

$$\beta^k_{i,j} = \frac{1}{4}(u^k_{i-1,j} + u^k_{i+1,j} + u^k_{i,j-1} + u^k_{i,j+1} - \frac{\lambda}{\gamma} r_{i,j} + \operatorname{div}(d_{i,j} - b_{i,j})) \text{ for } 1 \leq i < n_x, 1 \leq j < n_y.$$

Then we impose the bounds $0 \leq u \leq 1$ by

$$u^{k+1}_{i,j} = \max\{\min\{\beta^k_{i,j}, 1\}, 0\}. \tag{22}$$

Updating averages c_1, c_2: The region averages c_1, c_2 need to be recomputed at each iteration as the regions defined by $u_{i,j}$ change. In the discrete case, the regions are given by the sets of indices marking the spatial locations on the pixel grid

$$\Omega_1^k = \{(i,j) : u_{i,j}^k \geq 0.5\}, \qquad \Omega_2^k = \{(i,j) : u_{i,j}^k \leq 0.5\}. \tag{23}$$

The areas $|\Omega_1^k|, |\Omega_2^k|$ of the two regions Ω_1^k, Ω_2^k are the cardinalities of the sets (23), namely, the counts of the indices contained in these two sets. Then the update formulas for c_1^k, c_2^k are

$$c_1^k = \frac{1}{|\Omega_1^k|} \sum_{(i,j)\in\Omega_1^k} f_{i,j}, \qquad c_2^k = \frac{1}{|\Omega_2^k|} \sum_{(i,j)\in\Omega_2^k} f_{i,j}. \tag{24}$$

3.3 More on Iterative Minimization

Now we elaborate some other important pieces to implement an efficient and robust iterative algorithm for segmentation. As in most iterative algorithms, we aim to ensure fast and consistent convergence to a minimizer of (16), or ideally of the energy (15). The value $E(u)$ serves as a measure of the quality of the segmentation encoded by u; the smaller$E(u)$, the better the segmentation. Accordingly, we track the energy values $E^k = E(u^k)$ through the iterations, continue as long as the E^k values are decreasing, and stop if the decrease between iterations becomes too small. Thus, our stopping criterion is

$$\|E^k - \bar{E}^{k-1}\| < \varepsilon_{tol} E^0, \tag{25}$$

where $\bar{E}^{k-1}$ is the average of the previous m energy values: $\bar{E}^{k-1} = \frac{1}{m}\sum_{l=k-m}^{k-1} E(u^l)$. We use the recent average $\bar{E}^{k-1}$ instead of E^{k-1} to introduce some robustness to small fluctuations in the energy computation from noise in the image, and to avoid premature stopping of the iterations because of this. Since $\bar{E}^{k-1}$ needs m energy values to compute the average, we always start with $m+1$ iterations (we set $m = 10$ in our experiments).

We have also found it helpful to tune and dampen the updates $b^{k+1} = b^k + \nabla u^{k+1} - d^{k+1}$ in Algorithm 3. A variable step size was employed in [8] to adjust the pace of the updates, and an upper bound was imposed on the step size in [9] to ensure convergence of the algorithm. We found the use of a fixed step size τ to be practical and effective, and executed the updates to b^{k+1} by

$$b^{k+1} = b^k + \tau(\nabla u^{k+1} - d^{k+1}). \tag{26}$$

We experimented with various step sizes τ, and picked a conservative value $\tau = 0.01$. However, values in the range $[0.001, 0.1]$ seemed to work well. Smaller step sizes resulted in more iterations, and larger step sizes led to a reduced number of iterations, but risked overshooting the minimum of the energy.

Finally, a good initialization is an important factor in achieving computational efficiency. Different works in the literature have used seeds or sets of shapes like circles as initialization [4]. A more effective strategy could involve a pre-clustering of the pixels based on their grayscale values into two clusters. In our experiments, we found the image itself to be a good initialization for the iterations. We normalized the grayscale image to the range $[f_{min}, f_{max}]$, the minimum and maximum values of f respectively, and initialized u^0 by

$$u^0 = (f - f_{min})/(f_{max} - f_{min}). \tag{27}$$

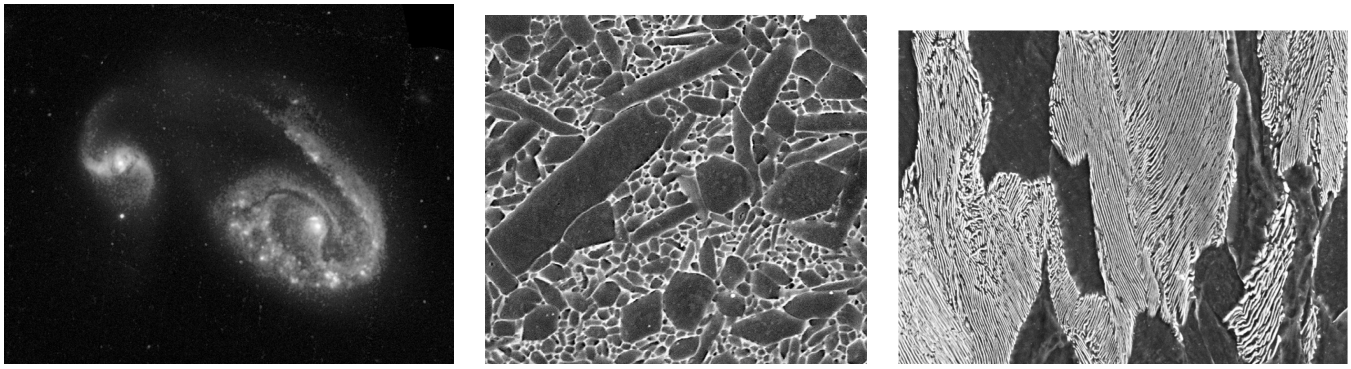
Incorporating the initialization, the stopping criterion and the step size gave us the final iterative algorithm for two-phase image segmentation, shown in Algorithm 4.

Algorithm 4: Minimization of discretized segmentation model

input : m, ε_{tol}, τ, f
output: u
Initialize u^0 by (27), $E^0 = E(u^0)$, $b^0 = 0$, $d^0 = 0$, $k = 0$
while $(k < m)$ ***or*** $\left(\|E^k - \bar{E}^{k-1}\| > \varepsilon_{tol} E^0\right)$ **do**
  Solve the u subproblem to obtain u^{k+1}
  Solve the d subproblem to obtain d^{k+1}
  $\Omega_1^{k+1} = \{(i,j) : u_{i,j}^{k+1} \geq 0.5\}$
  $\Omega_2^{k+1} = \{(i,j) : u_{i,j}^{k+1} < 0.5\}$
  $c_1^{k+1} = \dfrac{1}{|\Omega_1^{k+1}|} \sum_{(i,j)\in\Omega_1^{k+1}} f_{i,j}$
  $c_2^{k+1} = \dfrac{1}{|\Omega_2^{k+1}|} \sum_{(i,j)\in\Omega_2^{k+1}} f_{i,j}$
  $b^{k+1} = b^k + \tau\left(\nabla u^{k+1} - d^{k+1}\right)$
  $E^{k+1} = E(u^{k+1})$
  $\bar{E}^k = \dfrac{1}{m} \sum_{l=k-m+1}^{k} E^l$
  $k = k + 1$

4 Experiments

We implemented Algorithm 4 in Python, and tested it with experiments on real images. We used three test images: galaxy (image of two galaxies), microstructure (material microstructure image with voids), and pearlite (material microstructure: micrograph of etched pearlite), shown in Figure 1. Leveraging vectorized operations of the NumPy package in Python, we were able to code components of Algorithm 4 in a concise, easily-readable manner, which was also efficient for computation. We ran our experiments on a laptop computer with Intel™ Core i7 10th Gen processor and 32GB of memory. We used the same algorithm parameters $\gamma = 0.1, \tau = 0.01, m = 10, \varepsilon_{tol} = 10^{-4}$ for all the experiments, but varied the model parameter λ to illustrate its effect on the regularity of the solution. In the rest of this section, we describe the results of our experiments and our observations.



galaxy, 640×800 microstructure, 548×600 pearlite, 700×944

Figure 1: Images used in the Experiments section

Observing the iterations. We observed the evolution of the function u and the change in the energy for the three example images. In Figure 2, we illustrate the evolution of the segmentation $\{u(x) > 0.5\}$ by taking some snapshots from intermediate iterations of minimization as it converges towards the optimal solution. Looking closely at the galaxy example, we observe that the function u is still very close to the initial value f at iteration $k = 5$, and its energy is still high as seen in Figure 3. After 20 iterations, the function u starts to take the shape of our optimal solution. This can be observed from the energy plot as well. The energy drops sharply after about 30 to 50 more iterations (see first row of Figure 3).

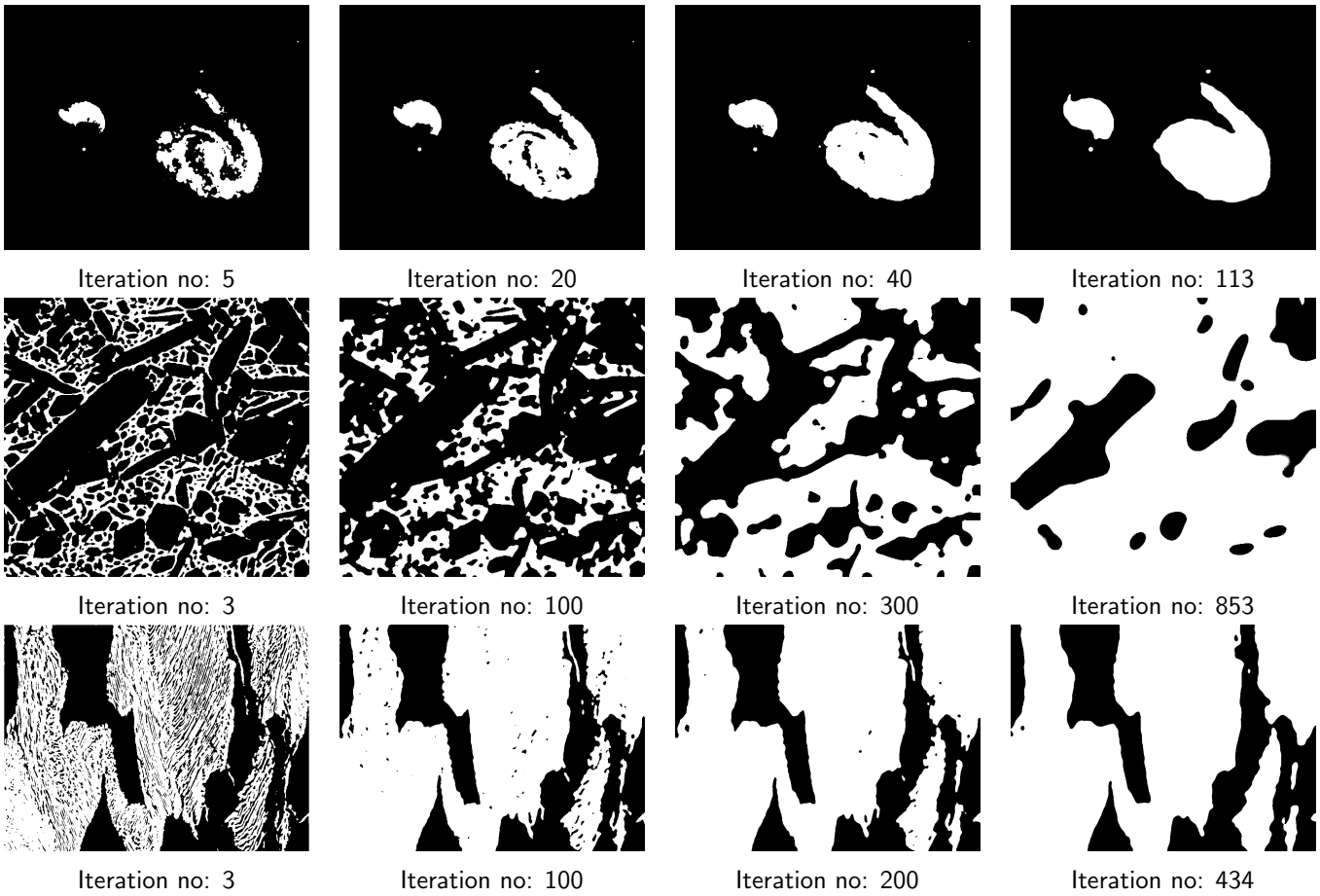


Figure 2: Snapshots of the evolving segmentation $\{u(x) > 0.5\}$ from intermediate iterations.

After the initial rapid progress, the updates between subsequent iterations are smaller. For example, in the evolution of u for the galaxy image, the changes between the last few hundred iterations are small. This is also observed in some of the other energy plots in Figure 3 where we see in some cases the energy steadily decreasing at a very slow rate. The same behavior can be observed for the other test images as shown in Figure 2 and Figure 3.

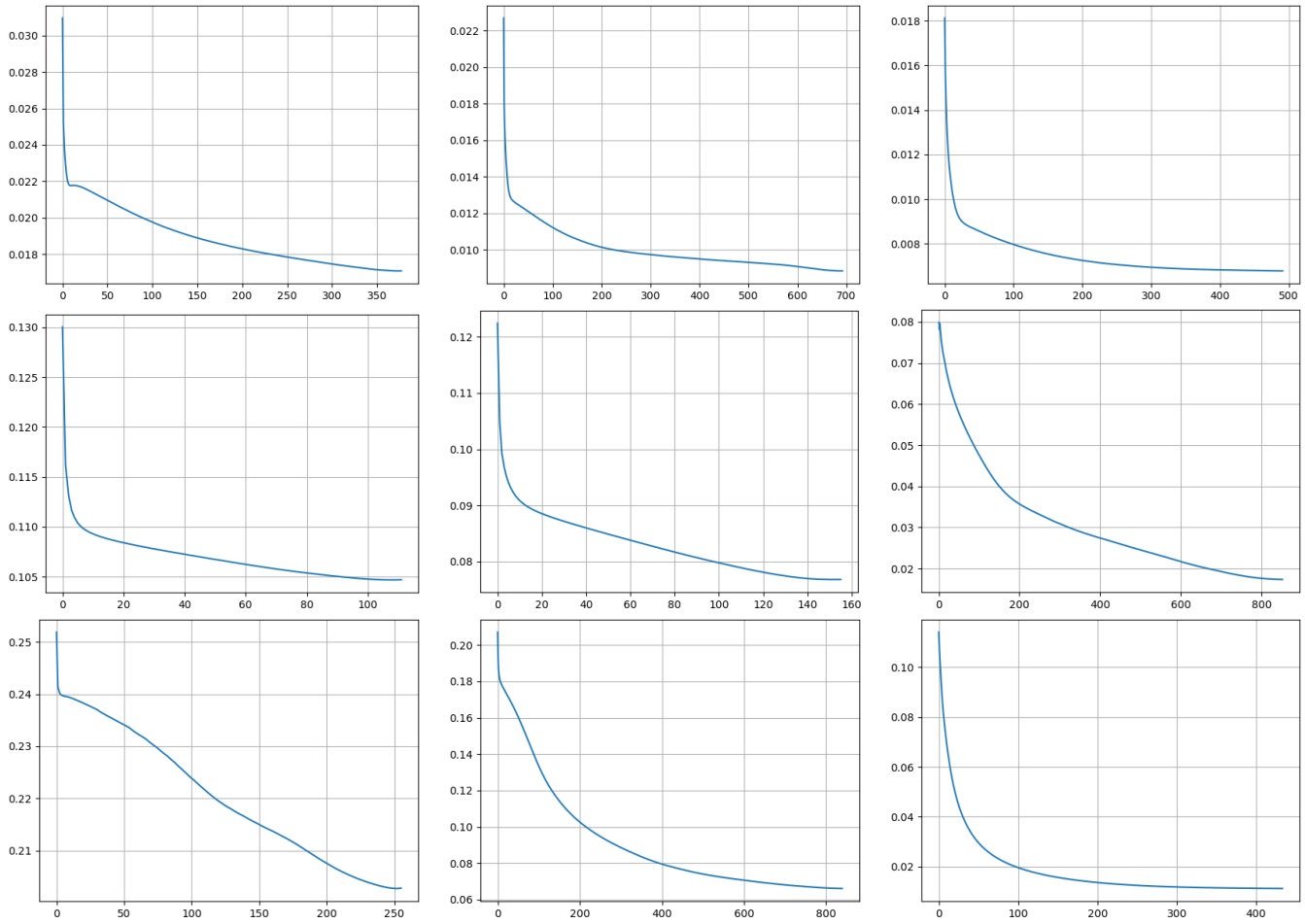


Figure 3: Evolution of the energy values for galaxy (top row), microstructure (middle row), and pearlite (bottom row) images in the cases of low (left column), medium (middle column) and high (right column) regularization.

The effect of data weight. We also considered the effects of the data weights λ on the segmentation results (see Figure 4). For a higher data weight λ, we observed lower regularization in segmentation. In this situation, the regions had rougher boundaries, and the segmentation seemed less smooth. We then considered a medium value of the data weight, and we observed a segmentation with better regularization than in our previous observation, but still with some rough boundaries. We then gradually decreased the data weight λ, and this resulted in increased regularization and smoother region boundaries. If we decreased λ very much, then the smoothing was excessive and destroyed many details in the segmentation, as shown in the middle row of Figure 4 for data weight $\lambda = 0.1$. However, the same data weight resulted in a desirable segmentation for the pearlite image (bottom row of Figure 4), as the strong smoothing closed the gaps in the bright regions.

We did not observe a direct correlation between the value λ and the number of iterations or computation time. The number of iterations in each experiment rather reflected the amount of work needed to reach the minimum of the energy, depending on how far the initial iterate u^0 was from the final segmentation. The details of the data weight experiments for the three test images are shown in Figure 4.

Comparisons with other algorithms. Finally, we compared the segmentations from our implementation to those obtained using two closely related algorithms: Otsu thresholding [15, 1] and

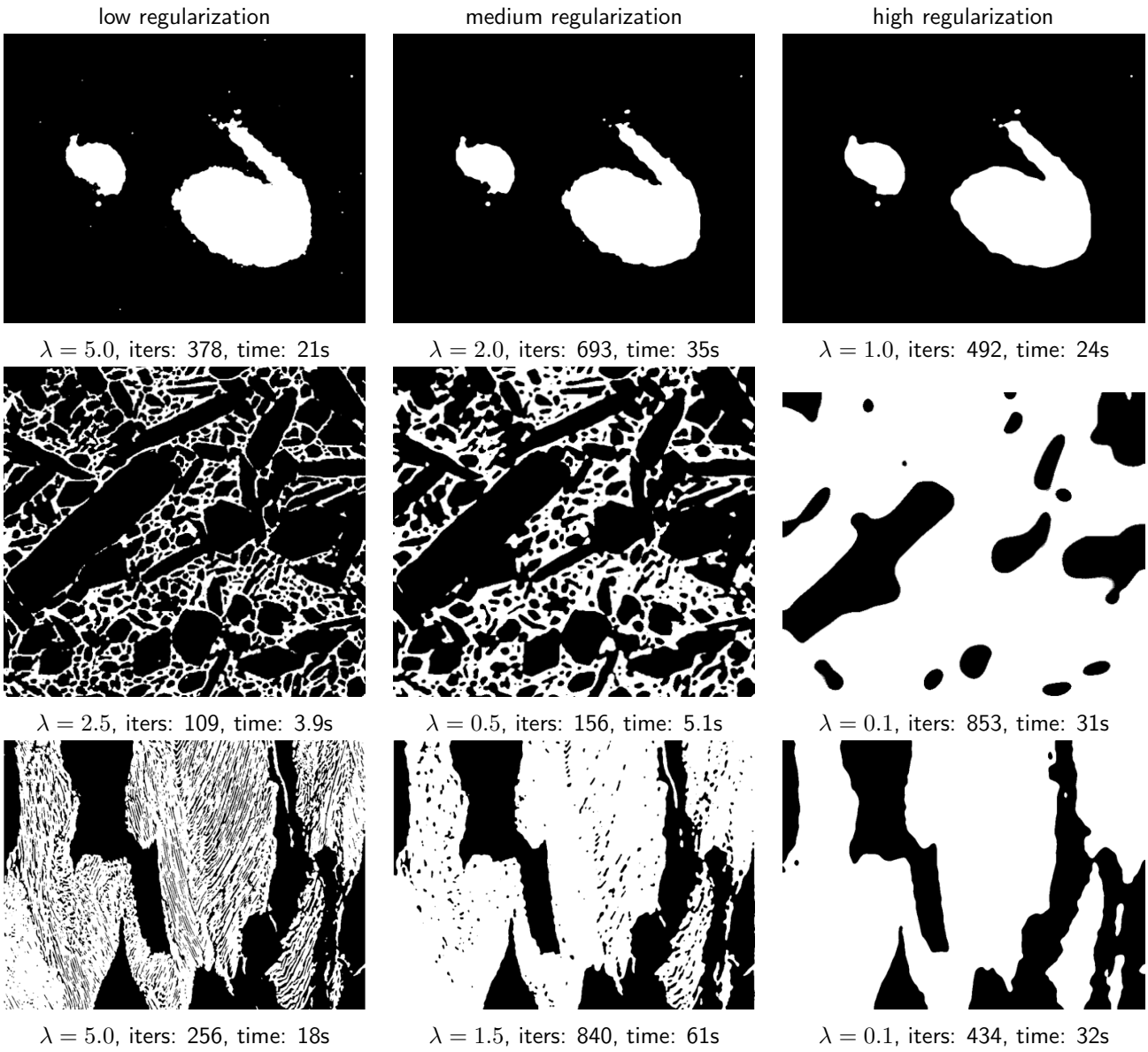


Figure 4: Higher values of λ result in reduced regularization of region boundaries (left), whereas lower values lead to increased regularization, therefore, smoother region boundaries (middle, right).

Chan-Vese segmentation [4, 5]. Otsu thresholding groups the image pixels into two regions, based on the pixel values and whether the values are higher or lower than a threshold value. The threshold value can be selected automatically by the Otsu algorithm, which is the option we used in our experiments with the implementation of [1]. The Chan-Vese model [4] is essentially the same energy minimized in this paper (which follows [7]). In [4], the segmentation energy is minimized by the gradient descent evolution of a level set function representing the domains. For this computation, we used the C++ implementation of [5]. This implementation had multiple parameters, such as the length penalty weight μ, which we set to 0.2 for galaxy and 0.1 for microstructure and pearlite images, other model parameters (for which we used the defaults) of $\nu = 0$ (area weight), $\lambda_1 = 1$ (fit weight of the foreground region), $\lambda_2 = 1$ (fit weight of the background region), and the iteration parameters were set to $dt = 1.0$, $maxiter = 1000$, $tol = 0.001$ for the galaxy and $tol = 0.002$ for the microstructure and pearlite images.

The segmentations obtained using these three methods are shown in Figure 5. As expected, the segmentation results from Otsu thresholding are noisy, because it does not have boundary or region

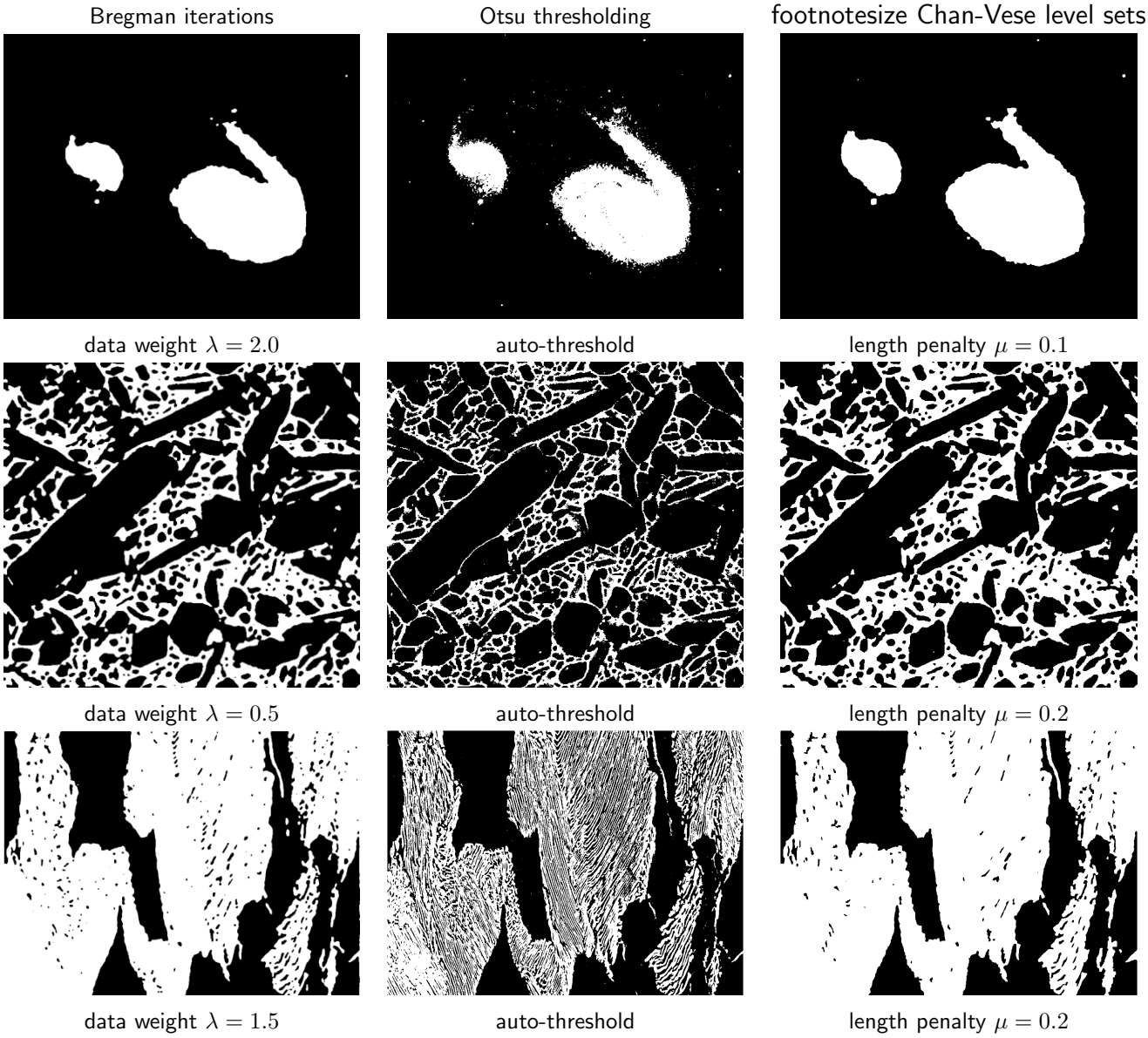


Figure 5: Segmentation results obtained using Bregman iterations (left), Otsu thresholding with auto-threshold (middle), Chan-Vese level set method (right). Bregman iterations and Chan-Vese algorithms produce comparable smooth segmentations, whereas Otsu thresholding segmentation is noisy.

regularization. The final results of the Chan-Vese segmentation are comparable to the results from our implementation of Bregman iterations. We include additional comparisons using images from [5, 1] in Figures 6 and 7. Note that the μ weight of the boundary length penalty in the Chan-Vese model, as well as our λ data weight, can be adjusted to obtain smoother (or less smooth) segmentations. The level set minimization of the Chan-Vese segmentation using [5] took many more iterations than our Bregman algorithm solutions (which had motivated this later approach). We should still note that the computational cost and the number of iterations of the Chan-Vese algorithm depend very much on the values of *tol*, *maxiter*, and *dt*, so it can be considerably higher or lower. In any case, the C++ implementation was very efficient and fast.

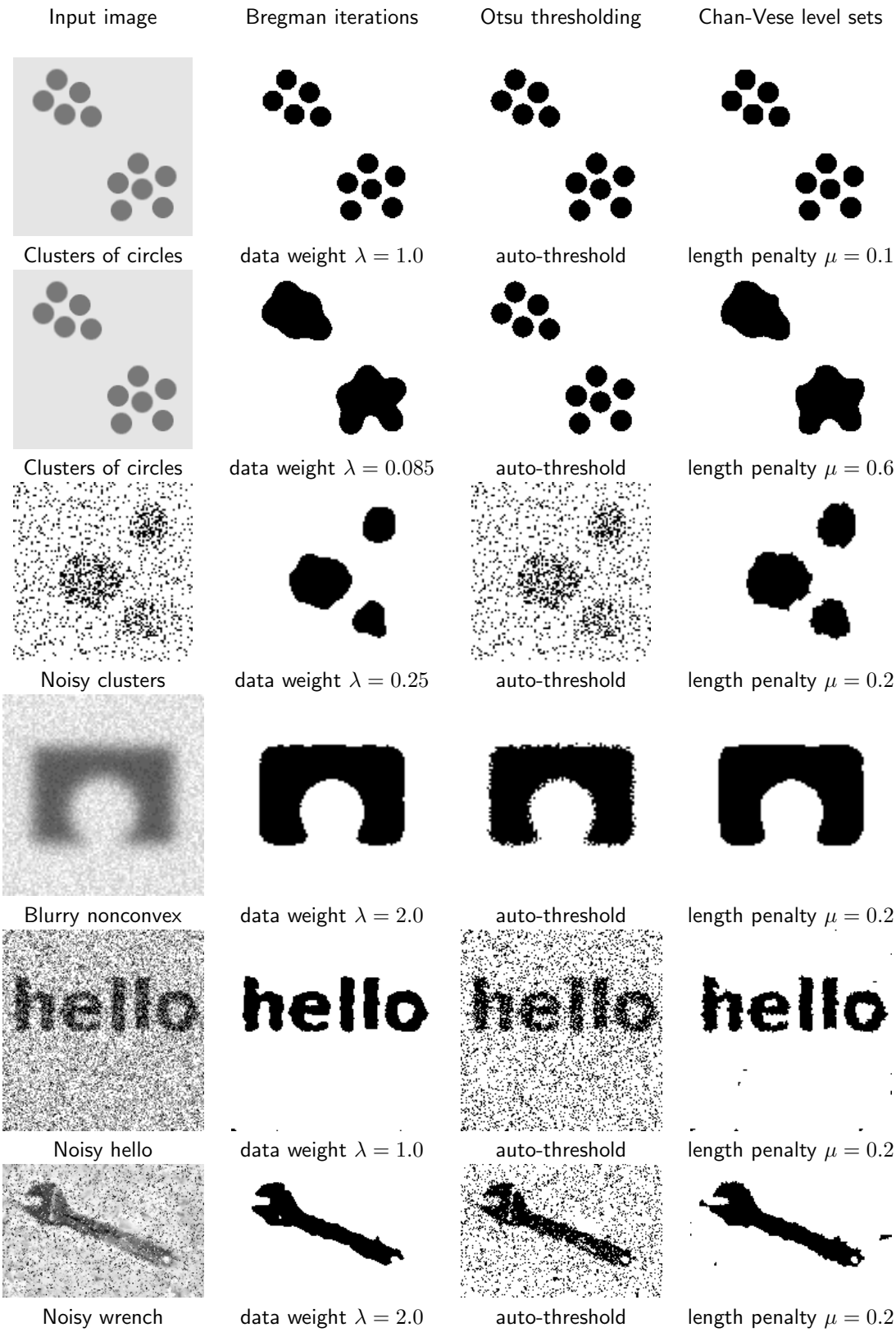


Figure 6: Segmentation results on synthetic images from [5, 1] (1st column) obtained using Bregman iterations (2nd column), Otsu thresholding with auto-threshold (3rd column), Chan-Vese level set method (4th column). Bregman iterations and Chan-Vese algorithms produce comparable smooth segmentations, whereas Otsu thresholding segmentation is noisy.

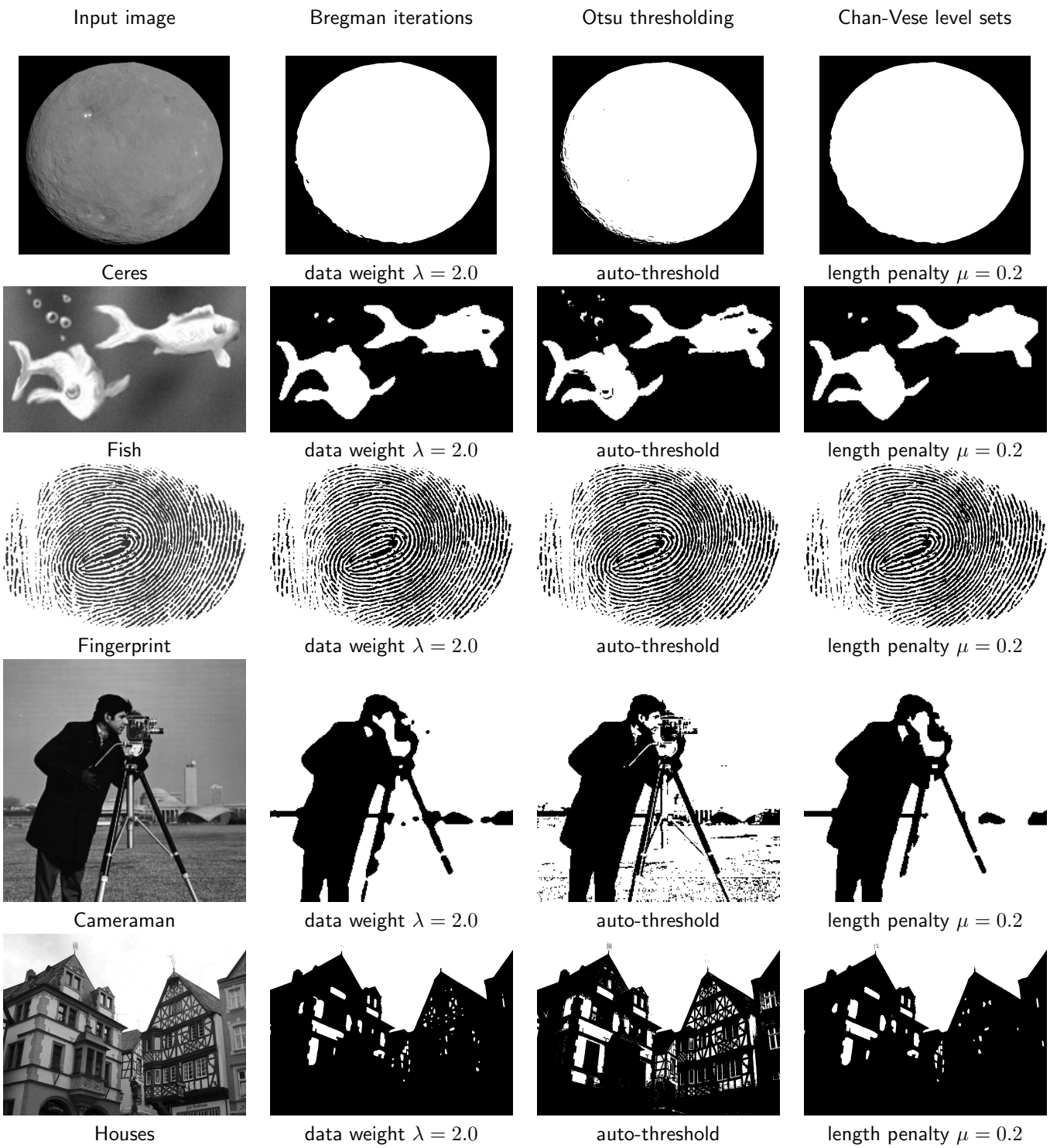


Figure 7: Segmentation results on real images from [5, 1] (1st column) obtained using Bregman iterations (2nd column), Otsu thresholding with auto-threshold (3rd column), Chan-Vese level set method (4th column).

Disclaimer

Certain equipment, instruments, software, or materials, commercial or non-commercial, are identified in this paper in order to specify the experimental procedure adequately. Such identification is not intended to imply recommendation or endorsement of any product or service by NIST, nor is it intended to imply that the materials or equipment identified are necessarily the best available for the purpose.

Acknowledgments

The first author was partially supported by the Mathematical Sciences Graduate Internship (MSGI) Program of U.S. National Science Foundation (NSF) during his internship at the U.S. National Institute of Standards and Technology (NIST), also by the Alexander von Humboldt Foundation German Research Chair funding and the associated Der Deutsche Akademische Austauschdienst (DAAD) Projects No. 57610033 and 57761435.

Image Credits

 Wikipedia[2],

 Figure 6 in [17],

 Wikimedia Commons[3],

 Pascal Getreuer [5], CC-BY,

 Ceres Dwarf Planet (NASA/JPL-Caltech/UCLA/MPS/DLR/IDA), Public Domain[4],

 Houses (Wikipedia), CC-BY-SA[5],

 Fingerprint (Wikipedia), CC-BY-SA 3.0[6],

 Cameraman, standard test image.

References

[1] J. P. Balarini and S. Nesmachnow, *A C++ Implementation of Otsu's Image Segmentation Method*, Image Processing On Line, 6 (2016), pp. 155–164, https://doi.org/10.5201/ipol.2016.158.

[2] A. Chambolle, *An Algorithm for Total Variation Minimization and Applications*, Journal of Mathematical Imaging and Vision, 20 (2004), pp. 1–8, https://doi.org/10.1023/B:JMIV.0000011325.36760.1e.

[2] https://de.m.wikipedia.org/wiki/NGC_5278/79
[3] https://upload.wikimedia.org/wikipedia/commons/7/77/Pearlite.jpg
[4] http://photojournal.jpl.nasa.gov/jpeg/PIA19562.jpg
[5] https://commons.wikimedia.org/wiki/File:Image_processing_pre_otsus_algorithm.jpg
[6] https://commons.wikimedia.org/wiki/File:Fingerprintforcriminologystubs.jpg

[3] T. F. Chan, S. Esedoğlu, and M. Nikolova, *Algorithms for Finding Global Minimizers of Image Segmentation and Denoising Models*, SIAM Journal of Applied Mathematics, 6 (2006), pp. 1632–1648, https://doi.org/10.1137/040615286.

[4] T. F. Chan and L. A. Vese, *Active Contours Without Edges*, IEEE Transactions on Image Processing, 10 (2001), pp. 266–277, https://doi.org/10.1109/83.902291.

[5] P. Getreuer, *Chan-Vese Segmentation*, Image Processing On Line, 2 (2012), pp. 214–224, https://doi.org/10.5201/ipol.2012.g-cv.

[6] ——, *Rudin-Osher-Fatemi Total Variation Denoising Using Split Bregman*, Image Processing On Line, 2 (2012), pp. 74–95. https://doi.org/10.5201/ipol.2012.g-tvd.

[7] T. Goldstein, X. Bresson, and S. Osher, *Geometric Applications of the Split Bregman Method: Segmentation and Surface Reconstruction*, Journal of Scientific Computing, 45 (2010), pp. 272–293, https://doi.org/10.1007/s10915-009-9331-z.

[8] T. Goldstein, M. Li, and X. Yuan, *Adaptive Primal-Dual Splitting Methods for Statistical Learning and Image Processing*, Advances in Neural Information Processing Systems, 2 (2015), pp. 2089–2097. https://dl.acm.org/doi/10.5555/2969442.2969473.

[9] T. Goldstein, B. O'Donoghue, S. Setzer, and R. Baraniuk, *Fast Alternating Direction Optimization Methods*, SIAM Journal of Imaging Sciences, 7 (2014), pp. 1588–1623, https://doi.org/10.1137/120896219.

[10] T. Goldstein and S. Osher, *The Split Bregman Method for L1 Regularized Problems*, SIAM Journal of Imaging Sciences, 2 (2009), pp. 323–343, https://doi.org/10.1137/080725891.

[11] L. Kiefer, M. Storath, and A. Weinmann, *PALMS Image Partitioning - A New Parallel Algorithm for the Piecewise Affine-Linear Mumford-Shah Model*, Image Processing On Line, 10 (2020), pp. 124–149, https://doi.org/10.5201/ipol.2020.295.

[12] J. Moreno, V. Surya Prasath, H. Proenca, and K. Palaniappan, *Fast and Globally Convex Multiphase Active Contours for Brain MRI Segmentation*, Multiscale Modeling and Simulation, 4 (2006), pp. 2281–2289, https://doi.org/10.1016/j.cviu.2014.04.010.

[13] D. B. Mumford and J. Shah, *Optimal Approximations by Piecewise Smooth Functions and Associated Variational Problems*, Communications on Pure and Applied Mathematics, (1989), https://doi.org/10.1002/cpa.3160420503.

[14] S. Osher, M. Burger, D. Goldfarb, J. Xu, and W. Yin, *An Iterative Regularization Method for Total Variation-Based Image Restoration*, Multiscale Modeling and Simulation, 4 (2006), pp. 2281–2289, https://doi.org/10.1137/040605412.

[15] N. Otsu, *A Threshold Selection Method from Gray-Level Histograms*, IEEE Transactions on Systems, Man, and Cybernetics, 9 (1979), pp. 62–66, https://doi.org/10.1109/TSMC.1979.4310076.

[16] L. I. Rudin, S. Osher, and E. Fatemi, *Nonlinear Total Variation Based Noise Removal Algorithms*, Physica D: Nonlinear Phenomena, 60 (1992), pp. 259–268, https://doi.org/10.1016/0167-2789(92)90242-F.

[17] J. J. Swab, A. A. Wereszczak, J. Tice, R. Caspe, R. H. Kraft, and J. W. Adams, *Mechanical and Thermal Properties of Advanced Ceramics for Gun Barrel Applications*, Army Research Laboratory, ARL-TR-3417, (2005).