University of California

Los Angeles

Theshold Dynamics for Statistical Density Estimation and Graph Clustering

A dissertation submitted in partial satisfaction

of the requirements for the degree

Doctor of Philosophy in Mathematics

by

Tijana Kostic

2013

Abstract of the Dissertation

Theshold Dynamics for Statistical Density Estimation and Graph Clustering

by

Tijana Kostic

Doctor of Philosophy in Mathematics

University of California, Los Angeles, 2013

Professor Andrea L. Bertozzi, Chair

In 1992 Merriman, Bence and Osher [7] proposed a computationally inexpensive threshold dynamics algorithm for the approximation of the motion by mean curvature. Since its introduction, numerous generalizations of the algorithm have been made, and the algorithm has been successfully used in a wide variety of computer vision applications, such as image segmentation, image inpainting, surface reconstruction etc.. The main focus of this work are the extensions of the original algorithm as well as multiple new applications such as probability density estimation and graph segmentation.

Part I discusses a threshold dynamics segmentation algorithm for estimating a probability density based on discrete point data. Since point data may represent certain activities, such as crime, this method can be successfully used for detecting regions of high activity, as well as locating the region where activities generally occur. To achieve the goal of accurately locating such regions, a binary segmentation version of the well-known Maximum Penalized Likelihood Estimation (MPLE) model is designed. The method is applied to different computational examples, including one with actual residential burglary data from the San Fernando Valley.

In Part II we present an adaptation of the classic Merriman-Bence-Osher (MBO) scheme utilizing a fully or semi nonlocal graph Laplacian for solving a wide range of learning problems in data clustering and image processing. Combining ideas from L^1 compressive sensing, image processing and graph methods, the diffuse interface model based on the Ginzburg-Landau functional was recently introduced to the graph community for solving problems in data classification. Here, we propose an algorithm for graph-based methods and also make use of fast numerical solvers for finding eigenvalues and eigenvectors of the graph Laplacian. To demonstrate the performance of our model, various computational examples are presented, which proves that the method is successful on images with texture and repetitive structure due to its nonlocal nature. A wide range of applications is discussed, including data labeling, image segmentation and image inpainting, which demonstrates the versatility of the proposed algorithm. The success of this algorithm also raises an important theoretical question: is it possible to define an analog of the motion by mean curvature of surfaces on graphs, and what properties would such notion possess.

The dissertation of Tijana Kostic is approved.

Joseph Teran

P. Jeffrey Brantingham

Chris Anderson

Andrea L. Bertozzi, Committee Chair

University of California, Los Angeles

2013

Table of Contents

List of Figures

List of Tables

Acknowledgments

I would like to express my deepest gratitude to my academic advisor, Professor Andrea Bertozzi. She helped my identify and pursue my research interests. This work would have not been possible without her guidance and mathematical expertise. I would also like to thank the members of my doctoral commitee, Professors Chris Anderson, Jeffrey Brantingham and Joseph Teran for their helpful and insightful comments during my thesis defense.

The research found in Chapter 2 was done in collaboration with Professor Andrea Bertozzi, and is from the published paper "Statistical Density Estimation using Threshold Dynamics for Geometric Motion", Journal of Scientific Computing 54.2-3 (2013):513-530. This work was dedicated to Professor Stanley Osher on the occasion of his 70th Birthday. I would like to thank Laura Smith and George Mohler for their helpful comments. This work was supported by ONR grant N000141210040, ONR grant N000141010221, AFOSR MURI grant FA9550-10-1-0569, NSF grant DMS-0968309 and ARO grant W911NF1010472, reporting number 58344-MA.

The work in Chapter 3 is from the article "An MBO scheme on graphs for segmentation and image processing". This research project was done in collaboration with my gifted colleague Ekaterina Merkurjev and Andrea Bertozzi. I would like to thank Ekaterina for her hard work and bright ideas. I would also like to thank Chris Anderson for providing a code for the Raleigh-Chebyshev procedure. I would like to acknowledge Arjuna Flenner, Yves van Gennip and Hayden Schaeffer for useful discussions regarding this work. This work was supported by ONR grants N000141210040, N000141010472, N00014120838 and AFOSR MURI grant FA9550-10-1-0569.

I would especially like to thank my family, my parents and my sister, for their endless love and support. Without their encouragement to always do my best I would have never made it this far. I want to thank my professors from University of Belgrade for nurturing my interest in mathematics. I feel grateful for their support to pursue a doctorate degree abroad. I also want to acknowledge Maggie Albert, Martha Contreras and Babette Dalton for helping me with numerous administrative matters. Finally, I would like to thank Hayden

Schaeffer, who has been my friend from the very first day of grad school. His academic support and sense of humor have been two very important parts of this wonderful journey.

Vita

2006	B.S. (Mathematics), University of Belgrade.
2007	M.A. (Mathematics), University of Belgrade.
2008–2009	Chancellor's Prize, UCLA.
2010	M.A. (Mathematics), UCLA.

Publications

T. Kostic, A. Bertozzi. "Statistical Density Estimation Using Threshold Dynamics for Geometric Motion." Journal of Scientific Computing 54.2-3 (2013): 513-530.

E. Merkurjev, T. Kostic, and A. Bertozzi. "An MBO scheme on graphs for segmentation and image processing." accepted in SIAM Journal on Imaging Sciences.

CHAPTER 1

Introduction to the MBO method

1.1 Variational Methods in Image Processing

Over the past few decades variational methods became state-of-the-art techniques in image processing. Each method of the class has an energy functional associated with it, a one that is constructed in such way that it models a certain property of the image. Due to the optimizational nature of these methods the goal is to, using variational techniques, minimize the given functional. Variational methods have been successfully used to solve many different restoration problems in image processing. In their groundbreaking work on image segmentation [47], Mumford and Shah proposed the following functional that translates the common image properties into a mathematical framework:

$$\min_{\substack{K \subseteq D \\ u: D \to \mathbf{R}}} MS(u, K) = \int_{D \setminus K} |\nabla u|^2 dx + \mu Length(K) + \lambda \int_D (u - f)^2 dx \tag{1.1}$$

where f is an image that should be segmented, u is a segmentation function, and K is a set of curves on which function u is discontinuous. The segmentation function u is supposed to be a piecewise constant approximation of the image f thus capturing the regions in the given image. The properties modeled by the constraints given in the functional (1.1) are the following: the first term in (1.1) controls the smoothness of the segmentation function u outside of the set of curves K, the second term controls the length of K and the last one encourages u to be close to the original image f. From the need to bound the length of the edge set K arises an assumption that u is a function of bounded variation. The idea to use total variation norm as a regularization term was introduced by Rudin, Osher and Fatemi in 1992 in their pioneering work on image denoising [40], where they proposed the following functional:

$$\min_{u\in BV(D)} \|u\|_{TV(D)} + \frac{\lambda}{2}\int_D (u-f)^2 dx \tag{1.2}$$

where u is a denoised function. Gradient descent minimization of the functional (1.2) yields the following equation

$$\begin{cases} \frac{\partial u}{\partial t} = \nabla \cdot \frac{\nabla u}{|\nabla u|} + \lambda(f-u) \\ \nu \cdot \nabla u = 0 & \text{on } \partial D. \end{cases} \tag{1.3}$$

To find a numerical solution of (1.3) the authors proposed a numerical scheme where they used an explicit discretization of the curvature term. Their approach leaves an open question: Is it possible to use the alternative ways to approximate the curvature term? In this work, we will use an alternative method, called the MBO scheme to approximate different curvature-like motions. In 2 we will describe the application of the MBO scheme for density estimation and in 3 we propose the MBO scheme for the graph based non-local methods.

1.2 The MBO Scheme

In case of motion by mean curvature a planar curve moves with a normal velocity that is proportional to the curvature at any given point. In their work [7], Merriman, Bence and Osher (MBO) introduced an intuitive and efficient level set method to approximate the motion of an interface by its mean curvature flow. The methods developed to approximate the motion by mean curvature prior to the MBO scheme were complicated and computationally inefficient, while the MBO scheme can successfully handle even the cases when the curve has self intersections and multiple junctions.

1.2.1 Intuition of the MBO scheme

The authors of the MBO analyzed the motion by mean curvature of the curve C by studying the diffusion equation $\chi_t = \Delta\chi$, where χ is the characteristic function of the set Σ and $\partial\Sigma = C$, i.e. C represents a sharp front between two phases of the characteristic function. Considering a point P on the boundary of the set, and switching to local polar coordinate system (r, θ) with origin at the center of the curvature of P, as it is shown in 1.1, the heat equation we are discussing becomes:

$$\chi_t = \frac{D}{r}\chi_r + D\chi_{rr} + \frac{D}{r^2}\chi_{\theta\theta}$$

Since $\chi_\theta = 0$ because of the local circular symmetry the equation can be written as:

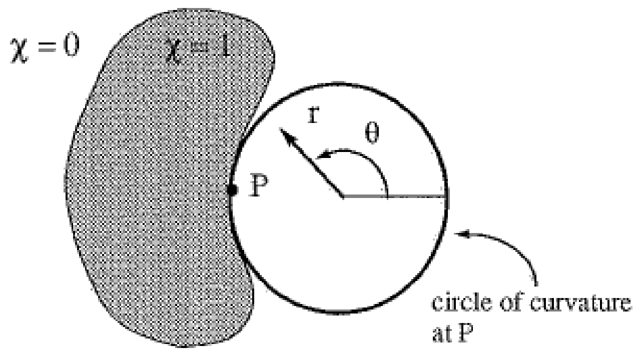


Figure 1.1: The parameters in the suggested change of coordinates. This picture originally appeared in [7].

$$\chi_t = \frac{D}{r}\chi_r + D\chi_{rr}$$

which reveals that advective velocity is proportional to mean curvature. While the boundary moves at the advective speed, diffusion simultaneously blurs the front. However, the $\chi = \frac{1}{2}$ level set is invariant to blurring. From there it follows that, the $\chi = \frac{1}{2}$ level set yields motion by mean curvature. Image 1.2 shows us the geometric prospective of the motion. Based on the previous observation the following numerical scheme is proposed:

- **Step 1** Let $v(x) = S(\delta t)u_n(x)$ where $S(\delta t)$ is a propagator by time δt of the equation:

  $$v_t = \Delta v$$

  with appropriate boundary conditions.

- **Step 2** Threshold

$$u_{n+1}(x) = \begin{cases} 0 & \text{if } v(x) \in (-\infty, \frac{1}{2}] \\ 1 & \text{if } v(x) \in (\frac{1}{2}, \infty). \end{cases}$$

The only parameter in the proposed numerical scheme is time step δt. We choose δt such that is small enough so the local analysis derived in 1.2.1 is still valid, but on the other hand, big enough so the curve does not get stuck.

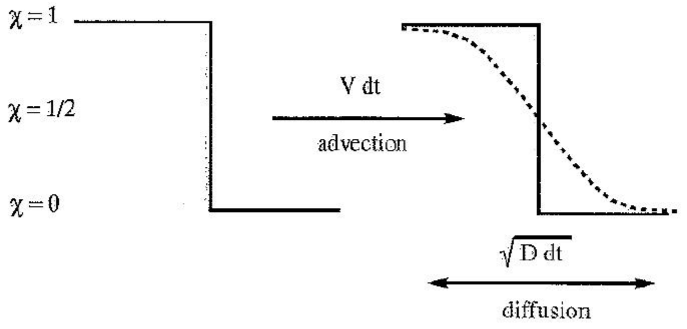


Figure 1.2: $\frac{1}{2}$-level set is invariant to diffusion. This image originally apeared in [7].

1.2.2 Applications of the MBO scheme

The simplicity of the MBO scheme inspires researchers to devise similar thresholding schemes to approximate different curvature-like motions. In 2.2 we describe an application the MBO scheme on image segmentation. In 2.5 we propose a threshold dynamics algorithm for locating the regions of high density in geographic point data. In 2.6 we explain the implementation details and the concept of adaptive timestepping and adaptive time resolution, the numerical procedures we used to speed up our threshold dynamics scheme. The graph based methods are becoming increasingly popular in many spheres of image and data processing. However, the graph representation usually poses numerous computational challenges and the iterations of the algorithms featuring non-local data representation tend to be very expensive. In 3.4 and 3.5 we present the threshold dynamics graph-based methods for data labeling and inpainting. In 3.4.2, 3.5.1 and 3.5.2 we demonstrate that out threshold dynamics algorithm on average takes fewer iterations to converge, and thus is more efficient than some other famous graph-based algorithms.

1.3 Image Segmentation

We approach the density estimation problem that we solve in 2 as a segmentation problem. To construct the model we propose in 2.3 we use the Ginzburg-Landau functional as a regularizer, the concept previously introduced in image segmentation. These models are commonly called diffuse interface models.

1.3.1 Ginzburg-Landau functional in Image Processing

Rudin, Osher and Fatemi in their work [40] developed a denoising model that uses the total variation semi-norm as a regularizer. The idea to minimize the total variation of a noisy image seems to be a natural one. The TV regularization technique has since been popular in many other image processing applications such as segmentation, inpainting, cartoon texture decomposition etc.. In the case of image segmentation the Ginzburg-Landau functional appears in the literature as an alternative to the TV semi-norm.
Numerous image segmentation energy functionals use a binary segmentation function that takes a certain value inside the segmented region and a different one outside of the segmented region. In their pioneering work [47], Mumford and Shah propose the energy functional that uses the perimeter of the segmentation function as a regularizer. Many papers, such as [13], successfully use the total variation (TV) semi-norm

$$||u||_{TV} = \int_{\Omega} |\nabla u| dx \tag{1.4}$$

to approximate the perimeter of the front between the two values of the segmentation function. As an alternative to this approach, many researchers, such as Esedoglu and Tsai in their work [23], use the Ginzburg-Landau functional

$$GL(u) = \frac{\epsilon}{2}\int |\nabla u|^2 dx + \frac{1}{\epsilon}\int W(u)dx \tag{1.5}$$

to approximate the perimeter of the front. $W(u)$ is a double well potential. In this work, $W(u) = (u^2 - 1)^2$ is used. Note that due to the nature of the potential, the functional is used for binary data.
A proof in [38] shows that the perimeter is the limit in the sense of Γ- convergence of the Ginzburg-Landau functional. Therefore, one can write

$$GL(u) \to_\Gamma C|u|_{TV}. \tag{1.6}$$

This convergence allows the two functionals to be interchanged in some cases. One might prefer to use the GL functional instead of the TV semi-norm since its highest order term is purely quadratic which allows for efficient minimization procedures. In contrast, minimization of the TV semi-norm leads to a nonlinear curvature term, making it less trivial to solve numerically. However, recent advances, such as the split Bregman method described in [32], have made progress in such problems.

Due to its connection to the TV semi-norm, the Ginzburg-Landau functional has also often been used in image processing and in various image processing applications, such as inpainting [18, 3] and segmentation [22, 23]. In practice, one would minimize

$$E(u) = GL(u) + F(u, u_0) \tag{1.7}$$

where F is the fidelity term and u_0 is the initial state of the system. In the case of inpainting, the fidelity term is $C\int(u - u_0)^2$, where one integrates over the known region only. For denoising, the term is an L^2 fit, $C\int(u-u_0)^2$. In the case of deblurring, it is $C\int(K*u-u_0)^2$, where K is some kernel. Of course, a different norm, such as the L^1 norm, can be used.

When one minimizes the Ginzburg-Landau functional, the function u approaches either one of the two minimizers, 1 and -1, of the double well potential. However, the presence of the gradient term will force u to be somewhat smooth, i.e. without any sharp transitions between 1 and -1. Therefore, the function that minimizes the functional will have regions where it is close to -1, close to 1 and a thin region of scale $O(\epsilon)$ where it is somewhere in between. Since the minimizer appears to have two phases with an interface between them, models involving the Ginzburg-Landau functional are typically referred to as "diffuse interface models".

1.3.2 Chan-Vese method

The famous Mumford-Shah segmentation functional proposed in [47] is:

$$\min_{\substack{\Sigma \subseteq \Omega \\ u:\Omega\to\mathbf{R}}} MS(u,\Sigma) = \int_{\Omega\setminus\Sigma} |\nabla u|^2 dx + \mu Per(\Sigma;\Omega) + \lambda \int_{\Sigma} (u-f)^2 dx \tag{1.8}$$

where f is an image that should be segmented, and u is a segmentation function. Another important model built around the Mumford-Shah functional is the Chan-Vese model. In their work [13], Chan and Vese give a level set formulation of the piecewise constant Mumford-Shah model. This is an active contour model that uses an evolving curve to detect an object in the image. For instance, a curve, that was initially given as a circle around the object to be detected will move inward until it is aligned with the boundary of the object. Unlike classic active contour models that can only detect objects with sharp gradient edges, the Chan-Vese model is more robust and can segment objects with no visible boundaries. To describe their model, the authors define the evolving curve C, and an open set ω where $C = \partial\omega$. The level set function $\phi : \Omega \to \mathbb{R}$ is defined such that:

$$\begin{cases} \partial\omega = \{(x,y) \in \Omega : \phi(x,y) = 0\} \\ \omega = \{(x,y) \in \Omega : \phi(x,y) > 0\} \\ \bar{\omega}^C = \{(x,y) \in \Omega : \phi(x,y) < 0.\} \end{cases} \tag{1.9}$$

The level set form of the length term in the Mumford-Shah function is:

$$\begin{aligned} Length(C) &= \int_\Omega \mid \nabla H(\phi(x,y)) \mid dxdy \\ &= \int_\Omega \delta_0(\phi(x,y)) \mid \nabla\phi(x,y) \mid dxdy. \end{aligned} \tag{1.10}$$

H is the Heaviside function, and δ_0 is the Dirac measure. The piecewise constant Mumford-Shah energy functional is:

$$\min_{\substack{\Sigma\subseteq\Omega \\ c_1,c_2\in\mathbf{R}}} E(\Sigma, c_1, c_2) = Per(\Sigma;\Omega) + \lambda\int_\Sigma (c_1 - f)^2 dx + \lambda\int_{D\setminus\Sigma}(c_2 - f)^2 dx. \tag{1.11}$$

The level set formulation of 2.1 is:

$$\min_{\substack{\phi:\Sigma\to\mathbf{R} \\ c_1,c_2\in\mathbf{R}}} F(\phi, c_1, c_2) = \int_D |\nabla H(\phi)| + \lambda\left\{H(\phi)(c_1 - f)^2 + (1 - H(\phi))(c_2 - f)^2\right\}dx. \tag{1.12}$$

The minimization of the energy $F(\phi, c_1, c_2)$ from 1.12 for the fixed function ϕ gives us the following values of c_1 and c_2:

$$c_1(\phi) = \frac{\int_\Omega u_0(x,y)H(\phi(x,y))dxdy}{\int_\Omega H(\phi(x,y))dxdx} \tag{1.13}$$

if $\int_\Omega H(\phi(x,y))dxdy > 0$ and

$$c_2(\phi) = \frac{\int_\Omega u_0(x,y)(1 - H(\phi(x,y)))dxdy}{\int_\Omega (1 - H(\phi(x,y)))dxdx} \tag{1.14}$$

if $\int_\Omega (1 - H(x,y))dxdy > 0$. After the optimal values for constants c_1 and c_2 are determined, the time-dependent Euler-Lagrange equation for ϕ is found:

$$\phi_t = H'_\epsilon(\phi)\left\{\nabla\cdot\left(\frac{\nabla\phi}{|\nabla\phi|}\right) - \lambda\left\{(c_1 - f)^2 - (c_2 - f)^2\right\}\right\} \tag{1.15}$$

where H_ϵ is an approximation of the Heaviside function, and a semi-implicit numerical scheme is created to solve (1.15). Computational examples of Chan-Vese algorithm are shown in

Figure 1.3 and Figure 1.4. Due to the computational complexity of the numerical algorithm that solves equation (1.15), Gibou and Fedkiw created algorithms that do not explicitly solve (1.15). In their work [29], they designed the hybrid k-means level set method and applied it to images that were previously processed using the Perona-Malik diffusion. Another algorithm that successfully minimizes the piecewise constant Mumford-Shah functional without solving the gradient descent equation (1.15) is proposed in [60]. Numerous modifications of the piecewise constant Mumford-Shah model, and different ways to minimize it appeared in the literature (see [21], [15],[23], [28], [63] and [64]).
Although Chan-Vese model is not a diffuse interface model, Esedoglu and Tsai in their work [23] propose a piecewise constant Mumford-Shah functional with the Gizburg-Landau functional. The idea to use this type of regularization instead of the TV semi-norm largely influences the direction of our research. More detailed description of work by Esedoglu and Tsai is given in 2.2

1.4 Maximum Penalized Likelihood Estimation

Geographic point data can be used to represent any kind of events and activity, such as crime.

Let us assume our data consists of n points $\mathbf{x}_1$,$\mathbf{x}_2$,...,$\mathbf{x}_n$ and is a sample of n independent random variables with common density d_0. Maximum likelihood estimation is a standard method for estimating the density based on given data. In the case of a parametric model, we know that d_0 belongs to the family of density functions $D = \{d(\cdot, \theta) : \theta \in \Theta\}$, and our goal is to find a parameter θ_0 such that $d(\cdot, \theta_0) = d_0(\cdot)$. This class of problems are known as parametric density estimates. In 1922, According to Fisher's model [24], an optimal parameter $\theta_0 \in \Theta$ (where Θ is a set of all parameters) satisfies the following:

$$\theta_0 = -\min_{\theta \in \Theta} \frac{1}{n} \sum_{i=1}^{n} \log d(\mathbf{x}_i, \theta). \tag{1.16}$$

However, in many applications, a parametric model may not be available, or information

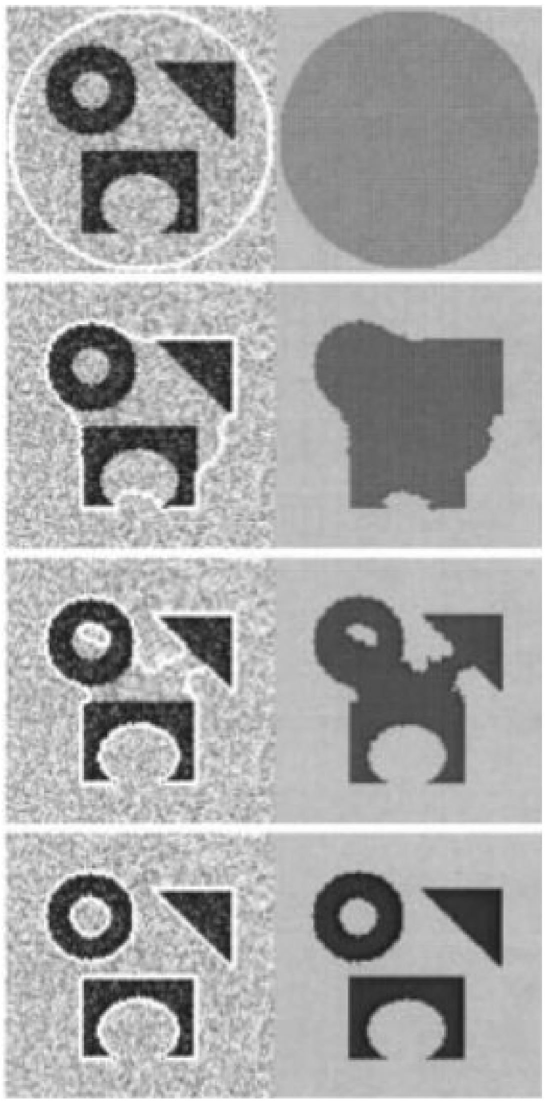


Figure 1.3: Detection of objects in a noisy image. The curve evolution is shown. The intensity of the pixels inside the curve is proportional to the value of c_1 from 1.13. The image originally appeared [13].

about the family of density functions may be unknown, in which case we are dealing with a nonparametric density estimate. The analog of the model (1.16) is an ill-posed problem, i.e. finding a probability density function d_0 such that

$$d_0 = - \min_{\substack{d:\Omega\to\mathbb{R} \\ \int d=1}} \frac{1}{n} \sum_{i=1}^{n} \log d(\mathbf{x}_i) \tag{1.17}$$

has no solution, and thus can not be directly applied. A class of standard methods to

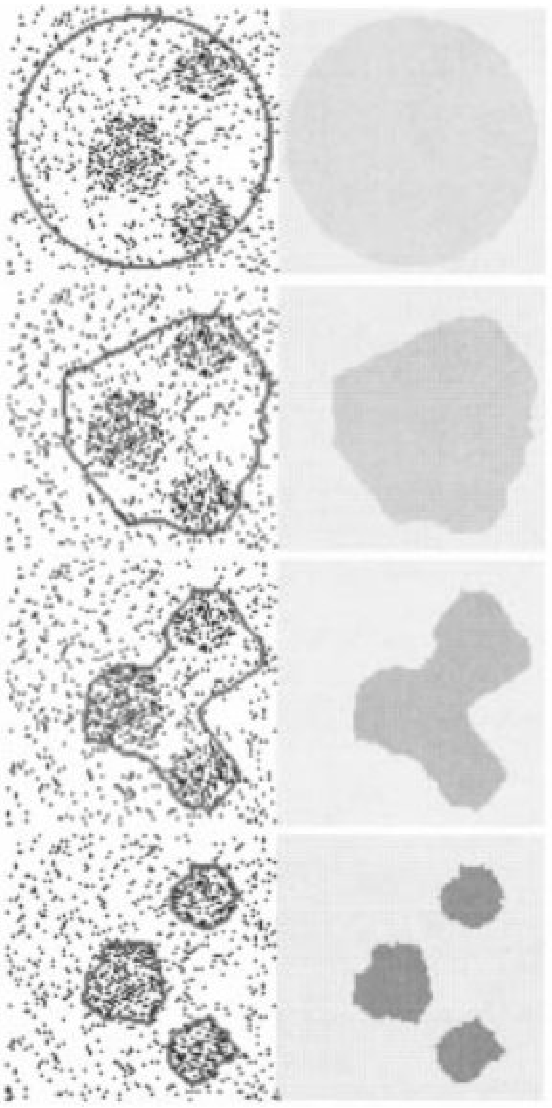


Figure 1.4: Detection of the clusters in the point data. The curve evolution is shown. The intensity of the pixels inside the curve is proportional to the value of c_1 from 1.13. This image originally appeared in [13].

solve this problem are kernel density estimation methods [58, 59]. These methods impose smoothness by approximating the density function by a sum of kernel functions.

$$d(\mathbf{x}) = \frac{1}{nh} \sum_{i=1}^{n} k\left(\frac{\mathbf{x} - \mathbf{x}_i}{h}\right),$$

where h is a bandwidth. The kernel density functions are chosen to be smooth, radially symmetric and with non-compact support. The Gaussian distribution appears in the literature as a common choice for the kernel.

To avoid "rough" density functions, with large values concentrated around data points and small values elsewhere, the roughness penalization function $R(d)$ was introduced, and this method is known as maximum penalized likelihood estimation:

$$d(\mathbf{x}) = \max_{\substack{d:\Omega\to\mathbb{R} \\ \int d=1}} \left\{ \sum_{i=1}^{N} \log(d(\mathbf{x}_i)) - \alpha R(d) \right\}. \tag{1.18}$$

In 2 we propose an MPLE segmentation model with the goal to efficiently and precisely locate the dense regions in point data sets.

1.5 TV Regularization MPLE Methods

Different penalty functionals appear in literature, such as $R(d) = \int_\Omega |\nabla\sqrt{d}|^2$, or $R(d) = \int_\Omega |\nabla^3 \log d|^2$ from [20]. These and many other standard penalty functional enforce smoothness on density function, but do not perform well when the density function has sharp gradients, i.e. is piecewise constant. To resolve this issue, Koenker and Mizera in [37] as well as Sardy and Tseng in [55] propose the penalty functional to be the TV semi-norm. This approach was also successfully used in [39] and [26]. In our work, since we assume the density is a step function, choosing a penalty functional that can successfully handle sharp gradients is crucial. As previously mentioned , instead of the TV semi-norm, we chose the Ginzburg-Landau functional to be the penalty functional. In [26] the authors, inspired by the Split Bregman technique, present an efficient minimization algorithm for two dimensional TV-based MPLE. The model proposed in [26] is

$$\min_{\substack{d:\Omega\to\mathbb{R} \\ \int d=1}} \left\{ \sum_{i=1}^{N} -\mu \log(d(\mathbf{x}_i)) + \mid \nabla d \mid \right\}. \tag{1.19}$$

To eliminate the constraint $\int d = 1$, the authors transform (1.19) using an augmented Lagragian model approach. To apply the Split Bregman techique, the new variable $\vec{s} = \nabla d$

is introduced, and the model becomes:

$$\min_{d,\vec{s}} \left\{ \sum_{i=1}^{N} -\mu \log(d(\mathbf{x}_i)) + \mid \vec{s} \mid + \frac{\lambda}{2}\|\vec{s} - \nabla d\|^2 + \gamma(1 - \sum_{i,j} d)^2 \right\} \tag{1.20}$$

where γ and λ are positive constants. The additional Bregman vectors are added into the functional 1.20 and which gives us the final version of the problem:

$$(d^k, \vec{s}^k) = \min_{d,\vec{s}} \left\{ \mid \vec{s} \mid - \mu \log(d(\mathbf{x}_i)) + \frac{\lambda}{2}\|\vec{s} - \nabla d - \vec{b}^{k-1}\|^2 + \gamma(1 - \sum_{i,j} d_{i,j} - b_1^{k-1})^2 \right\}, \tag{1.21}$$

$$\vec{b}^k = \vec{b}^{k-1} + \nabla d^k - \vec{s}^k, \tag{1.22}$$

$$b_1^k = b_1^{k-1} + \sum_{i,j} u_{i,j}^k - 1. \tag{1.23}$$

To minimize with respect to $\vec{s}$ the authors use the shrinkage function and to minimize with respect to d they use the gradient descent of (1.21), which yield the following formula:

$$-\frac{\mu w_{i,j}}{d_{i,j}} - \lambda \Delta d_{i,j} + \lambda(\nabla^T \vec{b_{i,j}} - \nabla^T \vec{s_{i,j}}) + \gamma(\sum_{i,j} d_{i,j} + b_1 - 1) = 0, \tag{1.24}$$

where w has value 1 only in data points $\mathbf{x}_i$ and 0 elsewhere. Note that (1.24) can be easily transformed into a quadratic equation with respect to $d_{i,j}$. Since a density function can only take non-negative values, the positive root is a value of $d_{i,j}$. In the case of piecewise constant density functions this algorithm performs significantly better than the kernel-based algorithms. The kernel algorithms produce overly smooth solutions that do not approximate the density very well. The comparison is shown in Figure 1.5. Standard density estimation methods do not incorporate geographical information, which may lead to the density function taking positive values in unrealistic locations. To prevent this, Smith et al. in their work [39] propose a modified version of the TV MPLE algorithm that successfully integrates spacial information into the model. Spacial information is usually obtained from city and county boundary maps, census data hyperspectral images and many other sources. Using prior information to obtain a valid region D, the goal is to find a density function that only take

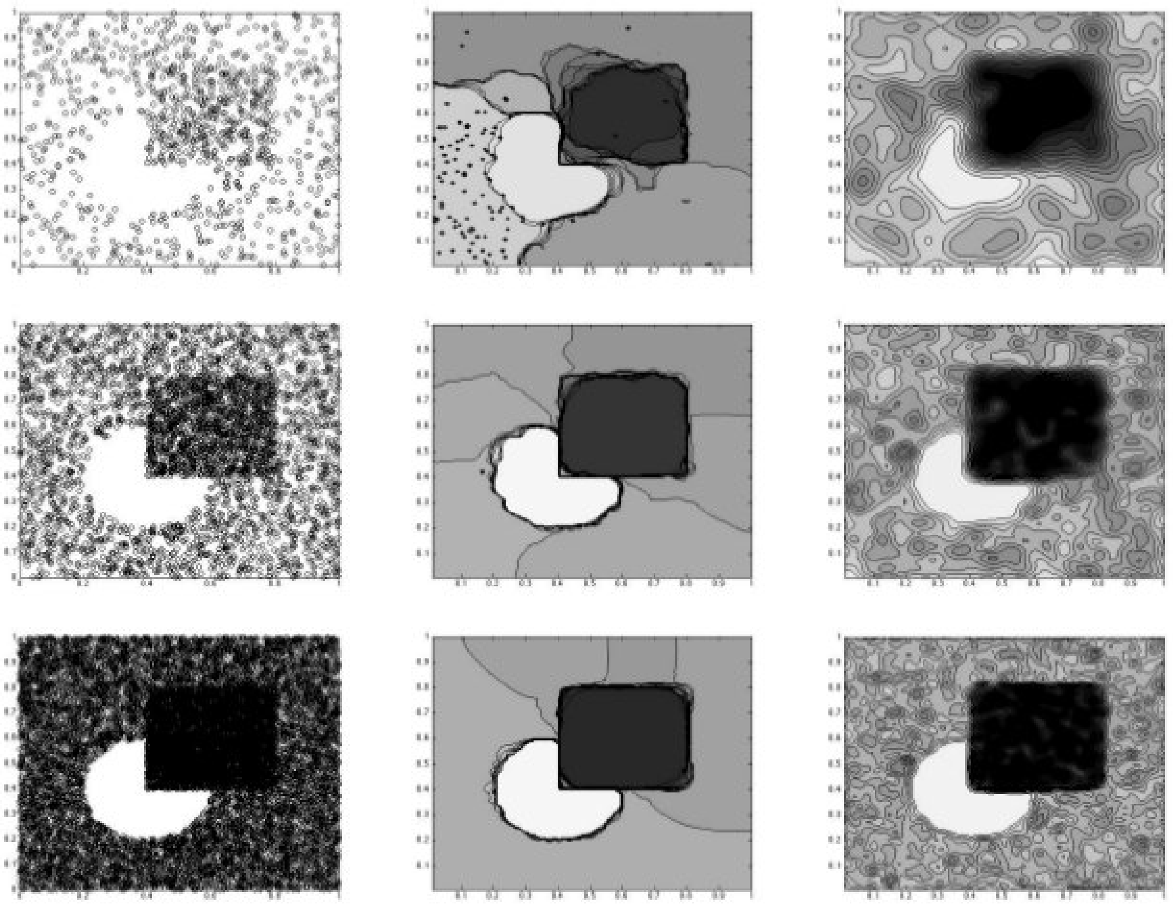


Figure 1.5: The images of the data set is given in the left row. The solutions obtained by the algorithm from [26] is in the middle and the solutions obtained using a kernel based method is on the right. This image originally appeared in [26].

non-zero values inside D. This can be achieved by aligning the level curves of the density function d with ∂D. Since $\frac{\nabla d}{|\nabla d|}$ gives the unit normal vectors to the level curves of d the condition is:

$$\frac{\nabla \chi_D}{\mid \nabla \chi_D \mid} = \frac{\nabla d}{\mid \nabla d \mid} \tag{1.25}$$

where χ_D is a characteristic function of D. One of the ways to formulate this constraint is by using an auxiliary variable. Let $\theta = \frac{\nabla \chi_D}{|\nabla \chi_D|_\epsilon}$ where $\mid v \mid_\epsilon = \sqrt{v_x^2 + v_y^2 + \epsilon^2}$. One of the ways to enforce a discontinuity on ∂D is to minimize $\mid d \mid - \theta \cdot \theta$. After integration by parts this term is $\int_\Omega \mid \nabla d \mid + d\nabla \cdot \theta dx$. The idea to control the boundary by aligning the normal vectors of the level sets is proposed by Buades et al. in [5]. The similar approach is also used for hyperspectral image fusion in [44]. The Modified total variation penalty function proposed in [39] is:

$$d_0((x)) = \min_{\substack{d:\Omega\to\mathbb{R} \\ \int d=1}} \left\{ \sum_{i=1}^{N} -\mu \log(d(\mathbf{x}_i)) + \int_\Omega \mid \nabla d \mid + \lambda d \nabla \cdot \theta d\mathbf{x} \right\}. \tag{1.26}$$

Another way to enforce the previously mentioned discontinuity constraint is by using the Ambrosio-Tortorelli approximating function $z_{epsilon}(\mathbf{x})$ where $z_\epsilon \to (1 - \delta(\partial D))$. Moreover,

$$z_\epsilon((x)) = \begin{cases} 1 & \text{if } distance(\mathbf{x}, \partial D), \\ 0 & \text{if } \mathbf{x} \in \partial D. \end{cases} \tag{1.27}$$

Thus the penalty functional is:

$$d_0((x)) = \min_{\substack{d:\Omega\to\mathbb{R} \\ \int d=1}} \left\{ \sum_{i=1}^{N} -\mu \log(d(\mathbf{x}_i)) + \frac{1}{2} \int_\Omega z_\epsilon^2 \mid \nabla d \mid^2 d\mathbf{x} \right\}. \tag{1.28}$$

In the case of both (1.26) and (1.28) the minimization is carried out by the adaptations of the Split Bregman algorithm. In Figure 1.6 we can see the comparison of the results from [26] and [39] as well as the performance of the kernel based algorithms.

1.6 Graph Methods in Image Segmentation

In chapter 3 we present a novel graph-based algorithm for data labeling and image inpainting. The algorithm utilizes the symmetric graph Laplacian, a concept that originates form graph theory and is recently introduced to the image processing community. In this section we show the connection between the normalized cut methods for image segmentation and the graph Laplacian. Graph-based methods have been present in the image segmentation literature for over thirty years and the formulations using graph cuts appeared more recently. In graph theory the cut is defined as a total weight of the edges between two graph partitions A and B, where $G = (V, E)$ and $A \cup B = V$:

$$cut(A, B) = \sum_{u \in A, v \in B} w(u, v). \tag{1.29}$$

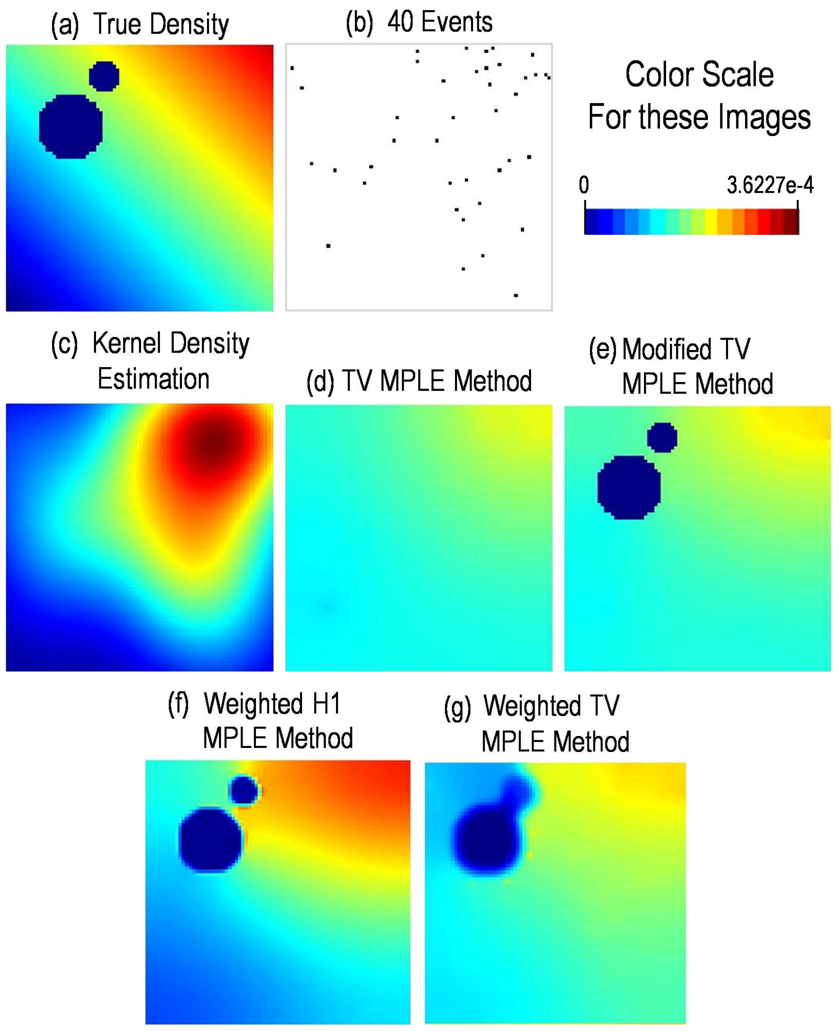


Figure 1.6: This is a sparse example with 40 events. The true density is given in (a), and it is the same density from the example in the introduction. The sampled events are shown in (b). Figures (c) and (d) show the two current density estimation methods, Kernel Density Estimation and TV MPLE. Figures (e), (f), and (g) show the density estimates from modified TV MPLE methods. The color scale represents the relative probability of an event occurring in a given pixel. The images are 80 pixels by 80 pixels. This is image originally appeared in [39].

Wu and Leahy [66] propose a clustering method where the optimal bipartitioning satisfies the min cut condition. However, the min cut condition will generally be biased toward very small sets. To avoid this Shi and Malik propose a different measure of fitness of a cut called the normalized cut:

$$Ncut(A,B) = \frac{cut(A,B)}{assoc(A,V)} + \frac{cut(A,B)}{assoc(B,V)} \tag{1.30}$$

where $assoc(A,V) = \sum_{u \in A, t \in V} w(u,t)$ is the total weight of all edges from nodes in A to all nodes in the graph and $assoc(B,V)$ is similarly defined. There is a way to mathematically formulate (1.30) using an $N = \mid V \mid$ dimensional indicator vector x where $x_i = 1$ if node i is in A and $x_i = -1$ otherwise. Let $d(i) = \sum_j w(i,j)$. With d and x (1.30) becomes:

$$Ncut(A,B) = \frac{\sum_{x_i>0, x_j<0} -w_{i,j} x_i x_j}{\sum_{x_i>0} d_i} + \frac{\sum_{x_i<0, x_j>0} -w_{i,j} x_i x_j}{\sum_{x_i<0} d_i}. \tag{1.31}$$

Let D be an $N \times N$ diagonal matrix with d on its diagonal, and W be an $N \times N$ symmetrical matrix with $W(i,j) = w_{i,j}$,

$$k = \frac{\sum_{x_i>0} d_i}{\sum_i d_i}, \tag{1.32}$$

and $\mathbf{1}$ be an $N \times 1$ vector of all ones. Further computations and a substitution $y = (\mathbf{1} + x) - b(\mathbf{1} - x)$ where $b = \frac{k}{1-k}$ transforms 1.31 into:

$$min_x Ncut(x) = min_y \frac{y^T (D-W) y}{y^T D y} \tag{1.33}$$

with the conditions $y(i) \in \{1, -b\}$ and $y^T D \mathbf{1}$. The authors propose the minimization of 1.33 by solving the generalized eigenvalue system:

$$(D - W)y = \lambda D y \tag{1.34}$$

with the constraint $y^T D \mathbf{1} = 0$. With the substitution $z = D^{\frac{1}{2}} y$ 1.34 becomes:

$$D^{-\frac{1}{2}} (D - W) D^{-\frac{1}{2}} z = \lambda z. \tag{1.35}$$

Since $z_0 = D^{\frac{1}{2}} \mathbf{1}$ is an eigenvector of 1.35 with eigenvalue 0, the condition $0 = yD\mathbf{1}$ is equivalent to $0 = z^T z_0$. Thus, z that minimizes λ in 1.35 is the second eigenvector of $D^{-\frac{1}{2}}(D-W)D^{-\frac{1}{2}}$ and λ is the second eigenvalue. Using z we can easily recover the partition. Since the scale of the eigenvector that yields the minimal normalized cut is unknown, to find the best partition the algorithm is usually applied for different values of the norm of the eigenvector. The algorithm can be applied recursively for multiway partitioning. The segmentation results obtain by this method are shown in Figure 1.7

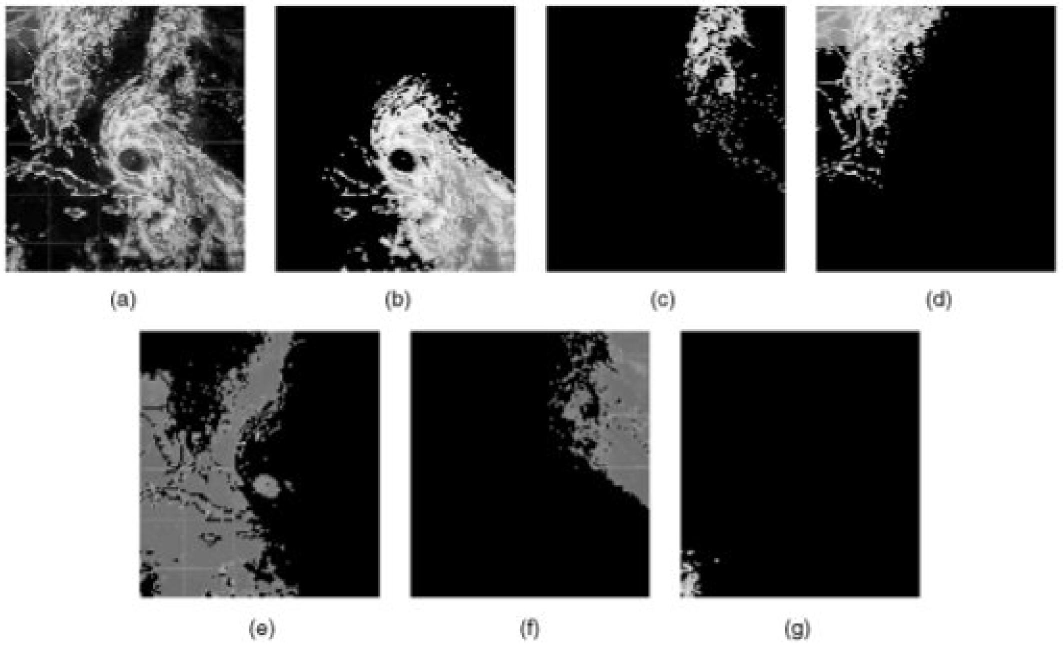


Figure 1.7: Segmentation using the normalized cut method. (a) shows a 126×106 weather radar image. (b)-(g) show the components of the partition with $Ncut$ value less than 0.08. This image originally appeared in [57].

1.7 Nonlocal Methods in Image Processing

In [5] Buades, Coll and Morel introduce a non-local filter for image denoising. The idea of non-local filtering comes from non-local averaging, the technique where averaging is applied based on similarity between the pixels, rather than inside a particular neighborhood. Non-local methods provide a continuous framework for the graph methods we describe in Section 1.6. This method, unlike its local counterparts, performs well in the case of textured and periodic images. The non-local filter proposed in [5] is:

$$NL(u)(x) = \frac{1}{c(x)} \int_\Omega e^{-\frac{d_a(u(x),u(y))}{h^2}} u(y) dy, \tag{1.36}$$

where

$$d_a(u(x), u(y)) = \int_\Omega G_a(t) \mid u(x+t) - u(y+t) \mid^2 dt, \tag{1.37}$$

G_a is a Gaussian with standard deviation a, $c(x)$ is normalized factor and h is a parameter:

$$c(x) = \int_\Omega e^{-\frac{d_a(u(x),u(y))}{h^2}} dy. \tag{1.38}$$

The discrete formulation of (1.36):

$$NL(u)(i) = \sum_j \alpha(i,j) u(j), \tag{1.39}$$

where

$$\alpha(i,j) = \frac{1}{c(i)} e^{-\frac{\|u(B_i)-u(B_j)\|^2}{h^2}}, \tag{1.40}$$

$u(B_i) = (u(k) : k \in B_i)$, where B_i is a small patch around pixel i.

In [31] Gilboa and Osher study the following non-local functional as a way to generalize (1.36):

$$J(u) = \frac{1}{4} \int_{\Omega \times \Omega} (u(x) - u(y))^2 w(x,y) dx dy \tag{1.41}$$

where $\Omega \subset \mathbb{R}^2$ and w is a weight function. We define $w(x,y) \in \Omega \times \Omega$ as positive and symmetric. The authors create a method for image denoising with a functional from (1.41) used as a regularizer. Thus, the type of regularization depends on the choice of w. The gradient descent of the functional (1.41) yields the following equation:

$$u_t(x) = -J'(u)(x) = -\int_\Omega (u(x) - u(y)) w(x,y) dy. \tag{1.42}$$

The new functional Lu is defined as:

$$Lu(x) = \int_\Omega (u(y) - u(x)) w(x,y) dy. \tag{1.43}$$

Assuming f is the noisy input image, the authors construct the energy functional:

$$E(u, f) = J(u) + \frac{\lambda}{2}\|u - f\|^2 \tag{1.44}$$

and minimize this energy in order to find a solution. From (1.44) we have that the solution, u, satisfies the Euler-Lagrange equation

$$u_t = Lu + \lambda(f - u). \tag{1.45}$$

To calculate the weights $w(i, j)$ the authors of [31] suggest a method based on similarity of the feature vectors $F_f(i)$ and $F_f(j)$. Each feature vector contains certain information on the image, particularly on the region surrounding the pixel, such as intensity values, edge indicator, dominant frequency, etc. Usually, for the sake of computational speed of the algorithm, the weights $w(i, j)$ take non-zero values only if j is inside a designated $\Omega_w(i)$ area around i. The weights are defined as:

$$w(i, j) = \begin{cases} g(F_f(i), F_f(j)) & j \in \Omega_w(i) \\ 0 & \text{otherwise} \end{cases} \tag{1.46}$$

where $g(s_1, s_2)$ is a similarity function with the following properties:

1. Positive, $g(s_1, s_2) > 0$.
2. Symmetric $g(s_1, s_2) = g(s_2, s_1)$.
3. Bounded, $g(s_1, s_2) \leq M < \infty$.
4. Maximal at equality, $g(s_1, s_1) \geq g(s_1, s_2), \forall s_1, s_2$.

One such function, often used in nonlocal applications is

$$g(s_1, s_2) = e^{-\frac{\|s_1 - s_2\|^2}{h^2}} \tag{1.47}$$

where h is a parameter.

To implement the algorithm the following forward Euler discretization of (1.45) is used

$$u_i^{n+1} = u_i^n + \Delta t \sum_{j \in \Omega_w(i)} w(i,j)(u_j^n - u_i^n) + \lambda \Delta t (f_i - u_i^n) \tag{1.48}$$

with the timestep restriction

$$1 \geq \Delta t\Big(\sum_{j \in \Omega_w(i)} w_{i,j} + \lambda \Big). \tag{1.49}$$

The presented algorithm from 1.48 can be adapted for inpainting problems. The comparison of the models from [5] and [31] is given in Figure 1.8.

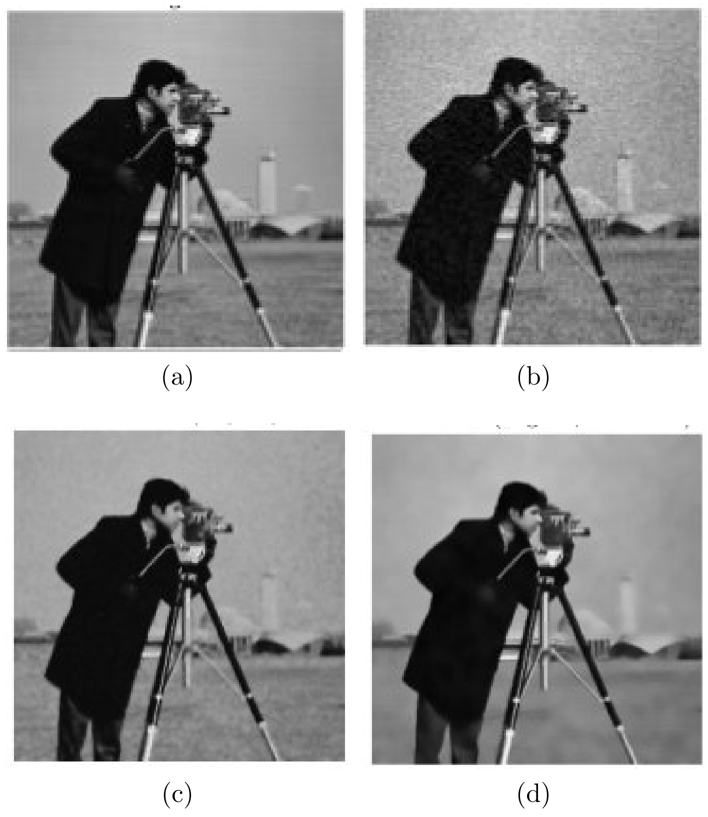


Figure 1.8: Denoising using nonlocal means: Image (a) presents the original image. Image (b) is the noisy input image. Image (c) is the solution obtained by the original non-local method from [5], $PNSR = 14.59$. Image (d) is the solution obtained by the method from (1.48), $PNSR = 16.32$.

1.8 Diffuse Interface Models on Graphs

In graph theory, a graph is commonly represented as an ordered pair $G = (V, E)$ where V are vertices and E are edges. A function on a graph G, u is defined as a set of values in all vertices from V. Function u can be used to define a weighed graph on the structure G. The weights of the edges are determined using an appropriate similarity function. The role of the similarity function will be explained in 3.2.4. In their paper [8] Bertozzi and Flenner introduce diffuse interface models in a graph framework by proposing a graph version of the Ginzburg-Landau functional:

$$GL(u) = \epsilon u \cdot L_s u + \frac{1}{\epsilon} \int W(u) dx \tag{1.50}$$

where u is a graph function $L_s u$ is a symmetric graph Laplacian that will be described in more detail in 3.2.2. The authors of [8] propose a classification algorithm by minimizing the graph Ginzburg-Landau function with a fidelity term:

$$E(u) = \epsilon u \cdot L_s u + \int |\nabla u|^2 dx + \frac{1}{\epsilon} \int W(u) dx + F(u, u_0). \tag{1.51}$$

The functional is minimized using the method of gradient descent resulting in the following equation:

$$\frac{\partial u}{\partial t} = -\epsilon L_s u - \frac{1}{\epsilon} W'(u) - \frac{\partial F}{\partial u}. \tag{1.52}$$

Note that this is just the Allen-Cahn equation with a fidelity term where Δu is replaced by a graph operator term $-L_s$. Choosing the fidelity term F to be $\frac{1}{2} C\lambda(x)(u - u_0)^2$ for some constant C, one obtains

$$\frac{\partial u}{\partial t} = -\epsilon L_s u - \frac{1}{\epsilon} W'(u) - C\lambda(x)(u - u_0). \tag{1.53}$$

Motivated by other minimization schemes for the continuous Ginzburg-Landau functional, the authors of [8] propose a convex splitting minimization scheme. The convex splitting is commonly used in other image processing models that are built around the Ginzburg-Landau functional and it involves writing the energy functional as a sum of the implicit convex and the explicit concave part. The results from [8] show that this method outperforms many state-of-the-art classification such as Bayesian decision trees from [50] and spectral clustering using the p-Laplacian from [61] and [9]. In 3.4 we present a new numerical scheme for solving (3.3). Unlike the scheme from [8], our scheme does not require convex splitting. In Section 3.4.2, we compare the results of the method from [8] and our method, where we show that our method generates results of comparable quality, but significantly faster.

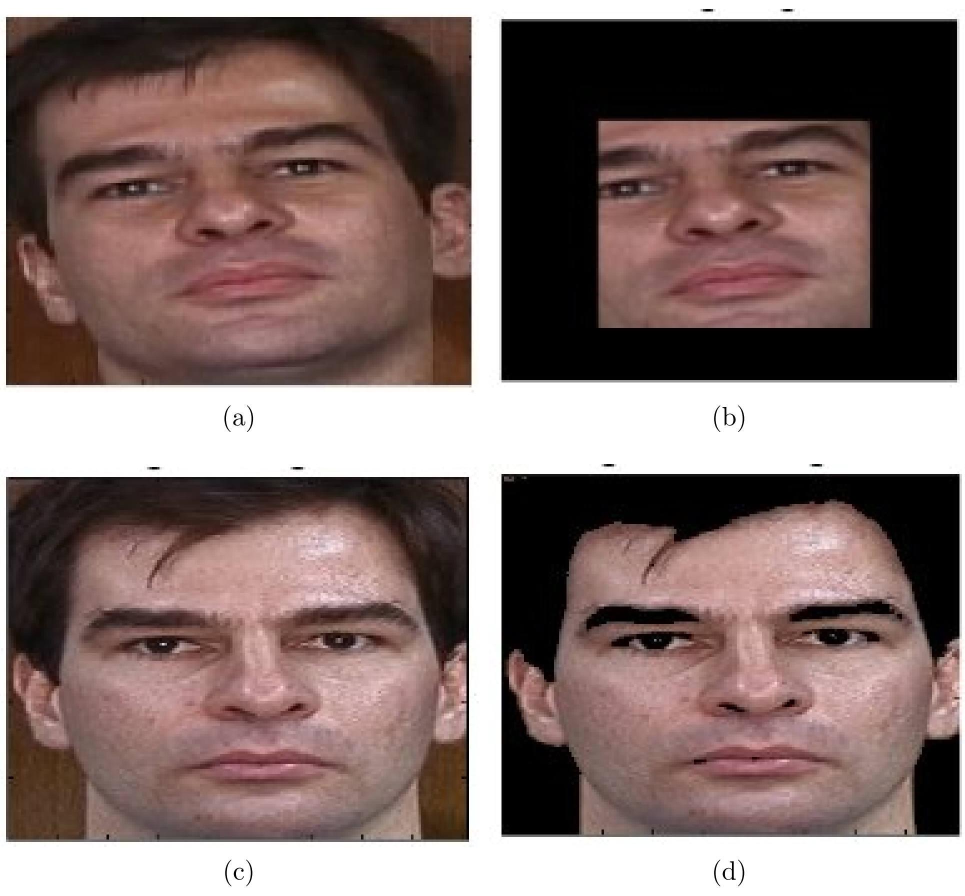


Figure 1.9: An example of the image labeling from [8]. The results indicate robustness of the algorithm to lightning condition and changes in texture Image (a) is the original image Image (b) is the labeled training region. Image (c) is the image that is supposed to be labeled. Image (d) is the labeling result obtained by the algorithm from [8]. This image originally appeared in [8].

1.9 Outline

This thesis is structured in the following way: 2 is dedicated to the MBO method for density estimation while in 3 we present the MBO scheme for data labeling. In sections 2.1 we give an overview of our method for density estimation. In 2.2 we introduce some of the recent work on the MBO schemes. In 2.3 and 2.4 we give the details of the two models for density estimation. In 2.5 and 2.6 we explain the numerical implementation of our method. In 2.7 we show or computational results. In 3.1 we give an overview of the different techniques we use in our work on data labeling. In 3.2 we present a short graph theory background. In 3.3 we discuss the nonlocal operators in more detail. In 3.4 we explain our method for data labeling and we show our results. In 3.5 we present our inpainting algorithm, along with our results for inpainting.

CHAPTER 2

Threshold Dynamics for Density Estimation

2.1 Introduction of the proposed method

Our goal is to estimate a probability density based on discrete point data via segmentation techniques. The idea to use binary segmentation techniques for density estimates arises from the need to quickly and accurately locate the regions of higher activity, as well as to calculate the local density. Instead of obtaining a complete density function estimate, such as seen in [26] and [39], our goal is to segment only one region at a time. Since point data may represent certain activities, such as crime, our method can be successfully used for detecting regions of high activity. In this work we design a binary segmentation version of the well-known Maximum Penalized Likelihood Estimation (MPLE) model, as well as a minimization algorithm based on thresholding dynamics originally proposed by Merriman, Bence and Osher [7]. We also present some computational examples, including one with actual residential burglary data from the San Fernando Valley.

2.2 Esedoglu-Tsai method

We find the modification of the MBO scheme proposed in [23] by Esedoglu and Tsai particularly interesting. In their paper [23], the authors proposed a diffuse interface version of the Mumford-Shah energy functional

$$\min_{\substack{\Sigma\subseteq\Omega \\ c_1,c_2\in\mathbf{R}}} E(\Sigma, c_1, c_2) = Per(\Sigma;\Omega) + \lambda\int_\Sigma (c_1 - f)^2 dx + \lambda\int_{D\setminus\Sigma} (c_2 - f)^2 dx, \tag{2.1}$$

$$MS_\epsilon(u, c_1, c_2) = \int_D \epsilon|\nabla u|^2 + \frac{1}{\epsilon}W(u) + \lambda\{u^2(c_1 - f)^2 + (1-u)^2(c_2 - f)^2\}dx \qquad (2.2)$$

for segmenting the image f. The first variation of the model (2.2) yields the following gradient descent equation:

$$u_t = 2\epsilon\Delta u - \frac{1}{\epsilon}W'(u) + 2\lambda\{u(c_1 - f)^2 + (1-u)(c_2 - f)^2\} \qquad (2.3)$$

and an adaptation of the MBO scheme was used to solve it. Esedoglu and Tsai proposed the following scheme:

- **Step 1** Let $v(x) = S(\delta t)u_n(x)$ where $S(\delta t)$ is a propagator by time δt of the equation:

  $$w_t = \Delta w - 2\tilde{\lambda}\left(w(c_1 - f)^2 + (1-w)(c_2 - f)^2\right)$$

  with appropriate boundary conditions.

- **Step 2** Set

  $$u_{n+1}(x) = \begin{cases} 0 & \text{if } v(x) \in (-\infty, \frac{1}{2}], \\ 1 & \text{if } v(x) \in (\frac{1}{2}, \infty). \end{cases}$$

Based on computational experiments, the authors presented their conclusions about the choice of δt timestep in the step **1.** of the algorithm. If δt is chosen too large compared to the parameter λ^{-1}, the interface tends to be overly smooth. An advantage of choosing larger values for δt is faster convergence. The segmentation could also benefit from the larger values for the parameter λ, as there is less penalty on high curvature of the interface in this case. The size of the spatial resolution has to be taken into account when the value for δt is chosen, as values much smaller than the spatial resolution could lead to the pinning of interface. These observations can also be used as guidance in parameter selection for our algorithm. The authors also show that MBO thresholding type methods for binary segmentation can easily be generalized to multi-phase segmentation methods. To accomplish that, the authors propose the four phase model based on the modified version of (2.2) with the sum of the two Ginzburg-Landau functionals built around two different segmentation functions, u_1 and u_2. In this case, the fidelity term naturally depends on both u_1 and u_2. After they find the gradient descent equations with respect to u_1 and u_2, they construct the thresholding numerical scheme to solve the obtained system of parabolic equations. We will adapt ideas by Esedoglu and Tsai to solve our MPLE problem and illustrate the usefulness of this simple method. Some extension of the MBO algorithms appeared in [54, 53, 45]. An efficient algorithm for motion by mean curvature using adaptive grids was proposed in [51].

2.3 General Model

For now we are going to focus on the segmentation function u. We assume our segmentation function is the characteristic function of the region Σ, where Σ is an area with a larger density. For any given data and any given segmentation function we obtain an explicit formula for the density. With w denoting the data, the total number of events is equal to $\int w$, while the number of events inside and the number of events outside of the region Σ are simply $\int wu$ and $\int w(1-u)$, respectively. Thus, the density $c_1(u)$ inside the region Σ is equal to

$$\frac{\int wu}{\int u \int w}$$

and the density $c_2(u)$ in the region Σ^C is equal to

$$\frac{\int w(1-u)}{\int w \int (1-u)}$$

. Finally, we write the density function as $c_1(u)u + c_2(u)(1-u)$. The established correspondence between the segmentation and the density function suggests that building a diffuse interface MPLE model around the segmentation function is possible. As the segmentation function takes only 0 and 1 values, the Ginzburg-Landau functional is a natural choice. Since the density is a rescaled segmentation function, using the Ginzburg-Landau functional for u, as opposed to the Ginzburg-Landau functional for the density seems both reasonable and convenient. In other words, we search for an appropriate u and the density values are defined automatically.

Here we propose the diffuse interface Maximum Penalized Likelihood Estimation model.

$$\arg\min_{u(x)} \int \epsilon|\nabla u|^2 + \frac{1}{\epsilon}W(u)dx - \mu \int w \log(c_1(u)u + c_2(u)(1-u))dx, \tag{2.4}$$

where $c_1(u) = \frac{\int wu}{\int u \int w}$ and $c_2(u) = \frac{\int w(1-u)}{\int w \int (1-u)}$,$w$ represents the given data, $W(u) = u^2(1-u)^2$ and μ is a parameter. As we already mentioned, the Ginzburg-Landau functional converges to the TV norm in the sense of Γ convergence, as $\epsilon \to 0^+$. As a consequence, the diffuse interface model we propose here converges to the TV-based MPLE that was used in [26],[39], when $\epsilon \to 0^+$. Now, variation of energy of (2.4) gives us the following cases for the L^2 gradient descent equation:

- If both and $c_1(u)$ and $c_2(u)$ (further we use c_1 and c_2 instead for simplicity of the notation) are non-zero:

$$u_t = 2\epsilon\Delta u - \frac{1}{\epsilon}W'(u) + \mu w[\frac{c_1 - c_2}{c_1}u + \frac{c_1 - c_2}{c_2}(1-u) + (\frac{\int (1-u)w}{\int 1-u} - \frac{\int uw}{\int u})]. \tag{2.5}$$

- If c_1 is equal to zero:

$$u_t = 2\epsilon\Delta u - \frac{1}{\epsilon}W'(u) + \mu[w(u-1) + (\frac{\int (1-u)w}{\int 1-u} - w)]. \tag{2.6}$$

- If c_2 is equal to zero

$$u_t = 2\epsilon\Delta u - \frac{1}{\epsilon}W'(u) + \mu[wu + (w - \frac{\int uw}{\int u})]. \tag{2.7}$$

2.4 Special Case Model

For the special case problem the density in a high region Σ^C is zero, so the density function is just a rescaled characteristic function. Thus, replacing the density function $c_1(u)u+c_2(u)(1-$

u) in the MPLE term of our general model by the segmentation u seems reasonable. Now, the model we propose is:

$$\arg\min_{u(x)} \int \epsilon|\nabla u|^2 + \frac{1}{\epsilon}W(u)dx - \mu \sum_{i,j} w_{i,j}\log(u_{i,j}). \tag{2.8}$$

Once again, assuming w is a function that approximates the data, in order to make everything well-defined we introduce the model with a small constant ν:

$$\arg\min_{u(x)} \int \epsilon|\nabla u|^2 + \frac{1}{\epsilon}W(u)dx - \mu \int w\log((1-\nu)u + \nu(1-u))dx. \tag{2.9}$$

The Euler-Lagrange equation for the energy functional (2.9) is:

$$u_t = 2\epsilon\Delta u - \frac{1}{\epsilon}W'(u) + \mu w(\frac{1}{1-\nu}u + \frac{1}{\nu}(1-u))(1-2\nu). \tag{2.10}$$

2.5 Proposed dynamics

We use the following gradient descent equation

$$u_t = 2\epsilon\Delta u - \frac{1}{\epsilon}W'(u) + A(u(\cdot,t))u + B(u(\cdot,t)) \tag{2.11}$$

with $A(u(\cdot,t))$ and $B(u(\cdot,t))$ being non-linear functions. Each of the gradient descent equations, (2.5), (2.6), (2.7) and (2.10) can be given in the form (2.11), where $A(u(\cdot,t))$ and $B(u(\cdot,t))$ take different vales in different cases.

$A(u(\cdot,t))$	$B(u(\cdot,t))$	Equation
$\mu w\frac{(c_1-c_2)^2}{c_1c_2}$	$\mu w\frac{c_1-c_2}{c_2} + (c_2-c_1)\int w$	(2.5)
μw	$\mu(c_2\int w - 2w)$	(2.6)
μw	$\mu(w - c_1\int w)$	(2.7)
$-\mu w\frac{(2\nu-1)^2}{(1-\nu)\nu}$	$\mu w\frac{1-2\nu}{\nu}$	(2.10)

Motivated by the MBO scheme for solving the Allen-Cahn equation, we propose a thresholding scheme to approximate the solution of the equation (2.11). Esedoglu and Tsai used a similar approach to minimize the Mumford-Shah segmentation functional in [23]. The first step we need to take toward generating a thresholding scheme is finding a good way to split the equation (2.11) into two steps analogous to those proposed in the MBO scheme. In that regard, finding a way that successfully deals with the non-linear forcing term of equation (2.11) is critical. Inspired by the algorithm presented in [23] we propose the following dynamics:

- **Step 1.** Let $v(x) = S(\delta t)u_n(x)$ where $S(\delta t)$ is a propagator by time δt of the equation:

  $$y_t = \Delta y - A(y(\cdot, t))y + B(y(\cdot, t))$$

  with appropriate boundary conditions.

- **Step 2.** Set

  $$u_{n+1}(x) = \begin{cases} 0 & \text{if } v(x) \in (-\infty, \frac{1}{2}] \\ 1 & \text{if } v(x) \in (\frac{1}{2}, \infty). \end{cases}$$

2.6 Numerical implementation

In the propagation phase of the algorithm we solve the following PDE:

$$v_t = \Delta v - A(v(\cdot, t))v + B(v(\cdot, t))$$

with the u_n being an initial condition. To generate the numerical results we denote the timestep by $\delta\tau$ is a timestep, and approximate the Laplacian by its five-point stencil, which gives us the following scheme:

$$\frac{v^{n+1} - v^n}{\delta\tau} = \Delta v^{n+1} + A(v^n(\cdot, t))v^n + B(v^n(\cdot, t)).$$

As the linear linear system described above is circulant we find the Fast Fourier transform (FFT) a convenient way solve it. Other solvers with even better computational complexity than the FFT, such as a multigrid algorithm, could be used. In the propagation phase our goal is not to achieve steady state, and iterations are repeated only until the total propagation time reached δt. After the propagation phase, a thresholding step is necessary to complete the iteration:

$$u_{i,j}^{n+1} = \begin{cases} 0 & \text{if } v_{i,j}^{l} \in (-\infty, \frac{1}{2}], \\ 1 & \text{if } v_{i,j}^{l} \in (\frac{1}{2}, \infty). \end{cases}$$

where l is a total number of iterations we made in the propagation phase. The small relative change of the L^2 norm between two consecutive iterations was used as a stopping criterion. In this implementation, the data function w is used as an initial condition, along with Dirichlet or Neumann boundary conditions.

2.6.1 Adaptive timestepping

The choice of timestep in the propagation phase, a "sub-timestep", can be chosen to optimize performance. In the early stage of computation, it is important to keep the sub-timestep small in order to obtain a good estimate in the propagation phase. However, as our algorithm is approaching steady state, a large number of iterations in the propagation phase pose a burden on the computational time. To successfully speed up the convergence of our algorithm, we used adaptive timestepping, a modified form of the scheme proposed in [4]. The scheme uses a dimensionless local truncation error calculated in every iteration, and the timestep is increased when the error is smaller then a chosen tolerance.The error at time t^n uses solution at three consecutive timesteps t^{n-1}, t^n and t^{n+1}.Let us define $e^{n+1} = \frac{(v^{n+1}-v^n)}{v^n}$ and $e^n = \frac{(v^n - v^{n-1})}{v^n}$, as well as $\Delta t_{old} = t^n - t^{n-1}$. The previous definitions allow us to define a dimensionless estimate of the local truncation error:

$$\left\| e^{n+1} - \frac{\Delta t}{\Delta t_{old}} e^n \right\|_{L^2}. \tag{2.12}$$

This algorithm is both quick and practical. Even though other more accurate ways to estimate the measure of error are available, such as step doubling, they are often much more computationally expensive.

We used adaptive timestepping at two different levels, in the propagation phase of the algorithm we adapt the sub-timestep, as well as adapting an initial sub-timestep for the future iterations. In the propagation phase of any iteration, we calculate a dimensionless truncation error estimate for different propagation times. Once an error is smaller than a given tolerance Tol_1 for a certain number of the consecutive iterations, we increase the timestep by 10%. We also estimate the dimensionless error in every iteration of the algorithm, and if we find an error to be smaller than Tol_2 the initial sub-timestep in the propagation phase of the next iteration will be increased be 10%. However, we never allow the initial sub-timestep to be larger than $\frac{1}{8}$ of the timestep. Notice that we are not adapting the timestep, the total propagation time in each iteration is the same.

2.6.2 Adaptive resolution

Another way to improve the computational time is to use adaptive resolution. As we mentioned before, we use the data function w as an initial condition when solving the equation (2.11). It is reasonable to assume that the more the initial condition "resembles" the solution, the less iterations the algorithm would take to obtain the solution. The main idea is to generate a lower resolution form of the data set, then use a low resolution solution to create a good initial guess for the high resolution solution. Providing a good initial guess for the higher resolution problem is particularly useful as the iterations when the algorithm is applied to the higher resolution versions of the data set tend to be slower. In this implementation, we typically applied this procedure several times on some sparse data sets. At each step we create the coarser form of the given data set, until we reach the version of the data set that has a satisfying density. Our experiments show that data sets with the total density between 0.05 and 0.2 are optimal for this algorithm. Once a sufficiently dense low resolution version of the data set is obtained, we run our algorithm to get the low resolution

solution, and start working our way up from there. The higher resolution approximation of the solution is then generated, and used as an initial condition in the next step. In the next step, we are solving the problem on the data set that has a higher resolution. It is important to mention that this process does not alter the original data set. We call this process n-step adaptive resolution where n is the total number of times we reduced the resolution of the original data set. The number of steps, n, is closely related to our choice of timestep. In case we are segmenting the region of higher density in our data, we noticed, through multiple experiments, that the timestep often can be given as $\omega 2^n$, where n is the number of levels in adaptive resolution, and $\omega \in [0.15, 0.2]$. In case we are locating the valid region, we usually allow a smaller timestep, but also a larger number of levels in adaptive resolution. However, starting with a problem that has a significantly lower resolution comparing to the original one, we might run into some problems. Decreasing resolution significantly may result in a very different looking data set, thus segmentation would not perform in an expected way, i.e. this first initial guess would not be a good approximation of the solution we are trying to find.

2.7 Computational Examples

2.7.1 Test Shapes

To verify the performance of our algorithm, we constructed three examples of the probability density maps featuring three shapes and they are shown in the figures 2.1(a), 2.2(a) and 2.3(a). Each of the densities were sampled, and the toy data sets are obtained, and presented in the figures 2.1(b) , 2.2(b) and 2.3(b) respectively. We applied the general version of our algorithm to solve these problems, and the results are shown in images 2.1(c), 2.2(c) and 2.3(c). Adaptive timestepping was used in all of the examples with three shapes, while the adaptive resolution was not used, given the small scale of these examples.

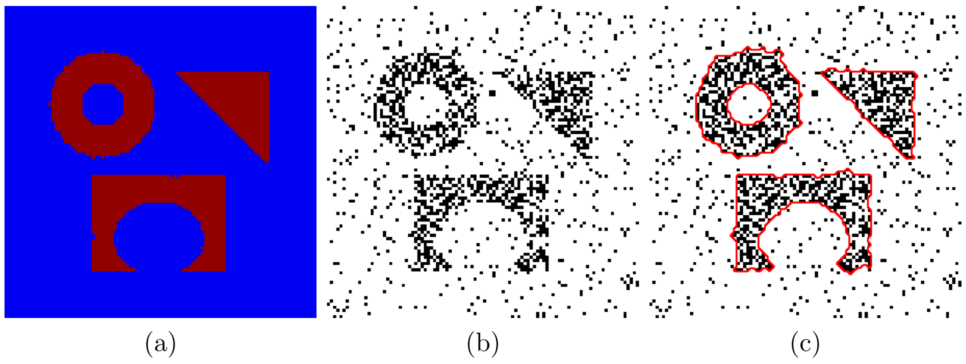


Figure 2.1: Segmentation of three shapes: Image (a) presents a plot of the density map. Assuming the size of (a) is 1×1, the densities of the dense region and the sparse region are 3.3943 and 0.3927, respectively. Image (b) is a plot of the data set, in a 100×100 pixel image, with 1449 events. Image (c) shows the contours of the segmented dense region. In the process of generating this result we used the timestep 1.6 along with the parameter $\mu = 0.13$. The densities of our estimated dense and sparse regions are 3.4811 and 0.3649.

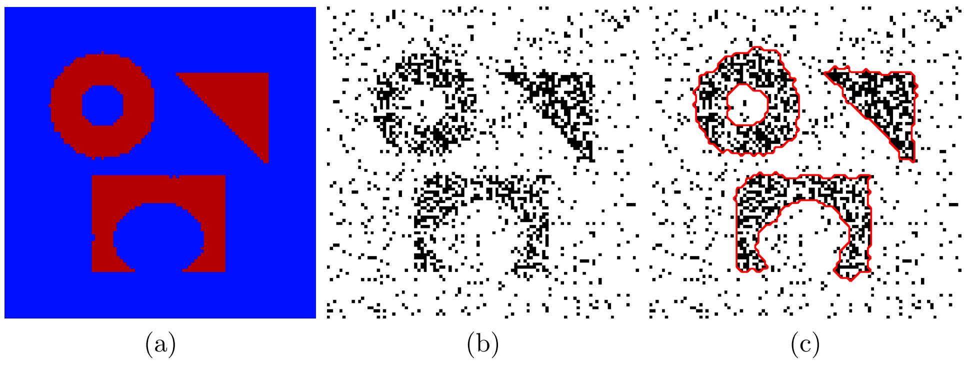


Figure 2.2: Segmentation of three shapes: Image (a) presents a plot of the density map. Assuming the size of (a) is 1×1, the densities of the dense region and the sparse region are 3.145 and 0.456, respectively. Image (b) is a plot of the data set, in a 100×100 pixel image, with 1539 events. Image (c) shows the contours of the segmented dense region. The choice of parameters was 1.6 for the timestep, and $\mu = 0.15$. The densities of our estimated dense and sparse regions are 3.311 and 0.422.

2.7.2 Orange County Coastline

In their work [39] Smith et al. constructed several test data sets using the spatial information of the Orange County coastline obtained from Google Earth. The authors determined the boundary between the valid and the invalid region by locating the ocean, rivers, parks and

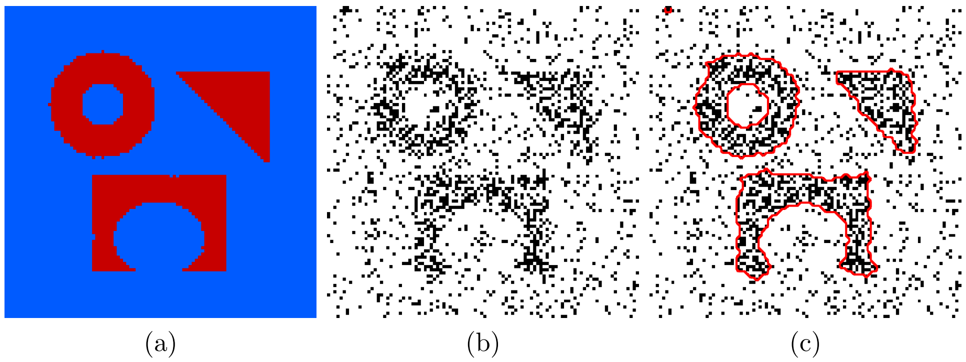


Figure 2.3: Segmentation of three shapes: Image (a) presents a plot of the density map. Assuming the size of (a) is 1×1, the densities of the dense region and the sparse region are 2.605 and 0.592, respectively. Image (b) is a plot of the data set, in a 100×100 pixel image, with 1696 events. Image (c) shows the contours of the segmented dense region. The choice of parameters was 1.6 for the timestep, and $\mu = 0.1$. The densities of our estimated dense and sparse regions are 2.780 and 0.573.

other features the valid region of a residential crimes map naturally excludes. Then, inside the valid region, they constructed three other regions and assigned different densities to them, which is shown on the density map in Figure 2.10, and we also, inspired by their examples, constructed the density maps shown in Figure 2.4 as well as in Figure 2.7. Sampling from these density functions we generated data sets shown in Figures 2.5(a), 2.6(a), 2.8(a), 2.9(a) and 2.11(a). A segmentation of the corresponding dense region is shown next to each data set. Note that there are four different levels of densities in Figure 2.10. The solution in Figure 2.11 shows that the different choice of the parameter μ can lead to segmentation of the dense regions at different levels. All images in this section have resolution of 600×1000 pixels.

2.7.3 San Fernando Valley Residential Burglary Data

We may also consider a special case of our problem, the case when we assume all events are located inside the region Σ, and our goal is to segment the region. The region Σ would represent a valid region of the given data set. In absence of geographic data that describe the location of the valid region, an accurate estimate of it can dramatically improve accuracy

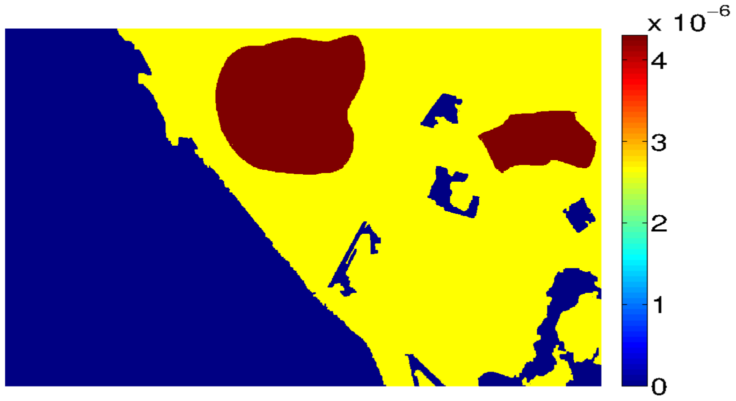


Figure 2.4: The toy density map of the Orange County Coastline featuring two disconnected dense regions.

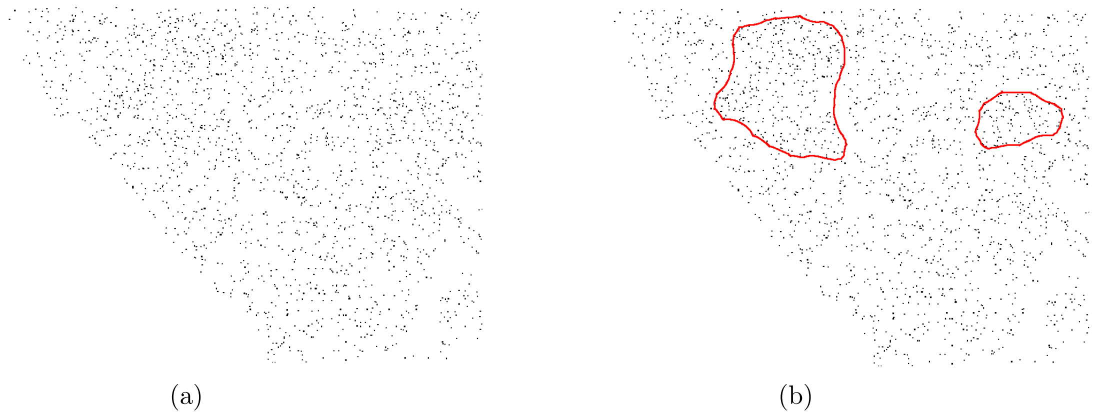


Figure 2.5: A 2000 event sample of the density function from Figure 2.4 is shown in (a). The contour of the segmented dense region is shown in (b). The choice of parameters is: $\mu = 0.1$, time step is 13.0 with the 3-step adaptive resolution. The data points in this figure are manually enhanced for the purpose of more clear display of the image.

of the density estimation, see [39] and [26]. In the following example, our goal was to, without using any spatial information, segment the valid region from the San Fernando Valley residential burglary data. The events in Figure 2.12(a) represent locations where burglaries took place during 2004 and 2005. The contour of our valid region estimate obtained by applying the special case model is also shown in Figure 2.12(a). Smith et al. in [39] performed the valid region estimate using census and other types of data to locate the residential area in the region of interest, and their result is Figure 2.12(b). They incorporated the valid region estimate from Figure 2.12(b) in their Weighted H^1 MPLE model to obtain

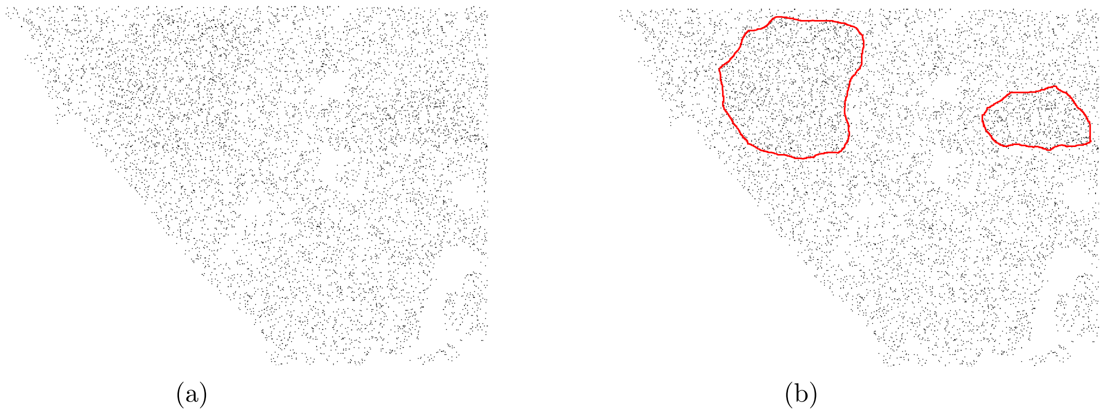


Figure 2.6: An 8000 event sample of the density function from Figure 2.4 is shown in (a). The contour of the segmented dense region is shown in (b). The choice of parameters is: $\mu = 0.092$, time step is 8.0 with the 2-step adaptive resolution. The data points in this figure are manually enhanced for the purpose of more clear display of the image.

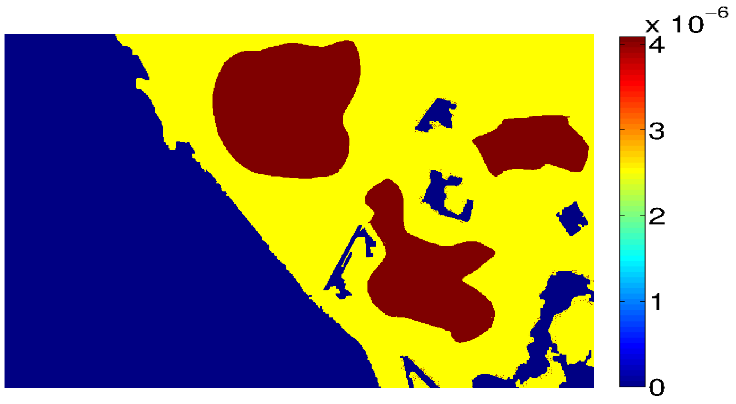


Figure 2.7: The toy density map of the Orange County Coastline featuring three disconnected dense regions.

the density estimate results from 2.12(c). The TV MPLE algorithm developed by Mohler et al. in [26], was used to generate the density estimate in 2.12(b). This method did not use any additional spatial information to locate the valid region.

2.8 V-fold cross validation

The choice of the smoothing parameter μ and the timestep can affect the performance of this algorithm. Our experiments show that in case we are segmenting the high density region

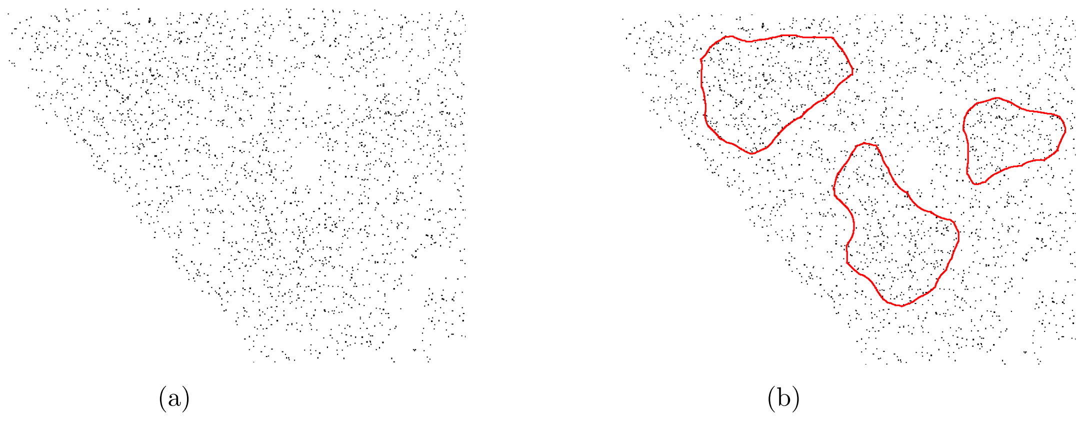


Figure 2.8: A 2500 event sample of the density function from Figure 2.7 is shown in (a). The contour of the segmented dense region is shown in (b). The choice of parameters is: $\mu = 0.11$, timestep is 11.0 with the 3-step adaptive resolution. The data points in this figure are manually enhanced for the purpose of more clear display of the image.

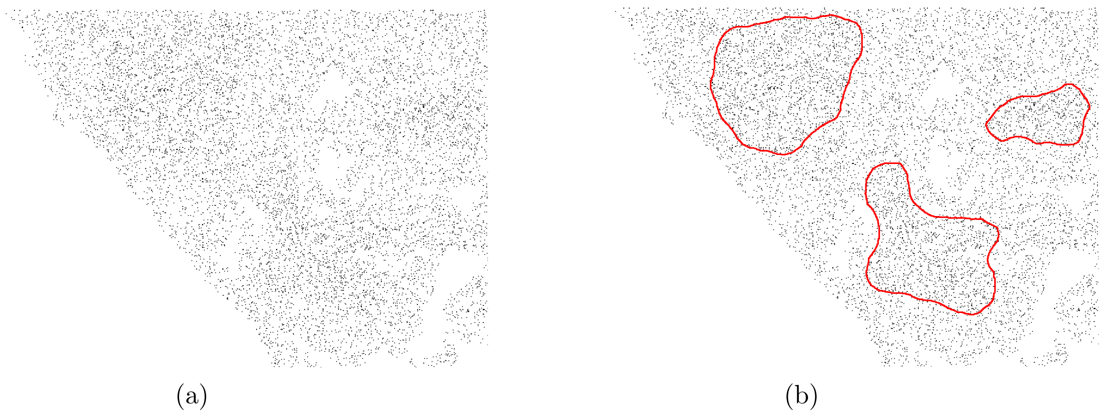


Figure 2.9: A 10000 event sample of the density function from Figure 2.7 is shown in (a). The contour of the segmented dense region is shown in (b). The choice of parameters is: $\mu = 0.075$, timestep is 7.5 with the 2-step adaptive resolution. The data points in this figure are manually enhanced for the purpose of more clear display of the image.

the optimal value of the parameter μ is not larger than 0.2. When our goal is to estimate the valid region, we typically assign larger values to the parameter μ. To estimate the value of the smoothing parameter we implemented a version of the V-fold cross validation algorithm. In their work [55] Sardy and Tseng proposed the V-fold cross validation based on the Kullback-Leibler information. In the V -fold cross validation, the original data set is partitioned into V disjoint subsets $\mathbf{x}_v = \{x_i, i \in S_v\}$ where S_v consists of all indexes of

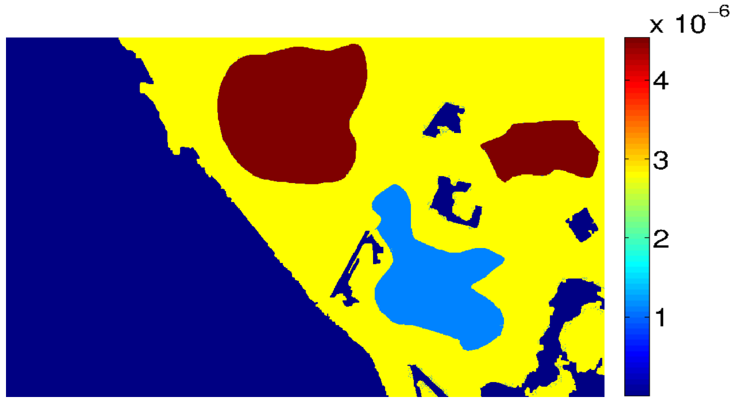


Figure 2.10: The toy density map of the Orange County Coastline, featuring two disconnected dense regions and one sparse region.

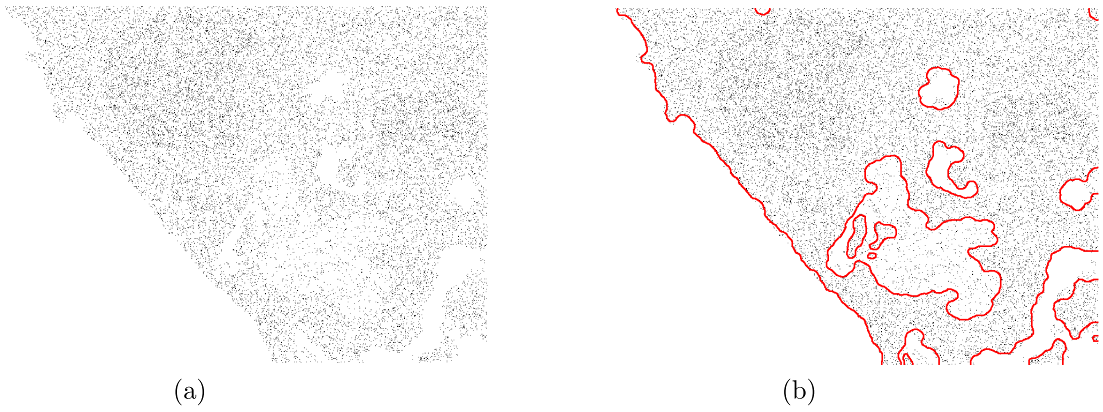


Figure 2.11: A 20000 event sample of the density function from Figure 2.10 is shown in (a). The contour of the segmented dense region is shown in (b). The choice of parameters is: $\mu = 0.03$, timestep is 2.2 with the 2-step adaptive resolution.

data points from the partition $v = 1, \ldots V$. Set $\mathbf{x}_{-v} = \{x_i, i \notin S_v\}$ is used as a training set, i.e. the algorithm is applied on $\mathbf{x}_{-v}$ with some particular value $\hat{\mu}$, and the density $\hat{d}_{\hat{\mu},-v}$ is estimated. Set $\mathbf{x}_v$ is a validating set, which means that $\{\hat{d}_{\hat{\mu},-v}(x_i)\}_{i \in S_v}$ is used to estimate the density on $\mathbf{x}_v$. Following these observations, the authors of [55] proposed the following estimate of Kullback-Leibler information $CV(\hat{\mu}) = -\sum_{v=1}^{V} \sum_{i \in S_v} \hat{d}_{\hat{\mu},-v}(x_i)$, and after the search of the set of parameters $\hat{\mu}$ the one that minimizes this quantity is selected. However, $CV(\hat{\mu})$ uses only the log-likelihood to predict the performance of the model for some value $\hat{\mu}$ of the smoothing parameter, but does not take the H^1 norm of the segmentation function of the estimated density into account. We denote the segmentation function that corresponds

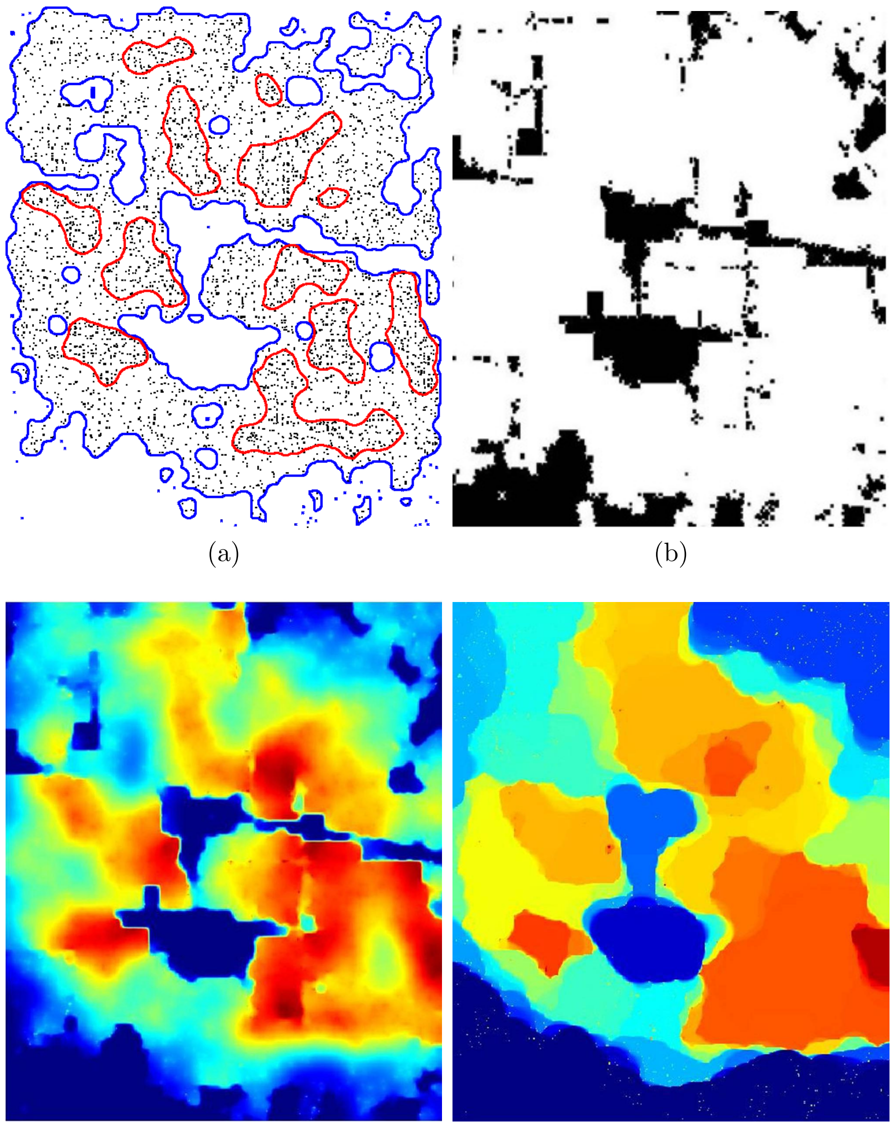

(a) (b) (c) (d)

Figure 2.12: San Fernando Valley residential burglary: (a) A plot of the data set with the contours of the solution, the red contour bounds the dense region, while the blue one bounds the valid region . The size of the original data set is 605×525 pixels. We used a timestep 2.0 with $\mu = 1.1$ and 3-step adaptive resolution to find the valid region, and $\mu = 0.08$, a timestep 3.5 with 1-step adaptive resolution to locate the dense region. The data points in this figure are manually enhanced for the purpose of more clear display of the image. (b) A valid region estimate from [39]. (c) and (d) are density estimates form [39] and [26] respectively.

to the density $\hat{d}_{\hat{\mu},-v}$ by $\hat{u}_{\hat{\mu},-v}$. Since the segmentation function is a binary function, the descrete H^1 and the descrete TV norm are equivalent, thus the H^1 norm also measures the length of the front between two phases. In some applications, such as the case when the dense and the surrounding sparse region have similar densities, the values of $CV(\mu)$ for the different values of μ tend to be similar. It is useful in those cases to also measure the H^1 norm of the obtained segmentation $\hat{u}_{\hat{\mu},-v}$, and incorporate that information in the V-fold cross validation. Because of that, we propose a slightly different technique, where we evaluate $CV_{H^1}(\hat{\mu}) = -\sum_{v=1}^{V}(\sum_{i \in S_v} \hat{d}_{\hat{\mu},-v}(x_i) - \xi \int |\nabla \hat{u}_{\hat{\mu},-v}|^2)$ for each value of $\hat{\mu}$ from some proposed set of parameters, and select the value that minimizes it. The results do not appear to be very sensitive to ξ, we used small values, comparable to those of $\hat{\mu}$. The evaluation of $CV_{H^1}(\mu)$ for a single value of parameter μ requires V different density estimates, which could cause the V-fold cross validation to be very computationally intense. However, all density estimates that have to be performed are independent of each other, which makes the V-fold cross validation a perfect candidate for parallelization. In this implementation, we used 10-fold cross validation, and the process of calculating CV_{H^1} is parallelized using 10 threads, which reduces the computational time of one evaluation of CV_{H^1} down to the computational time needed for one density estimate. The computational time one density estimate takes varies from $0.2s$ in the small scale examples (100×100 pixels) to around one minute in the large scale examples (600×1000 pixels). To demonstrate the performance of the proposed algorithm, Figures 2.13 and 2.14 show some computational examples with segmentations generated using a model with the smoothing parameter obtained through 10-fold cross validation. To find the parameter μ we performed the linear search of intervals.

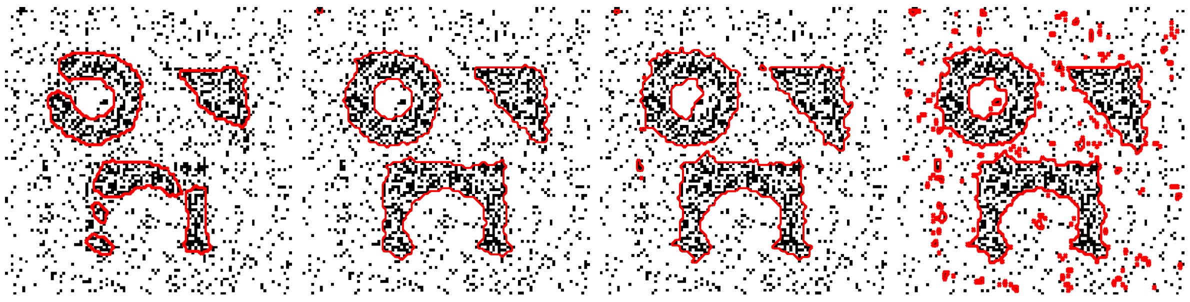

(a) (b) (c) (d)

Figure 2.13: The data set shown in 2.3 and the results obtained using different values of μ:(a) Smoothing parameter is too small $\mu = 0.12$. (b) We used μ obtained through our 10-fold cross validation with the H^1 norm, we evaulated CV_{H^1}. (c) 10-fold cross validation from [55], $CV(\mu)$, was applied to obatain μ used in this example. (d) Smoothing parameter $\mu = 0.3$, the value is larger than optimal.

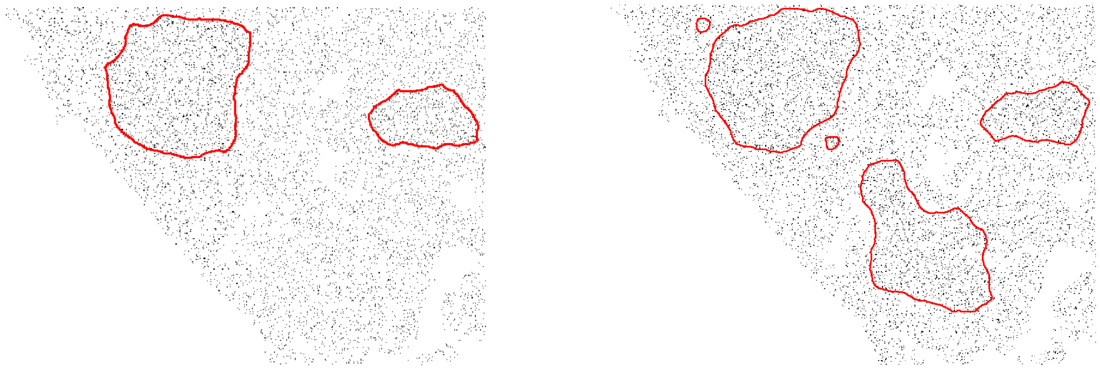

(a) (b)

Figure 2.14: 10-fold cross validation was applied on the data sets displayed in 2.6(a) and 2.9(a) to select the parameter μ. The data sets with the countours of the respective dense regions obtained using those values of μ are shown.

CHAPTER 3

Threshold Dynamics Methods for Data Labeling

3.1 Introduction

In this chapter we present a fast algorithm for a recent variational method in a graph setting. The method is inspired by diffuse interface models that have been used in a variety of problems, such as those in fluid dynamics and materials science. We consider data represented as nodes in a weighted graph, and each edge is assigned a numerical value describing the similarity between the nodes. In spectral graph theory, this approach is successfully used to perform various learning tasks in imaging and data clustering. The standard techniques of the theory are thoroughly described in [14, 46], and the graph Laplacian, which is discussed in more detail in section 3.2, is introduced as one of the fundamental concepts. In imaging, spectral methods are often used in image segmentation applications as shown in [57, 33, 62]. We are particulary interested in nonlocal total variation methods, as they are a link between spectral graph theory and diffuse interface models, and thus can be used as a motivation for our algorithm. These methods are used in numerous image processing applications. They were initially developed as methods for image denoising [5, 31], but were successfully applied to many other image processing problems such as inpainting and reconstruction in [30, 70, 27], image deblurring in [68] and manifold processing in [1].

As an alternative to L^1 compressed sensing methods, Bertozzi and Flenner introduce a graph-based model based on the Ginzburg-Landau functional in their work [8]. The gradient descent of the proposed functional:

$$E(u) = \epsilon u \cdot L_s u + \frac{1}{\epsilon}\int W(u)dx + F(u, u_0). \tag{3.1}$$

is resulting in the following equation:

$$\frac{\partial u}{\partial t} = -\epsilon L_s u - \frac{1}{\epsilon} W'(u) - \frac{\partial F}{\partial u}. \tag{3.2}$$

Function u is a graph generalization of a binary segmentation function we introduced in 1.3.2. Properties of the operator L_s will be explained in more detail in sections 3.2 and 3.3. Taking F to be $\frac{1}{2} C\lambda(x)(u - u_0)^2$ for some constant C, one obtains

$$\frac{\partial u}{\partial t} = -\epsilon L_s u - \frac{1}{\epsilon} W'(u) - C\lambda(x)(u - u_0). \tag{3.3}$$

The main purpose of this work is to develop a fast and simple method for solving (3.3) in the small ϵ limit. To achieve our goal, we created a graph based MBO scheme. Recall that the MBO scheme uses simple threshold dynamics to approximate motion my mean curvature. Since the Allen-Cahn equation is closely related to motion by mean curvature, the MBO scheme has been proven to be a very successful tool in solving different variants of the Allen-Cahn equation. For example, the authors of [23] propose an adaptation of the MBO scheme to minimize he piecewise constant Mumford-Shah functional. Inspired by the efficency and the robustness of the MBO scheme, we decide to adapt it to solve (3.3). However, the implementation of the proposed scheme poses many computational challenges. The quadratic size of the graph Laplacian could make the iterative process of our algorithm very computationally expensive. To reduce the dimension of the graph Laplacian and make the computation more efficient, the authors of [8] propose the Nyström extension method [11] to approximate eigenvalues and the corresponding eigenvectors of the graph Laplacian. To maximize the performance of our algorithm we combine the Nyström extension with the Raleigh-Chebyshev algorithm proposed in [6]. The details of our algorithm are discussed in section 3.4, after the sections 3.2 - 3.3 on the relevant background. In this paper we go beyond the applications presented in [8], and devise a non-local inpainting algorithm. To show the efficiency of our algorithm, we compare the computational times of our algorithm against those obtained by the state-of-the-art nonlocal inpainting algorithm form [30].
Various computational examples are presented to demonstrate the performance of our algo-

rithm, which is successful on images with texture and repetitive structure due to its nonlocal nature.

3.2 Background on Graph Theory

3.2.1 The Origins of Spectral Graph Theory

Algebraic methods in graph theory are as old as the idea to use the adjacency matrix to describe a graph. It seems natural that mathematicians got interested spectral properties of the adjacency matrix, and the interpretation of the spectrum in terms of graph properties. There are several survey books, such as [16] that provide an overview of the vast literature on this subject. Algebraic methods become very successful, especially in dealing with symmetric adjacency matrices, i.e. matrices that describe undirected regular graphs. However, the recent developments in spectral graph theory are often oriented toward geometric methods. Geometric methods are very useful in applications on general graphs, and they successfully complement algebraic methods. The reason for this trend may be found in many known analogies between Riemannian geometry and spectral graph theory. Over the years, spectral methods are becoming increasingly popular, and they have been applied in various areas of mathematics and science. Since its beginnings, spectral graph theory has been applied in chemistry. Also, there are applications in theoretical physics and quantum mechanics.

3.2.2 The Laplacian

Consider an undirected unweighted graph $G = (V, E)$, where V and E are the sets of vertices and edges, respectively. We assume this graph does not contain loops or multiple edges. In an unweighted graph we define d_i, the degree of the vertex i, as the number of edges from

E that are incident with i. Let us define the matrix L:

$$L(i,j) = \begin{cases} d(i), & \text{if } i = j, \\ -1, & \text{if } i \text{ and } j \text{ are adjacent,} \\ 0, & \text{otherwise.} \end{cases} \tag{3.4}$$

In the case of weighted graphs we assume that the edge that connects vertices i and j has weight $w(i,j)$. The degree of the vertex $i \in V$, $d(i)$ is defined as

$$d(i) = \sum_{j \in V} w(i,j). \tag{3.5}$$

The definition of the matrix L can easily be extended to the case of weighted graph:

$$L(i,j) = \begin{cases} d(i), & \text{if } i = j, \\ -w(i,j), & \text{if } i \text{ and } j \text{ are adjacent,} \\ 0, & \text{otherwise.} \end{cases} \tag{3.6}$$

If we define the degree matrix D to be the $N \times N$ diagonal matrix with diagonal elements $d(i)$, then the matrix L can be written in matrix form as $D - W$, where W is the matrix $w(i,j)$. The matrix W is sometimes referred to as the "affinity matrix". Let us now define the symmetric graph Laplacian L_s:

$$L_s(i,j) = \begin{cases} 1, & \text{if } i = j, \\ -\frac{w(i,j)}{\sqrt{d_i d_j}}, & \text{if } i \text{ and } j \text{ are adjacent,} \\ 0, & \text{otherwise.} \end{cases} \tag{3.7}$$

The relationship between L and the symmetric Laplacian is the following:

$$L_s = D^{-\frac{1}{2}} L D^{-\frac{1}{2}} = I - D^{-\frac{1}{2}} W D^{-\frac{1}{2}} \tag{3.8}$$

where W is the adjacency matrix of the graph G. To obtain information on the eigenvalues of the matrix L_s and their bounds we will use the variational characterization:

$$\begin{aligned}
\frac{\langle g, L_s g\rangle}{\langle g, g\rangle} &= \frac{\langle g, D^{-\frac{1}{2}} L D^{-\frac{1}{2}} g\rangle}{\langle g, g\rangle}, \\
&= \frac{\langle f, Lf\rangle}{\langle D^{\frac{1}{2}} f, D^{\frac{1}{2}} f\rangle}, \\
&= \frac{\sum_{i\in V} f(i) Lf(i)}{\sum_{i\in V} f^2(i) d_i}, \\
&= \frac{\sum_{i,j\in V} (f(i) - f(j))^2 w(i,j)}{\sum_{i\in V} f^2(i) d_i}, \qquad (3.9)
\end{aligned}$$

where g is an arbitrary vector and $f = D^{-\frac{1}{2}} g$.

3.2.3 Spectrum of a graph

From equation (3.9) we can see that the smallest eigenvalue of L_s is zero and the rest of them are positive. Also, since L_s is a Hermitian matrix, there are n linearly independent eigenvalues, where n is a size of L_s. Let us denote the eigenvalues of L_s $0 = \lambda_0 \leq \lambda_1 \leq \cdots \leq \lambda_{n-1}$. The following lemmas present some basic facts about the spectrum of L_s. An important metric in a graph is the volume, which is defined as:

$$\operatorname{vol} G = \sum_i d(i). \qquad (3.10)$$

The diameter D of a graph is the maximum number of edges between any two vertices of G. The following theorem describes an upper bound of λ_1 in terms of D and $\operatorname{vol} G$.

Lemma 3.2.1. Let G be a graph with diameter $D \geq 4$, and let k denote the maximum degree of G. Then

$$\lambda_1 \leq 1 - 2\frac{\sqrt{k-1}}{k}\left(1 - \frac{2}{D}\right) + \frac{2}{D}. \qquad (3.11)$$

Proof. A sketch of the proof of is given in [14]. □

3.2.4 Choice of similarity function

In the application we propose in this work, the weight function $w(i,j)$ is a function that measures the degree of similarity between data assigned to vertices i and j. Although, several options for w are discussed in [65], the choice depends on the problem, so no general theory can be formulated.

One popular choice for the similarity function is the Gaussian function

$$w(i,j) = e^{-\frac{d(i,j)^2}{\sigma^2}} \tag{3.12}$$

where $d(i,j)$ is some distance measure between the data assigned to two vertices i and j, and σ is a parameter to be chosen. Von Luxburg in [65] explains that σ can be chosen to be on the order of $log(n) + 1$, where n is the number of vertices. This similarity function is an appropriate choice when vertices are, for example, points in $\mathbb{R}^n$, since two points that are close together are more likely to belong to the same cluster than two points that are far apart.

Another choice for the similarity function used in this work is the Zelnik-Manor and Perona weight function for sparse matrices described in [69]:

$$w(i,j) = e^{-\frac{d(i,j)^2}{\sqrt{\tau(i)\tau(j)}}} \tag{3.13}$$

where the local parameter $\sqrt{\tau(i)} = d(i,k)$ and k is the M^{th} closest vertex to vertex i. As noted in [8], one should use this similarity function for classification when there exist multiple scales to be classified. In [69], M is chosen to be 7, while in [61], it is 10. Depending on the data set, we use either (3.12) or (3.13).

The choice of $d(i,j)$ varies with the data set. If one wants to cluster points in $\mathbb{R}^n$, a reasonable choice for $d(i,j)$ is the Euclidean distance between points i and j. In the case of image processing, where the vertices are the pixels in the image, to construct $d(i,j)$, we use the concept of feature vectors, as in [8]. Each vertex i is assigned a n-dimenstional feature vector, and $d(i,j)$ is then the weighted 2-norm (where each coordinate of the vector is assigned a weight) of the difference of the feature vectors of pixels i and j. More details on $d(i, j)$ in this case are given in sections 3.5.1 and 3.5.2.

3.3 Nonlocal operators

In general, image processing methods that are local fail to produce satisfactory results on images with repetitive structures and textures because they only operate on small neighborhoods, without using any information about the whole domain. The advantage of nonlocal operators is that they contain data about the whole vertex set and are thus more successful with those types of images.

Zhou and Schölkopf in their papers [17, 72, 71, 73] formulated a theory of nonlocal operators that is related to the discrete graph Laplacian described in section 3.2.2. Buades, Coll and Morel applied this nonlocal theory to denoising algorithms in their work [5]. Osher and Gilboa proposed using nonlocal operators to define functionals involving the TV seminorm for various image processing applications in their work [31].

We review nonlocal calculus below, where all definitions are continuous. Let $\Omega \in \mathbb{R}^n$, $u(x)$ be a function $u : \Omega \to \mathbb{R}$ and the nonlocal derivative be defined as

$$\frac{\partial u}{\partial y}(x) = \frac{u(y) - u(x)}{d(x, y)}, \quad x, y \in \Omega \tag{3.14}$$

where d is some positive distance defined on the space and $0 < d(x, y) \leq \infty \; \forall x, y$. If the (symmetric) weight function is defined as

$$w(x, y) = \frac{1}{d(x, y)^2}, \tag{3.15}$$

the nonlocal derivative can be written as

$$\frac{\partial u}{\partial y}(x) = (u(y) - u(x))\sqrt{w(x, y)}. \tag{3.16}$$

We now consider vectors and denote them as $\vec{v} = v(x, y) \in \Omega \times \Omega$. Let $\vec{v}_1$ and $\vec{v}_2$ be two such vectors. We define the dot product and the inner product as

$$(\vec{v}_1 \cdot \vec{v}_2)(x) = \int_\Omega v_1(x,y)v_2(x,y)dy, \tag{3.17}$$

$$\langle \vec{v}_1, \vec{v}_2 \rangle = \langle \vec{v}_1 \cdot \vec{v}_2, 1 \rangle = \int_{\Omega\times\Omega} v_1(x,y)v_2(x,y)dxdy. \tag{3.18}$$

The magnitude of a vector can be defined as

$$|v|(x) = \sqrt{\vec{v}\cdot\vec{v}} = \sqrt{\int_\Omega v(x,y)^2 dy}. \tag{3.19}$$

while the nonlocal gradient $\triangledown_w u(x) : \Omega \to \Omega \times \Omega$ is the vector of all partial derivatives:

$$(\nabla_w u)(x,y) = (u(y) - u(x))\sqrt{w(x,y)}, \quad x, y \in \Omega. \tag{3.20}$$

With the above definitions, the nonlocal divergence $div_w \vec{v}(x) : \Omega \times \Omega \to \Omega$ is defined as the adjoint of the nonlocal gradient:

$$(div_w \vec{v})(x) = \int_\Omega (v(x,y) - v(y,x))\sqrt{w(x,y)}dy. \tag{3.21}$$

The Laplacian is now defined as

$$\Delta_w u(x) = \frac{1}{2} div_w(\nabla_w u(x)) = \int_\Omega (u(y) - u(x))w(x,y)dy. \tag{3.22}$$

Since the graph Laplacian was defined in section 3.2.2 as

$$Lu(x) = \sum_y w(x,y)(u(x) - u(y)) \tag{3.23}$$

one can interpret $-Lu(x)$ as a discrete approximation of $\Delta_w u$. Note that a constant of $\frac{1}{2}$ was needed here to relate the two Laplacians.

According to the nonlocal calculus described above,

$$\int_{\Omega} |\nabla u|^2 dx = \int_{\Omega\times\Omega} (u(y) - u(x))^2 w(x,y) dxdy. \tag{3.24}$$

Since

$$u \cdot Lu = \frac{1}{2} \sum_{x,y} w(x,y)(u(x) - u(y))^2 \tag{3.25}$$

one can consider $2u \cdot Lu$ to be the discrete graph version of $\int |\nabla u|^2 dx$.

In their paper [8], Bertozzi and Flenner replace the $\frac{\epsilon}{2} \int |\nabla u|^2 dx$ term of (1.5) by $\epsilon u \cdot Lu(x)$. However, normalization of the Laplacian is necessary (refer to Section 3.2.2), so instead they use

$$\epsilon u \cdot L_s u = \frac{\epsilon}{2} \sum_{x,y} \frac{w(x,y)(u(x) - u(y))^2}{\sqrt{d(x)d(y)}}. \tag{3.26}$$

When the variational solution u takes the values -1 or 1,

$$u \cdot L_s u = C + 4 \sum_{x\in A, y\in \bar{A}} \frac{w(x,y)}{\sqrt{d(x)d(y)}} - 2 \left(\sum_{x\in A, y\in A} \frac{w(x,y)}{\sqrt{d(x)d(y)}} + \sum_{x\in \bar{A}, y\in \bar{A}} \frac{w(x,y)}{\sqrt{d(x)d(y)}} \right). \tag{3.27}$$

In this case, C is a constant that varies with the graph but not with the partition. The representation shows that the above is minimized when the normalized weights between vertices of different groups are small, but the normalized weights between vertices within a group are large. This is precisely the goal of graph clustering. Therefore, by replacing the $\frac{\epsilon}{2} \int | \bigtriangledown u|^2 dx$ term of (1.5) with $\epsilon u \cdot L_s u$, thus creating a graph based version of (1.5), and then minimizing the resulting equation, one achieves the desired segmentation.
The Γ-convergence of the graph based Ginzburg-Landau functional is investigated in [**?**]. The authors prove that as $\epsilon \to 0$, the limit is related to the total variation semi-norm and cut from (1.7).
Another important operator that arises from the need to define variational methods on graphs is the mean curvature on graphs. This non-local operator was introduced by Osher and Shen in [48] , who defined it via graph based p-Laplacian operators. p-Laplace operators

are a family of quasilinear elliptic partial differential operators defined for $1 \leq p < \infty$:

$$L^p(f) = \nabla \cdot (\mid \nabla f \mid^{p-2} \nabla f). \tag{3.28}$$

In the special case $p = 2$ p-Laplacian is just a regular Laplacian, for $p = 1$ p-Laplacian represents curvature.

The discrete graph version of p-Laplace operators is defined as:

$$L^p(u(x)) = \frac{1}{p} \sum_{(x,y)\in E} w(x,y)(\|\nabla u(x)\|^{p-2} + \|\nabla u(y)\|^{p-2})(u(x) - u(y)). \tag{3.29}$$

Note that the graph 2-Laplacian is just the graph Laplacian, which is consistent with the continuous case. Let us now define the mean curvature on graphs, the discrete analog of the mean curvature of the level curve of a function defined on a continuous domain of $\mathbb{R}^N$:

$$\kappa_w = \frac{1}{2} \sum_{(x,y)\in E} w(x,y)(\frac{1}{\|\nabla u(x)\|} + \frac{1}{\|\nabla u(y)\|})(u(x) - u(y)). \tag{3.30}$$

Note that in the case of unweighted mesh graph κ_w becomes a numerical discretization of mean curvature.

3.4 Data labeling algorithm

We construct a new data labeling algorithm by proposing a different approach to minimize (3.1) than the one in [8] to obtain a more simple and efficient method that eliminates the diffuse interface parameter ϵ. Our scheme is based on a variation of the MBO scheme.

As was shown in section 1, for small ϵ, the MBO thresholding scheme can be used to evolve the Allen-Cahn equation to steady state. The scheme consists of two steps: a heat equation propagation step and a thresholding step.

A candidate for the threshold dynamics of (3.1) is found by splitting equation (3.3), which is the Allen-Cahn equation plus an extra fidelity term. There are several options, including splitting the equation into three steps, but we choose the possibility in which equation (3.3) is split so that the thresholding step resembles the one in the original MBO scheme, as is done in [23].

Therefore, our algorithm consists of alternating between the following two steps to obtain approximate solutions $u_n(x)$ at discrete times:

- **S**tep 1. (heat equation with forcing term) Propagate using

$$\frac{\partial y}{\partial t} = -L_s y - C_1 \lambda(x)(y - u_0) \tag{3.31}$$

starting with u_n. Note that C_1 can be different from the original C.

- **S**tep 2. (thresholding) Set

$$u^{n+1}(x) = \begin{cases} 1, & \text{if } y(x) \geq 0 \\ -1, & \text{if } y(x) < 0 \end{cases}$$

Note that we now use 0 as the thresholding value (instead of $\frac{1}{2}$ as in the original MBO scheme) since the values of u are concentrated at -1 and 1, not 0 and 1.

We have decided to discretize (3.31) above in the following manner:

$$\frac{u^{n+1} - u^n}{dt} = -L_s u^{n+1} - C_1 \lambda(x)(u^n - u_0). \tag{3.32}$$

Note that the symmetric Laplacian is calculated implicitly. This is due to the stiffness of the operator, which is caused by a wide range of its eigenvalues. An implicit term is needed, since an explicit scheme requires all the scales of the eigenvalues to be resolved numerically.

The scheme is solved using the spectral decomposition of the symmetric graph Laplacian. Let $u^n = \sum_k a_k^n \phi_k(x)$ and $C_1 \lambda(u^n - u_0) = \sum_k d_k^n \phi_k(x)$, where $\phi(x)$ are the eigenfunctions of the symmetric Laplacian. Using the obtained representations and equation (3.32), we obtain

$$a_k^{n+1} = \frac{a_k^n - dt d_k^n}{1 + dt\lambda_k} \tag{3.33}$$

where λ_k are the eigenvalues of the symmetric graph Laplacian.

Therefore, the new algorithm consists of the following:

- **S**tep 1. Create a graph from the data, choose a similarity function and then calculate the symmetric graph Laplacian.
- **S**tep 2. Calculate the eigenvectors and eigenvalues of the symmetric graph Laplacian. It is only necessary to calculate a portion of the eigenvectors.
- **S**tep 3. Initialize u.
- **S**tep 4. Apply the two-step scheme (to minimize the Ginzburg-Landau functional) described above for a certain number of iterations until a stopping criterion is satisfied. Use the following method:

```
1. Let $a_k^0 = \sum_x u_0(x)\phi_k(x)$ and $d_k^0(x) = 0$ for all $x$.
2. Until a stopping criterion is satisfied, do the following:
    a. Repeat for some number $s$ of steps:
        1. $a_k^n \leftarrow \frac{a_k^n - \delta t d_k^n}{1+\delta t\lambda_k}$
        2. $y(x) = \sum_k a_k^n \phi_k(x)$
        3. $d_k^n = \sum_x C_1(y - u_0)(x)\phi_k(x)$
    b. (thresholding part)
```

$$u^{n+1}(x) = \begin{cases} 1, & \text{if } y > 0 \\ -1, & otherwise \end{cases}$$

c. Let $a_k^{n+1} = \sum_x u_{n+1}(x)\phi_k(x)$ and $d_k^{n+1} = \sum_x C_1(y - u_0)(x)\phi_k(x)$

The parameter δt is chosen using trial and error. The stopping criteria we use in our work is $\frac{||u_{new}-u_{old}||_2^2}{||u_{new}||_2^2} < \alpha = 0.0000001$.

3.4.1 Computation of eigenvectors

Our method involves the computation of eigenvalues and associated eigenvectors of the symmetric graph Laplacian. In practice, one needs to compute only a fraction of the eigenvalues and eigenvectors (since eigenvectors with very small eigenvalues are not very significant computationally), and different methods of doing so are used depending on the size of the domain.

When the graph is sparse and is of moderate size, around 5000×5000 or less, we use a Rayleigh-Chebyshev procedure outlined in [6]. It is a modification of an inverse subspace iteration method that uses adaptively determined Chebyshev polynomials. The procedure is also a robust method that converges rapidly and that can handle cases when there are eigenvalues of multiplicity greater than one.

When the graph is very large, such as in the case of image data labeling, the Nyström extension method, to be described in the next section, is used.

3.4.1.1 Nyström extension for fully connected graphs

Nyström extension [8, 10, 11, 52] is a matrix completion method often used in many image processing applications, such as kernel principle component analysis [19] and spectral clustering [43]. This procedure performs much faster than many alternate techniques because it uses approximations based on calculations on small submatrices of the original large matrix. When the size of the matrix becomes very large, this method is especially valuable.

Note that if λ is an eigenvalue of $\hat{W} = D^{-\frac{1}{2}}WD^{-\frac{1}{2}}$, then $1 - \lambda$ is an eigenvalue of L_s, and the two matrices have the same eigenvectors. We formulate a method to calculate the eigenvectors and eigenvalues of $\hat{W}$ and thus of L_s.

Let w be the similarity function, λ be an eigenvalue of W, and ϕ its associated eigenvector.

The Nyström method approximates the eigenvalue equation

$$\int_{\Omega} w(y,x)\phi(x)dx = \lambda\phi(x) \tag{3.34}$$

using a quadrature rule, a technique to find weights $c_j(y)$ and a set of L interpolation points $X = \{x_j\}$ such that

$$\sum_{j=1}^{L} c_j(y)\phi(x_j) = \int_{\Omega} w(y,x)\phi(x)dx + E(y) \tag{3.35}$$

where $E(y)$ represents the error in the approximation.

We use $c_j(y) = w(y, x_j)$ and choose the L interpolation points randomly from the vertex set V. Denote the set of L randomly chosen points by $X = \{x_i\}_{i=1}^{L}$ and its complement by Y. Partioning Z into $Z = X \cup Y$ and letting $\phi_k(x)$ be the the k^{th} eigenvector of W and λ_k its associated eigenvalue, we obtain the system of equations

$$\sum_{x_j \in X} w(y_i, x_j)\phi_k(x_j) = \lambda_k \phi_k(y_i) \qquad \forall y_i \in Y, \ \ \forall k \in 1, ..., L. \tag{3.36}$$

This system of equations cannot be solved directly since the eigenvectors are not known. To overcome this problem, the L eigenvectors of W are approximated using calculations involving submatrices of W. Let W_{XY} be defined as

$$\begin{bmatrix} w(x_1, y_1) & \dots & w(x_1, y_{N-L}) \\ \vdots & \ddots & \vdots \\ w(x_L, y_1) & \dots & w(x_L, y_{N-L}) \end{bmatrix}$$

where W has dimension $N \times N$. The matrices W_{YX}, W_{XX} and W_{YY} can be defined similarly. Notice that W_{XY}= ${W_{YX}}^T$. Then the matrix W can be written as

$$\begin{bmatrix} W_{XX} & W_{XY} \\ W_{YX} & W_{YY} \end{bmatrix}$$

To calculate the eigenvalues and eigenvalues of $\hat{W}$, one must correctly normalize the above weight matrix. The correct normalization is achieved by the following calculations, where we denote by $\mathbf{1}_K$ the K-dimensional unit vector. Let the matrices d_X and d_Y be defined as

$$\begin{aligned} d_X &= W_{XX}\mathbf{1}_L + W_{XY}\mathbf{1}_{N-L}, \\ d_Y &= W_{YX}\mathbf{1}_L + (W_{YX}W_{XX}^{-1}W_{XY})\mathbf{1}_{N-L}. \end{aligned} \tag{3.37}$$

If $A./B$ denotes componentwise division between matrices A and B, and v^T denotes the transpose of vector v, then define the matrices $\hat{W}_{XX}$ and $\hat{W}_{XY}$ as

$$\begin{aligned} \hat{W}_{XX} &= W_{XX}./(s_X s_X^T) \\ \hat{W}_{XY} &= W_{XY}./(s_X s_X^Y) \end{aligned} \tag{3.38}$$

where $s_X = \sqrt{d_X}$ and $s_Y = \sqrt{d_Y}$. It is shown in [8] that if $\hat{W}_{XX} = B_X D B_X^T$, and if A and Γ are matrices such that

$$A^T \Gamma A = \hat{W}_{XX} + \hat{W}_{XX}^{-\frac{1}{2}} \hat{W}_{XY} \hat{W}_{YX} \hat{W}_{XX}^{-\frac{1}{2}} \tag{3.39}$$

then the eigenvector matrix V consisting of L eigenvectors of $\hat{W}$ and thus of L_s is given by

$$\begin{bmatrix} B_X D^{\frac{1}{2}} B_X^T A \Gamma^{-\frac{1}{2}} \\ \hat{W}_{YX} B_X D^{-\frac{1}{2}} B_X^T A \Gamma^{-\frac{1}{2}} \end{bmatrix}$$

while $I - \Gamma$ contains the corresponding eigenvalues of L_s in its diagonal entries. Therefore, the efficiency of the Nyström extension method lies with the fact that when computing the eigenvalues and eigenvectors of an $N \times N$ matrix, where N is large, it approximates them using calculations involving only much smaller matrices, the largest of which has dimension $N \times L$, where L is small. Although this method is very efficient, there are problems when it is applied to binary image inpainting, especially when the image has a repetitive structure. This occurs because of singular or nearly singular matrices that arise in the calculations of the Nyström extension method. Therefore, in this case, we use the Rayleigh-Chebyshev procedure of [6] to calculate the eigenvalues and associated eigenvectors.

3.4.2 Results for data labeling

We applied our labeling algorithm on three data sets: the two moons data set, an image and the House of Representatives voting records from 1984. A comparison of the results to those of the method of Bertozzi and Flenner in [8] is displayed in tables 3.1 and 3.2. The tables show that our method significantly reduces the number of iterations and the minimization time. There are four parameters that were involved in this problem: the number of eigenvectors, C_1, σ and dt. As long as the number of eigenvectors was not too small compared to the sample size, there was enough information to produce an accurate result. The fidelity term C_1 was also chosen to be relatively big so that the fidelity region is preserved. The weight matrix parameter σ was chosen so that the weights contain a wide range of numbers from 0 to 1; in other words, the situation in which most weights are very close to 0 (or 1) was avoided. The time range dt was the most difficult parameter, and its value differed per data set. It was mostly chosen by trial and error, but in all cases it was neither too small or too big (in which case there is either no evolution in the iterations or frequent oscillations). The algorithm is relatively robust with the above conditions. In the case of semi-supervised labeling, $\lambda(x)$ was set to 1 on the known region and 0 elsewhere, since our fidelity term assumed a least-squares fit on the information supplied. Note that the fidelity term allows for a small amount of misclassification in the known data.

3.4.2.1 Two moons

This data set was used by Bühler *et al.* in [9] in relation to spectral clusering using the p-Laplacian. It is constructed from the following two half circles in $\mathbb{R}^2$ with radius one. The first half circle is centered at the origin and is in the upper half plane. The second half circle is formed by taking the lower half of the circle centered at $(1, 0.5)$. A thousand points are chosen uniformly from each of the two half circles. The two thousand points are then embedded in $\mathbb{R}^{100}$, and i.d.d. Gaussian noise with standard deviation 0.02 is added to each

coordinate. The goal is to segment those two half circles using unsupervised binarization. This is achieved using the mean zero constraint. An affinity matrix W is created using the weight function $w(i,j) = e^{-\frac{d(i,j)^2}{\sqrt{(\tau(i)\tau(j))}}}$, a weight function introduced by Zelnik-Manor and Perona in [69], where $\tau(i)$ is the Euclidean distance between point i and the Mth closest point to it, and $d(i,j)$ is the Euclidean distance between points i and j. The matrix $W(i,j)$ is made sparse by setting $W(i,j)$ equal to zero if point j is not among the Mth closest points to point i. It is then "symmetrized" by setting $W(i,j) = max(W(i,j), W(j,i))$. To calculate the eigenvectors, the Rayleigh-Chebyshev procedure [6] is used, since the graph is not large and Nyström extension is inefficient for sparse graphs [8]. Since the problem is unsupervised binarization, in step IV of the algorithm, there is no fidelity term so $\lambda(x) = 0$ for all x. Thus, $d_k^n = 0$ for all k and n. Due to the mean zero constraint, $\int u(x)dx = 0$, before thresholding, one applies the mean constraint to y by subtracting its mean from each element of y. For initialization of u, we use the sign of the second eigenvector of the symmetric Laplacian after the mean zero constraint has been applied to it. We compared our results to the method of Bertozzi and Flenner in [8] by running simulations on 35 different randomly generated two moons data sets, each taking around 2 seconds to run. The average accuracy was 96.0520% and 96.0460% for our method and the method in [8], respectively. However, 40 iterations in the minimization procedure were used, compared to 300 needed using the method in [8]. Therefore, our method resulted in a significant decrease in the number of iterations. The minimization time was also decreased from 0.105 seconds to 0.002 seconds. These results are displayed in Tables 3.1 and 3.2.
We also compared our results to a spectral clustering method of thresholding the second eigenvector of L_s. The results are displayed in Figure 3.1. Clearly, clustering using the second eigenvector does not result in an accurate binarization.

3.4.2.2 Image data labeling

We also applied our algorithm to label objects in images of cows from the Microsoft image database. The goal was semi-supervised image labeling, where two images are inputted into the algorithm, one of which has been hand labeled into the two classes. The algorithm labels

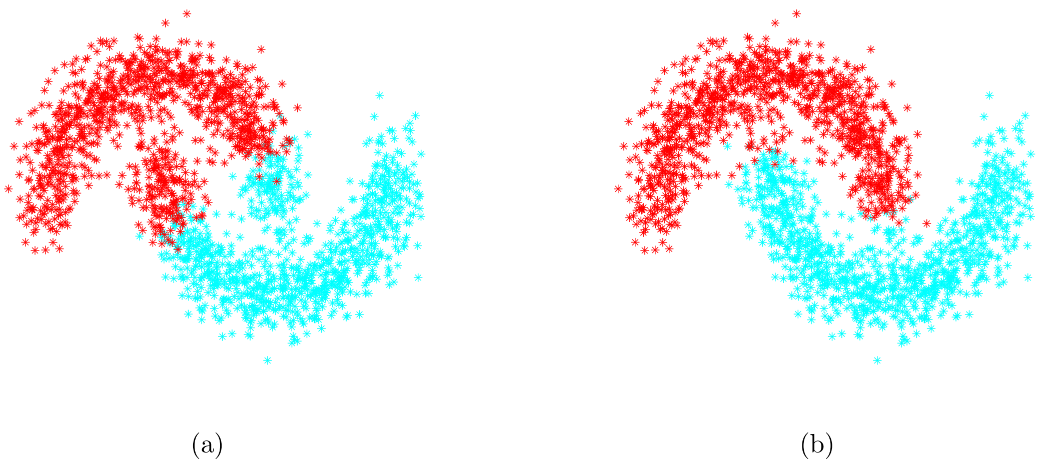


Figure 3.1: Binarization by thresholding the second eigenvector in image(a) and our method in image (b), respectively. The spectral method achieves only 83.75% accuracy while or method is 97.7% accurate. The four parameters s (in step IV of our algorithm), number of eigenvectors, dt, and M (parameter in the Zelnik-Manor and Perona weight function) are set to 3, 25, 0.725 and 13, respectively.

the second image based on the labeling of the first. A fully connected graph is constructed in this case, and the entries in the affinity matrix are calculated using feature vectors. Every pixel in the image is assigned a feature vector consisting of intensity values of pixels in its neighborhood, which was of size 7×7 in our labeling tests. We use the formula $w(i,j) = e^{-\frac{d(i,j)^2}{\sigma^2}}$, where $d(i,j)$ is the weighted 2-norm of the difference of the feature vectors of pixels i and j, and we add along the three RGB channels of the image. The weighted 2-norm modifies the components of the entered vector by giving more weight to the pixels close to the original pixel and less weight to those farther away. We use a linearly decreasing kernel, where the weight decreases linearly. This construction can be used to label different types of objects using, for example, their color and texture features. Note that the weight function can be modified according to the image. For example, a weight function calculated using the spectral angle may be more effective in the labeling of hyperspectral images. To obtain eigenvalues and eigenvectors of L_s, the Nyström extension method is used, since the size of the graph is very large ($70,000 \times 70,000$). For the problem, in the fidelity term, $\lambda(x)$ was set to 1 on the hand labeled image and 0 on the unlabeled image. On the hand labeled

	Minimization time in method in [8]	Minimization time in our method
two moons	0.105 s	0.002 s
grass label	8 s	3.5 s
cow label	18 s	3.5 s
sky label	6 s	1.8 s
voting data set	0.035 s	0.002 s

Table 3.1: Comparison of minimization time of the two methods

image, we initialized u to be 1 for one class and -1 for the other class. On the unlabeled image, u_0 was set to zero. The results are displayed in Figure 3.2, where it is shown that our algorithm is robust to mislabeling in the hand labeled image. To transfer the label for the grass, cows and sky, our method needed about 29, 29, 27 seconds, respectively. The number of iterations in the minimization procedure (step 4 of the algorithm) and minimization time as compared to the method in [8] are displayed in Tables 3.1 and 3.2. The calculations show that our method significantly reduces the minimization time and the number of iterations.

3.4.2.3 House voting records from 1984

We applied our algorithm to the US House of Representatives voting records data set, which consists of 16 different votes from each of the 435 individuals. The goal was to assign each individual to either the Republican or the Democrat party using the prior knowledge of the party affiliation of only five individuals, two Democrats and three Republicans. The votes were taken in 1984 from the 98th United States Congress, 2nd session. An affinity matrix is constructed using calculations involving feature vectors. A 16-dimensional feature vector is assigned to each individual consisting of his/her 16 votes. A "yes" vote is set to 1, a "no" vote is set to -1, while a "did not vote" recording is set to 0. The weight function used is $w(i,j) = e^{-\frac{d(i,j)^2}{\sigma^2}}$ where $d(i,j)$ is the 2-norm of the difference between the feature vectors of points i and j. The graph is made sparse by setting $W(i,j)$ equal to zero if point j is not among the M^{th} closest points to point i. The graph is then "symmetrized" by setting $W(i,j) = max(W(i,j), W(j,i))$. To calculate the eigenvectors, a SVD solver is used. In step IV of the algorithm, the function u is initialized to 1 for the two Democrats, -1 for the three Republicans and 0 for the rest of the Representatives. The three Republicans were chosen to be the first, second and eighth person in the list. The Democrats were chosen to be the third and fourth person in the list. In the fidelity term, $\lambda(x)$ was set to 1 for each of five known individuals and 0 for the rest. The parameters C_1 (fidelity term parameter), s (in step IV of our algorithm), number of eigenvectors, dt, σ and M are set to 9.25, 3, 45, 4.675, $\sqrt{5}$ and 10, respectively. We obtained an accuracy of 94.023%. Only 5 iterations in the minimization procedure were needed compared to 450 iterations needed by the method in [8]. Each simulation took about 0.7 seconds, and the minimization time was decreased by more than 15 times. The information is shown in Tables 3.1 and 3.2. Some of the votes predicted the party affiliation very well, i.e. above 85%. We investigated the accuracy of our algorithm when these votes were removed. With top two, top six and top eight most predictive votes removed, our method obtained an accuracy of 90.1149%, 88.34448% and 81.1494%, respectively. The order of the top eight predictive votes from the most predictive to least predictive is vote 4, 14, 1, 2, 15, 6, 3 and 8.

	# of iterations in method in [8]	# of iterations in our method
two moons	300	40
grass label	130	22
cow label	274	29
sky label	84	11
voting data set	400	5

Table 3.2: Comparison of # of iterations of the two methods

3.5 Image inpainting algorithm

The problem of fitting information in the missing pixels of an image is an important inverse problem in image processing with various applications. Obviously, the goal is to produce a modified image that will look natural to an observer. The problem of inpainting may also be seen as the problem of removing occlusive objects from an image. Sparse reconstruction refers to the problem of recovering randomly distributed missing pixels. There are numerous approaches to solve these problems in the current literature. Local TV methods became state-of-the-art techniques for image impainting. However, since they do not perform well on images with high texture, methods that decompose images into cartoon and texture and simultaneously inpaint both are developed [41, 56]. The problem is also solved with nonlocal inpainting methods. We are particularly interested in the nonlocal inpainting algorithm from [30] as we develop a computationally efficient nonlocal method. Some very successful nonlocal methods for inpainting and sparse reconstruction are given in [49] and [25]. Recently, the class of methods that use dictionaries of small patches that commonly appear in natural images became increasingly popular. Those methods, besides inpainting, are also successful in denoising as shown in [36]. In addition, a method for image inpainting using Navier-Stokes fluid dynamics is proposed in [2]. The authors use Navier-Stokes dynamics to propagate isophotes into the inpainting region, thus simulating the way painting restoration is done. Wavelets and framelets are also successfully applied to solve inpainting problems [18, 35]. We modify our labeling algorithm slightly for the purpose of binary and grayscale image inpainting. The algorithm consists of the same 4 steps:

- Create a graph from the data using pixels as vertices, choose a similarity function and then create the symmetric graph Laplacian.
- Calculate the eigenvectors and eigenvalues of the symmetric graph Laplacian. It is only necessary to calculate a fraction of the eigenvectors.
- Initialize u.
- Apply the two-step scheme (to minimize the Ginzburg-Landau functional) detailed in Section 2 for a certain number of iterations until a stopping criterion is satisfied.

However, there are some important differences to be discussed in sections 3.5.1 and 3.5.2. Our algorithm is an efficient image inpainting algorithm that is able to correct images with repetitive structure or those with high texture content.

3.5.1 Binary image inpainting

Although the key steps of the labeling algorithm remain the same when it is modified for image inpainting, there are key differences to be noted. For example, it is clear that if a damaged image is used to construct the adjacency matrix W, the results might not be accurate, so we first apply a fast and simple H^1 inpainting algorithm on the image and then use the result to create W. Although the latter algorithm is very fast, it does not perform well on images with high textures and repetitive structures nor does it preserve edges [**?**], something that is achieved by our algorithm. The matrix W is built by using a window of a certain size around each pixel. We set $W(i, j) = 0$ for all pixels j that are not in the window of pixel i. Inside the window, $W(i, j) = w(i, j)$, where the weight function is calculated in the same way as in section 3.4.2.2,i.e,. using feature vectors and the Gaussian weight function. No updating of the matrix W is necessary in the case of binary image inpainting. The Rayleigh-Chebyshev procedure is used to calculate the eigenvectors and eigenvalues of the graph Laplacian for binary inpainting. As mentioned before, the Nyström extension method encounters some problems when dealing with binary images. In step IV of the algorithm, $\lambda(x)$ in the fidelity term is set to 0 on the inpainting region (which is given the value 0.5 on a 0 to 1 intensity scale) and to 1 on the rest of the image, while u_0 is set to 0 on the inpainting region, 1 on the white area and -1 on the black area. The same stopping criterion is used.

3.5.2 Grayscale image inpainting

To generalize to graycale inpaining, we split the signal bit-wise into channels, as in [18]:

$$u(x) = \sum_{m=0}^{K-1} u_m(x) 2^{-m} \tag{3.40}$$

where u_m denotes the m^{th} component or digit in the binary representation of the signal, and $u_m \in \{0, 1\}$ for $\forall x$. A fully connected graph is created in the same way as in section 3.4.2.2. Again, we first apply the H^1 inpainting algorithm on the image, and use the result to build the matrix W. The Nyström extension method is used to calculate the eigenvalues and corresponding eigenvectors since the size of the graph is very large. In step IV of the algorithm, $\lambda(x)$ in the fidelity term is set to 0 on the inpainting region (which is either black or white) and to 1 on the rest of the image. The initialization of u varies with the bit. In the inpainting region, u_0 is 0, while in the rest of the image, it is 1 on the area where the bit is 1 and -1 on area where the bit is 0. The same stopping criterion is used, except $\alpha = 0.0001$. For some images, step IV is performed for a certain number of iterations. Updating the matrix W is often necessary for grayscale inpainting, since the adjacency matrix formed from the damaged image is usually not good enough to restore texture and complex patterns, as it contains "bad" regions whose values lie far from the true value. In our tests, every few iterations, the matrix is updated using the result from the last iteration as the "new image".

3.5.3 Binary image inpainting results

We applied our algorithm on an image of Barbara and one of stripes. The results and their PSNR are displayed in Figure 3.3. In both cases, the algorithm was able to recover the texture and repetitive structure present in the image, something that is unfeasible for simple algorithms, such as local TV inpainting.

3.5.4 Grayscale image inpainting results

We applied our algorithm on an image of Barbara and a chessboard-like pattern. The goals ranged from removing occlusive objects, such as a flower, text or a rectangle, to sparse reconstruction. The results along with their PSNR are displayed in Figures 3.4-3.9. Figure 3.9 is a reconstruction of the original image **??**. In all cases, repetitive structure and texture were recovered.

We compare our results to local and nonlocal TV inpainting. Local TV inpainting fails to recover texture and repetitive structure. While the results of nonlocal TV inpainting are comparable to those of our method, our method is more efficient. Timing results are displayed in Table 3.3. We also show our method and nonlocal TV inpainting at certain iterations in Figure 3.10. To implement the nonlocal TV inpainting algorithm, we used the Bregmanized version detailed in [67] and modified it for inpainting. The stopping condition was the same as in our inpainting algorithm, and a quick H^1 inpainting algorithm was run on the image before the weights were calculated.

	Total time for nonlocal TV	Total time for our method
chessboard-like pattern	266 s	48 s
text inpainting	410 s	67 s
small rectangle inpainting	1882 s	443 s
large rectangle inpainting	3397 s	832 s
50% inpainting	1402 s	333 s

Table 3.3: Timing comparison between our method and the nonlocal TV

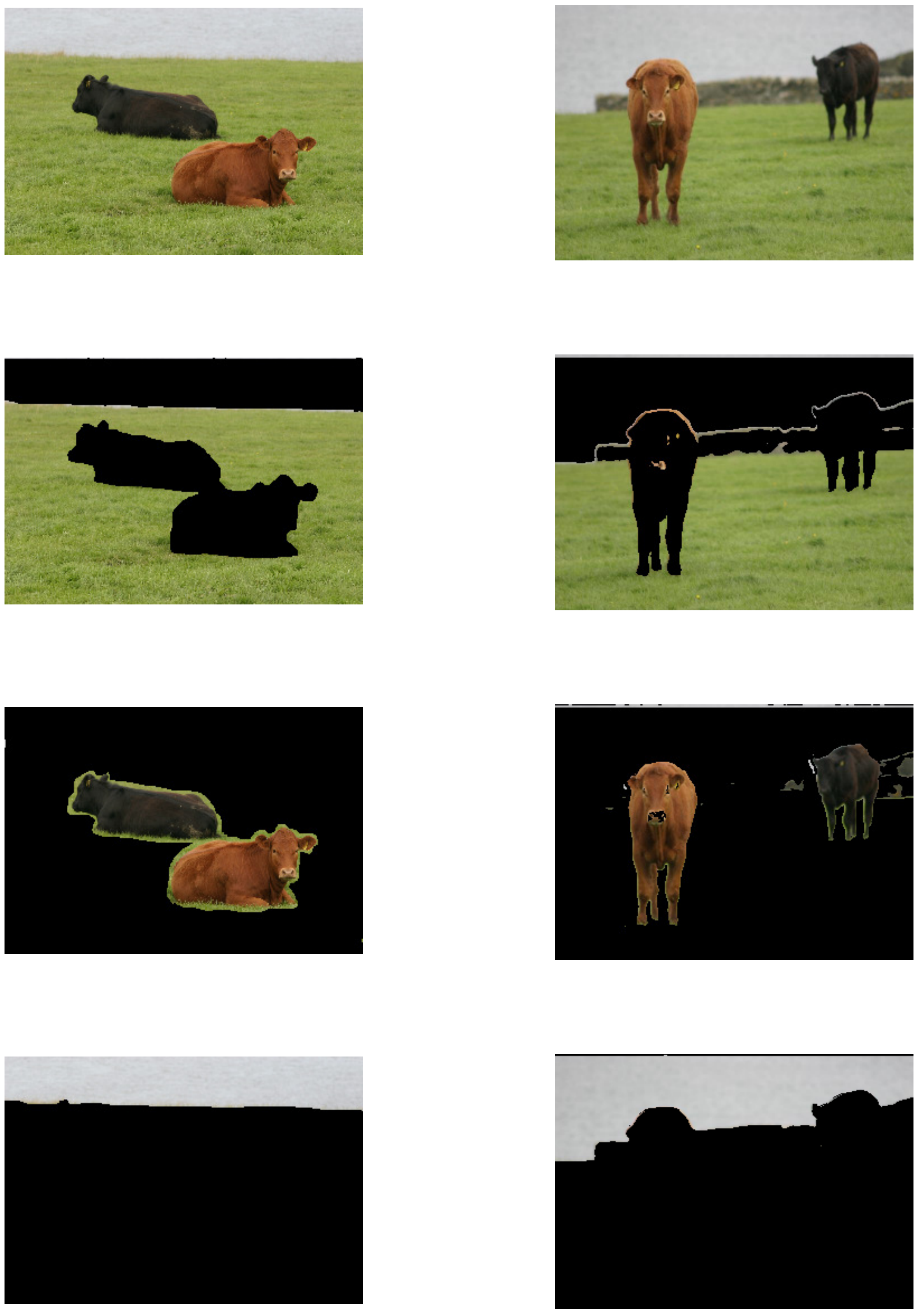


Figure 3.2: In the top row we see the original images. The left column contains the images of the original labels, and the right one contains the images of the grass, cow and sky label transferred using our algorithm. The number of eigenvectors, C_1 and σ were set to 200, 30 and 22, respectively. The parameter dt was 0.03, 0.003 and 0.17 for the grass, cow and sky label, respectively.

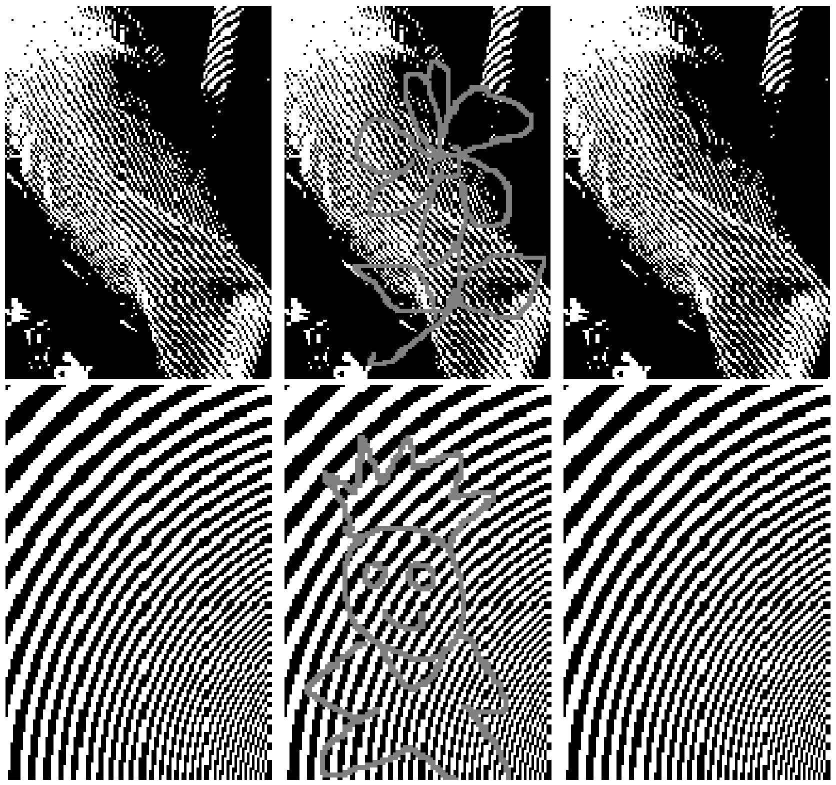
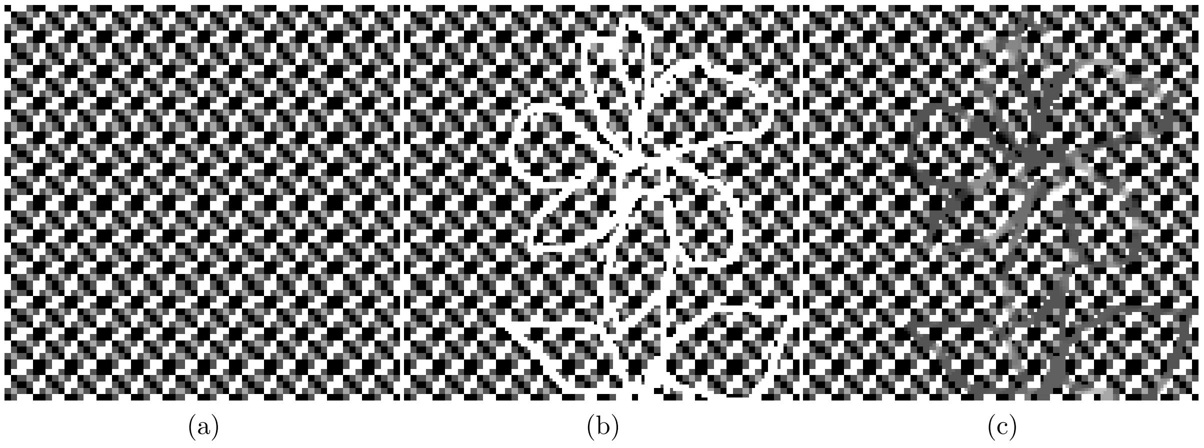


Figure 3.3: Binary Inpainting. In the top row we present the Barbara example and our result acheves the PSNR of 20.6896. The bottom row contains the fingerprint example and our solution has the PSNR of 25.0687. For the Barbara image, the simulation took 113 seconds, and there were 6 iterations in the two-step scheme. We used $C_1 = 700$, $dt = 0.003$, $\sigma = 45$, 31×31 neighborhood for feature vector calculation, 21×21 window and calculated 400 eigenvectors. For the image of stripes, the simulation took 66 seconds, and there were 4 iterations in the two-step scheme. We used $C_1 = 700$, $dt = 0.002$, $\sigma = 45$, 17×17 neighborhood for feature vector calculation, 21×21 window and calculated 200 eigenvectors.



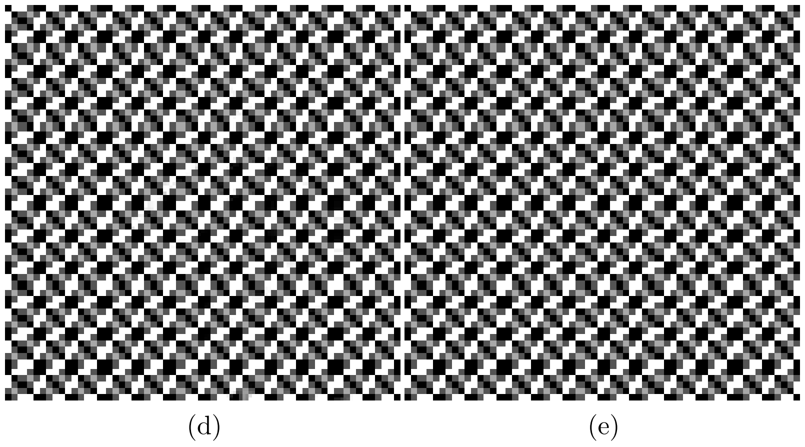
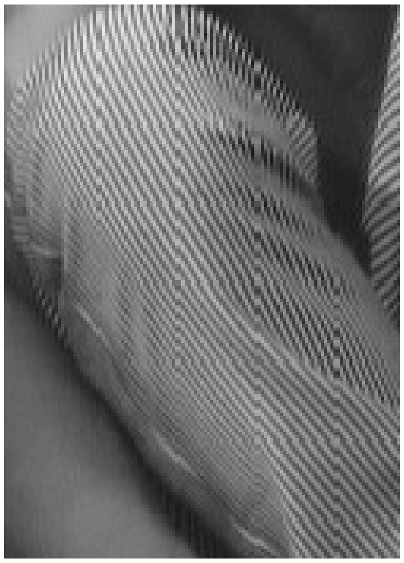
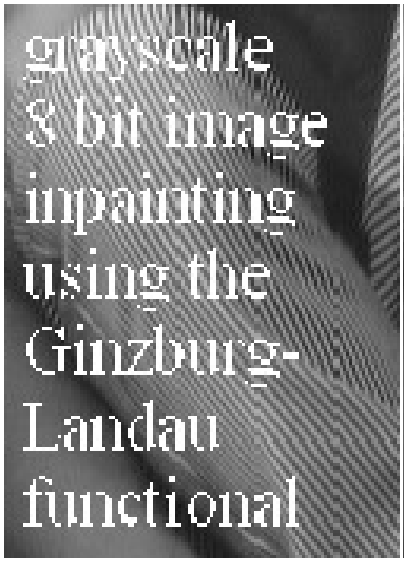
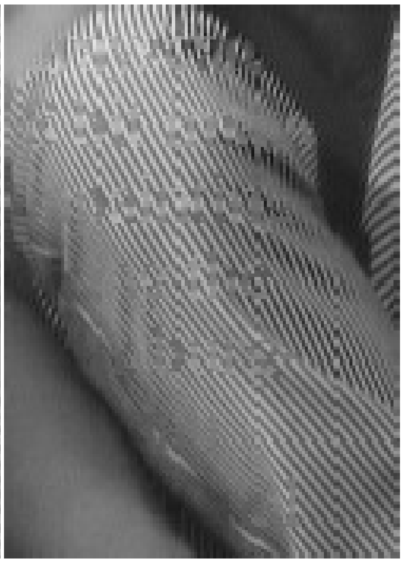


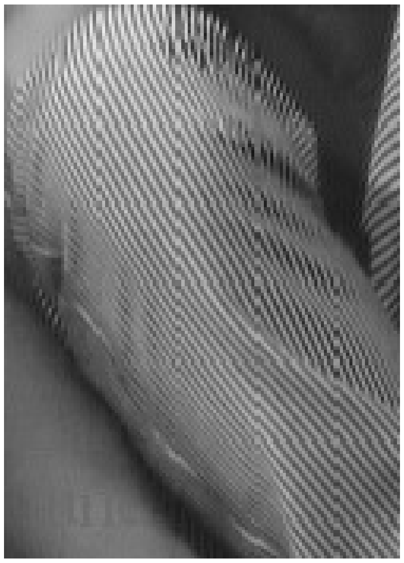
Figure 3.4: Pattern. Image (a) is the original image while image (b) is the damaged one. Images (c), (d) and (e) are the results obtained by the local TV method, the non-local TV method and our method, respectively. The local TV achives the PSNR of 16.5520, the non-local result has the PSNR 41.3891, while our methods obtains the perfect reconstruction. The simulation took 48 seconds, and there were 2 iterations in the two-step scheme. We used $C_1 = 700$, $dt = 0.005$, $\sigma = 20$, 41×41 neighborhood for feature vector calculation, and calculated 600 eigenvectors. No updating of W was necessary. The nonlocal inpainting took 266 seconds.


(a)



(b)


(c)


(d)

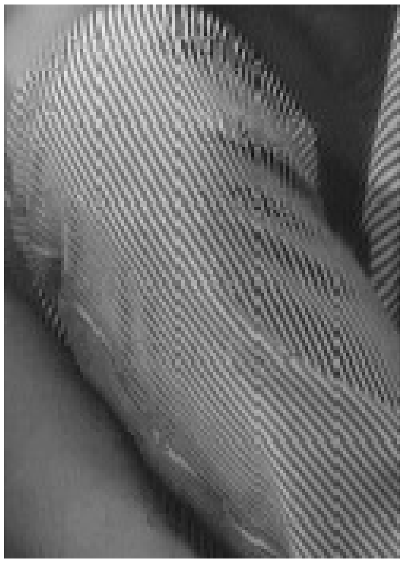
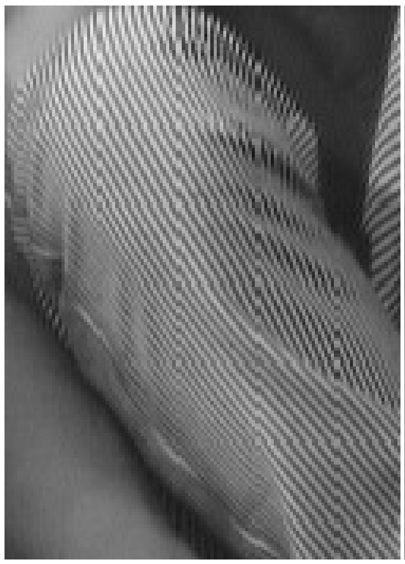
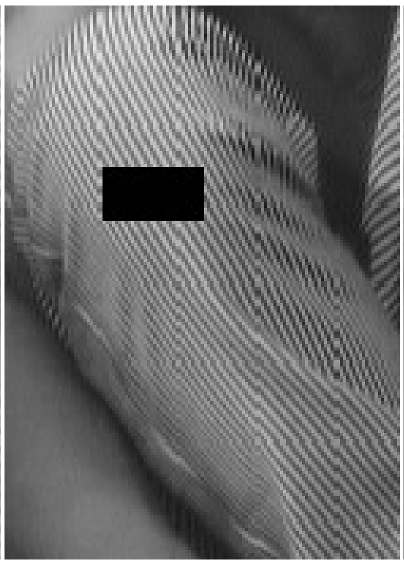
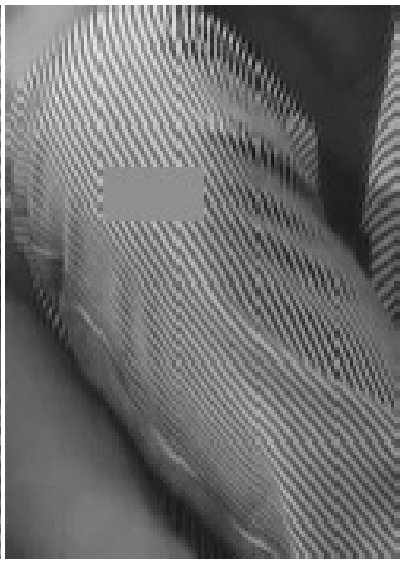
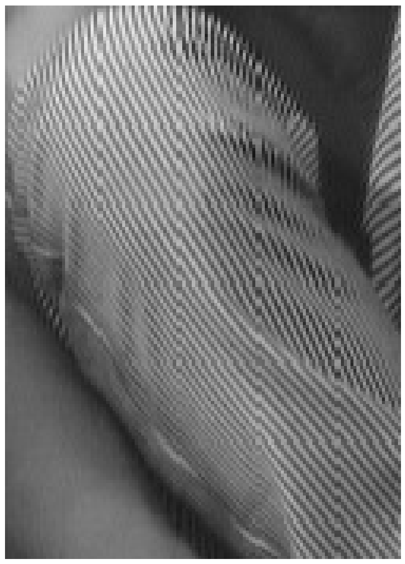

(e)

Figure 3.5: Text Inpainting. Image (a) is the original image, image (b) is the damaged image. Images (c), (d) and (e) are the results obtained by the local TV, non-local TV and our method respectively. The PSNR values for the local, non-local and our result are 29.1508, 35.6896 and 34.0688 respectively. The simulation took 67 seconds, and there were 4 iterations in the two-step scheme. We used $C_1 = 700$, $dt = 0.005$, $\sigma = 5$, 21×21 neighborhood for feature vector calculation, and calculated 500 eigenvectors. We update W every other iteration. The nonlocal inpainting took 410 seconds.


(a)


(b)


(c)


(d)

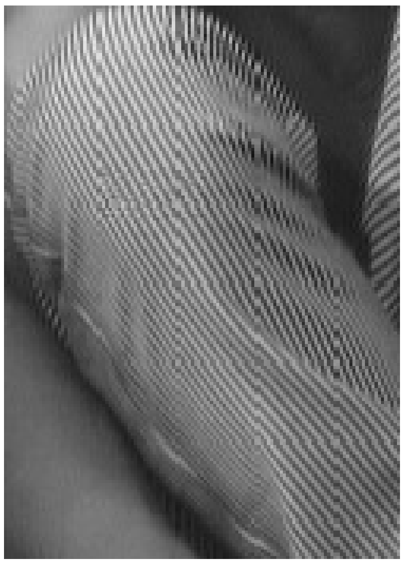
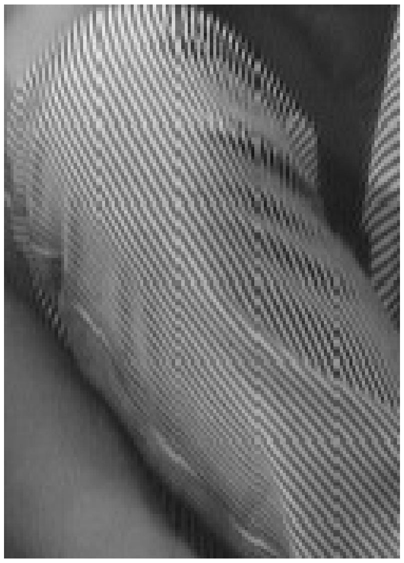
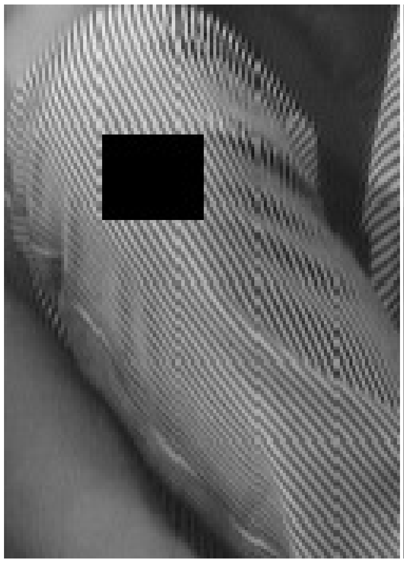
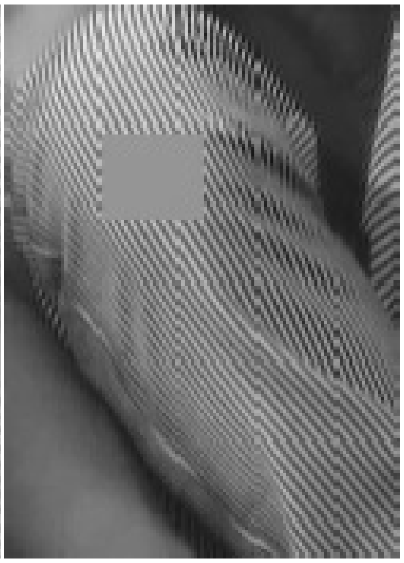
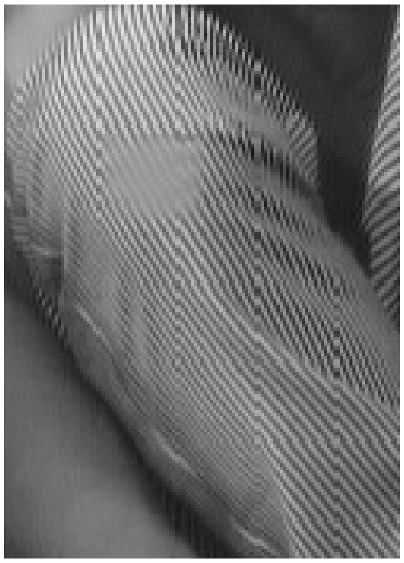

(e)

Figure 3.6: Small Rectangle Inpainting. Image (a) is the original image, image (b) is the damaged image. Images (c), (d) and (e) are the results obtained by the local TV, non-local TV and our method respectively. The PSNR values for the local, non-local and our result are 32.8517, 44.1469 and 44.2848 respectively. The simulation took 443 seconds, and there were 13 iterations in the two-step scheme. We used $C_1 = 700$, $dt = 0.01$, $\sigma = 4$, 31×31 neighborhood for feature vector calculation, and calculated 500 eigenvectors. We update W every iteration. The nonlocal TV inpainting took 1882 seconds.


(a)


(b)


(c)


(d)

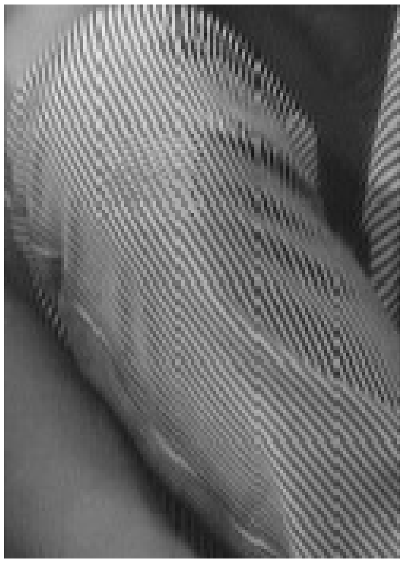
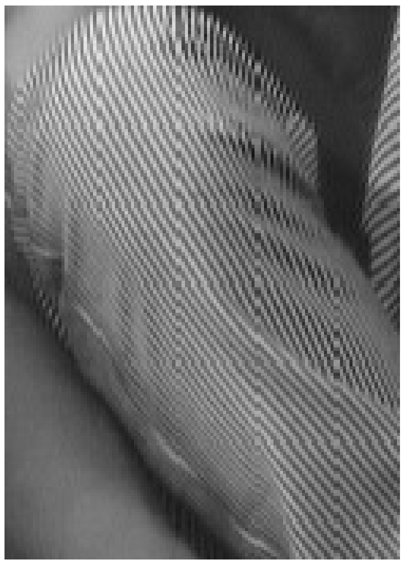
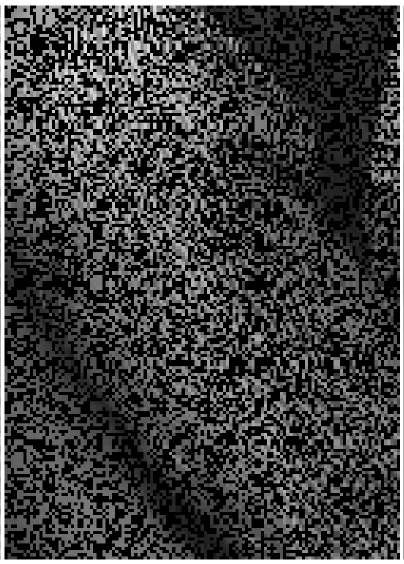

(e)

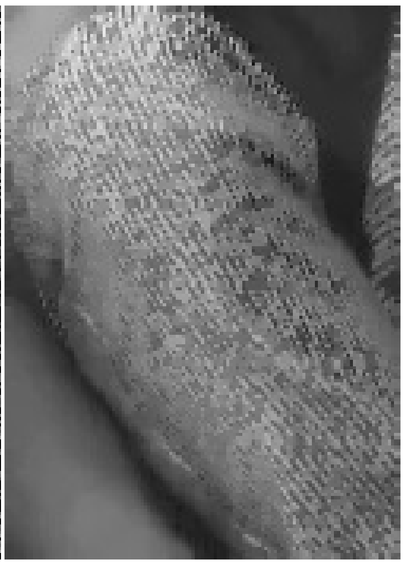
Figure 3.7: Large Rectangle Inpainting. Image (a) is the original image, image (b) is the damaged image. Images (c), (d) and (e) are the results obtained by the local TV, non-local TV and our method respectively. The PSNR values for the local, non-local and our result are 31.3673, 35.0663 and 37.0315 respectively. The simulation took 832 seconds, and there were 13 iterations in the two-step scheme. We used $C_1 = 700$, $dt = 0.014$, $\sigma = 4$, 45×45 neighborhood for feature vector calculation, and calculated 500 eigenvectors. We update W every iteration. The nonlocal inpainting took 3397 seconds.



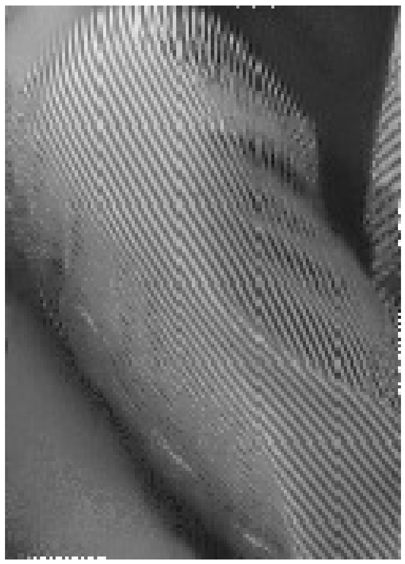
(a)



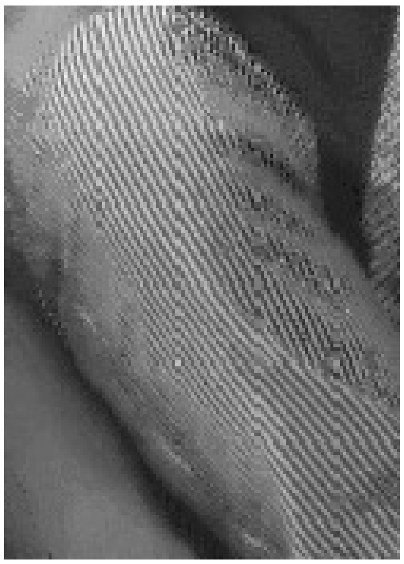
(b)



(c)



(d)



(e)

Figure 3.8: .
50% Random Inpainting. Image (a) is the original image, image (b) is the damaged image with 50% randomly distributed missing pixels. Images (c), (d) and (e) are the results obtained by the local TV, non-local TV and our method respectively. The PSNR values for the local, non-local and our result are 23.6049, 27.8196 and 27.1651 respectively. The simulation took 333 seconds, and there were 50 iterations in the two-step scheme. We used $C_1 = 700$, $dt = 0.005$, $\sigma = 4$, 7×7 neighborhood for feature vector calculation, and calculated 400 eigenvectors. We update W every iteration. The nonlocal inpainting took 1402 seconds.

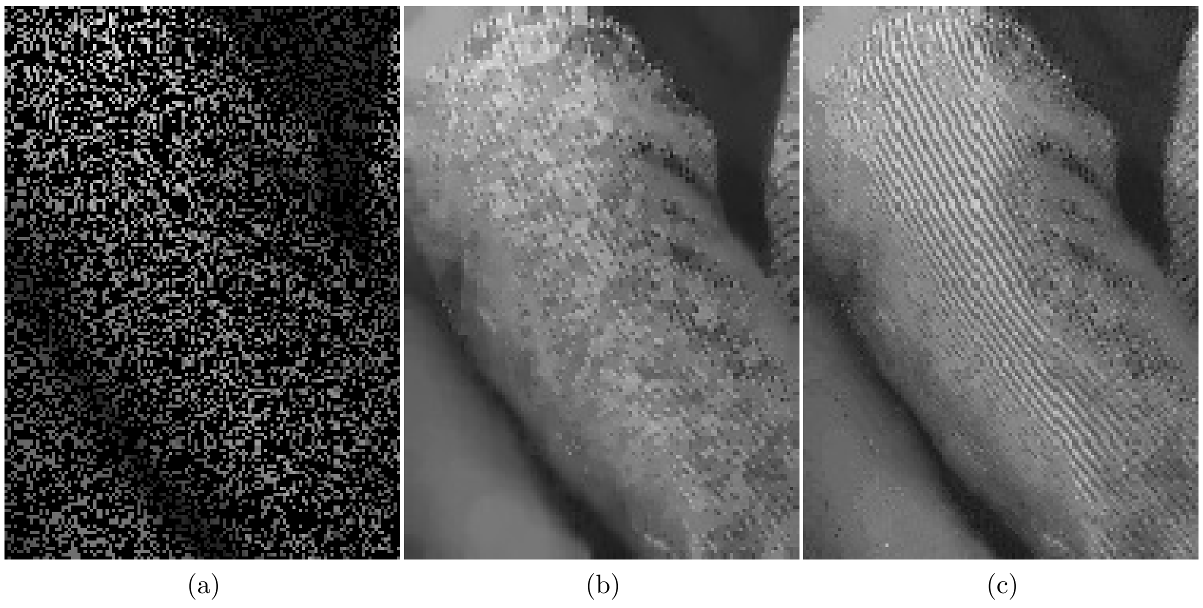


Figure 3.9: 35% Random Inpainting. Image (a) is a damaged image with 35% randomly distributed missing pixels. Images (b) and (c) are the results obtained by the local TV, and our method respectively. The PSNR values for the local, non-local and our result are 22.6530 and 24.1266 respectively. The simulation took 1200 seconds, and there were 150 iterations in the two-step scheme. We used $C_1 = 700$, $dt = 0.012$, $\sigma = 4$, 7×7 neighborhood for feature vector calculation, and calculated 500 eigenvectors. We update W every other iteration.

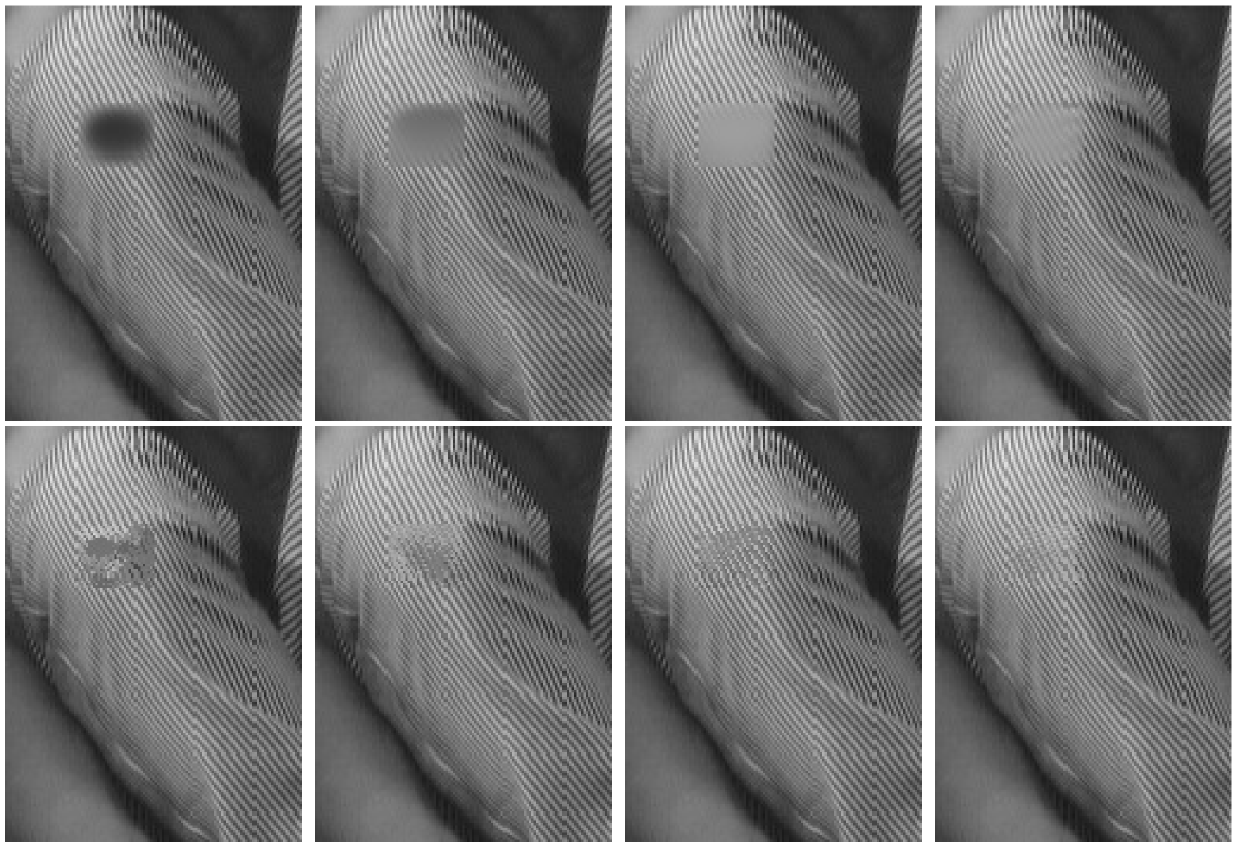


Figure 3.10: The comparison of our method (the bottom row) and the non-local TV inpainting (the top row) after 2, 5, 8 and 13 iterations

CHAPTER 4

Conclusion

In this work we presented different variations of the MBO scheme originally proposed in [7]. Our work demonstrates efficiency and versatility of the MBO approach, and it has inspired other researchers to extend the MBO scheme beyond its traditional form. The multiclass segmentation appears to be a natural extension of our data classification algorithm. The MBO numerical schemes are also applied to the modularity function minimization, a popular technique for counting and identifying communities in a graph. Detecting "communities" in a graph, is a very important problem in sociology, computer science and many other disciplines. In this chapter we also present a spline method for the density estimation problem discussed in Chapter 2.

4.1 Multiclass segmentation

The multiclass graph-based segmentation is one of the most important problems in machine learning. As we show in 3.2.4, it is possible to construct a weighted graph based on the similarity between the data points. In such a graph structure, certain groups of data points will form clusters based on the connectivity of the data points within a group. The goal is to detect such clusters. This problem is closely related to the image segmentation, thus the techniques developed be the imaging community present an invaluable source of ideas for the multiclass segmentation. The authors of [12] utilize the Ginzburg-Landau functional to create a diffuse interface model for the multiclass semi-supervised segmentation of high dimensional data.

In 3.1 we introduce a graph segmentation function that plays an important role in the model we propose in there. The authors of [12] generalized this notion by introducing a label vector $\mathbf{U} = (\mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_{N_D})^T$ where N_D is a number of data points. For each data point i there is a vector $\mathbf{u}_i \in \mathbb{R}^K$ where the kth component of $\mathbf{u}_i$ measures the likelihood of data point i being in the class k. For each data point i vector $\mathbf{u}_i$ is an element of the Gibbs simplex Σ^K defined as:

$$\Sigma^K = \left\{ (x_1, ..., x_K) \in [0,1]^K, \sum_{i=1}^{K} x_i = 1 \right\}. \tag{4.1}$$

In its general form, the energy functional used for data classification can be written as:

$$E(\psi) = \|\psi\|_a + \mu \|\psi - \hat{\psi}\|_b^p \tag{4.2}$$

where $\|\psi\|_a$ is the regularization term, and $\|\psi - \hat{\psi}\|_b^p$ is the regularization term that incorporates the known classification values $\hat{\psi}$. In [12], the proposed model uses the multiclass version of the Ginzburg-Landau functional as a regularizer and a standard L^2 norm fidelity term

$$E(\mathbf{U}) = \frac{\epsilon}{2}\langle \mathbf{U}, L_s \mathbf{U} \rangle + \frac{1}{2\epsilon} \sum_{i \in V} \left(\prod_{k=1}^{K} \frac{1}{4} \|\mathbf{u}_i - \mathbf{e}_k\|_{L_1}^2 \right) + \sum_{i \in V} \frac{\mu_i}{2} \|\mathbf{u}_i - \hat{\mathbf{u}}_i\|^2, \tag{4.3}$$

where

$$\langle \mathbf{U}, L_s \mathbf{U} \rangle = trace(\mathbf{U}^T L_s \mathbf{U}), \tag{4.4}$$

$\mathbf{e}_k$ is a unit vector whit kth component being equal to 1 and $\hat{\mathbf{u}}_i$ is set to be equal to $\mathbf{e}_k$ if node i is known to be in class k. There are two different schemes proposed in [12] that can be used for the energy minimization of (4.3), one that uses a convex splitting and the other one that uses the MBO splitting. In the case of convex splitting function 4.3 is decomposed into convex and concave parts as:

$$\begin{aligned}
E(\mathbf{U}) &= E_{convex}(\mathbf{U}) + E_{concave}(\mathbf{U}), \\
E_{convex}((U)) &= \frac{\epsilon}{2}\langle \mathbf{U}, L_s \mathbf{U} \rangle + \frac{C}{2}\langle \mathbf{U}, \mathbf{U} \rangle, \\
E_{concave}((U)) &= \frac{1}{2\epsilon} \sum_{i \in V} \left(\prod_{k=1}^{K} \frac{1}{4} \|\mathbf{u}_i - \mathbf{e}_k\|_{L_1}^2 \right) + \sum_{i \in V} \frac{\mu_i}{2} \|\mathbf{u}_i - \hat{\mathbf{u}}_i\|^2 - \frac{C}{2}\langle \mathbf{U}, \mathbf{U} \rangle,
\end{aligned} \tag{4.5}$$

with $C \in \mathbb{R}$ denoting a constant that guarantees the convexity of E_{convex} and the concavity of $E_{concave}$. The time discretization scheme for gradient descent of the convex splitting is

$$U_{ik}^{n+1} + dt\frac{\delta E_{convex}}{\delta U_{ik}}(U_{ik}^{n+1}) = U_{ik}^{n} - dt\frac{\delta E_{concave}}{\delta U_{ik}}. \tag{4.6}$$

When the functional derivatives are evaluated, equation 4.6 becomes:

$$\mathbf{U}^{n+1} + dt(\epsilon L_s \mathbf{U}^{n+1} + C\mathbf{U}^{n+1}) = \mathbf{U}^n - dt\left(\frac{1}{2\epsilon}\mathbf{T}^n + \mu(\mathbf{U} - \hat{\mathbf{U}}) - C\mathbf{U}^n\right), \tag{4.7}$$

where

$$T_{ik} = \sum_{l=1}^{K}(1 - 2\delta_{kl})\|\mathbf{u}_i - \mathbf{e}_l\|_{L^1}\prod_{\substack{m=1\\m\neq l}}^{K}\frac{1}{4}\|\mathbf{u}_i - \mathbf{e}_m\|_{L^1}, \tag{4.8}$$

μ is a diagonal matrix with elements μ_i, and $\hat{\mathbf{U}} = (\hat{\mathbf{u}}_1, ...), \hat{\mathbf{u}}_{N_D})^T$. Solving (4.7) for $\mathbf{U}^{n+1}$ gives the equation:

$$\mathbf{U}^{n+1} = \mathbf{B}^{-1}\left((1 + Cdt)\mathbf{U}^n - \frac{dt}{2\epsilon}\mathbf{T}^n - dt\mu(\mathbf{U}^n - \hat{\mathbf{U}})\right) \tag{4.9}$$

where

$$\mathbf{B} = (1 + Cdt)\mathbf{I} + \epsilon dt L_s. \tag{4.10}$$

Equation (4.9) represents a numerically stable implicit scheme.
Another way to minimize 4.3 is to generalize the MBO scheme, similarly to the graph version of the MBO scheme we proposed in 3.4. The multiclass MBO scheme, just like its standard counterpart can be divided into two phases, the heat propagation phase and the thresholding phase, and the algorithm consists of alternating between the two:

1. Diffusion:

$$\frac{\mathbf{U}^{n+\frac{1}{2}} - \mathbf{U}^n}{dt} = -L_s\mathbf{U}^n - \mu(\mathbf{U}^n - \hat{\mathbf{U}}). \tag{4.11}$$

2. Thresholding:

$$\mathbf{u}_i^{n+1} = \mathbf{e}_k, \tag{4.12}$$

where vector $\mathbf{e}_k$ is the vector closest to $\mathbf{u}_i^{n+\frac{1}{2}}$.

Both (4.9) and (2) are tested on multiple benchmark data sets, such as MNIST, COIL and WebKB. They achieve excellent results in term of accuracy and usually outperform all state-of-the-art methods. On all of the data sets, the multiclass MBO reaches convergence in fewer iterations.

4.2 Network modularity optimization using the MBO scheme

Communities of a graph are defined as groups of nodes with a higher density of connections within them than between them. Research on graph communities is focused on decomposing a graph into (possibly overlapping) communities. There are numerous application of this approach, such as in studies of the social networks, legislation cosponsorship in the United States Congress, functional modules in biology networks and many others. A very popular way of detecting the communities of a graph is via maximizing the modularity function of a graph partition. The modularity function Q measures the fraction of total edge weight within each community minus the edge weight that would be expected if edges were places randomly. Let $g = \{g_i\}_{i=1}^N$ be a partition of a graph. Let the quantity g_i represent the community assignment of the node n_i, assuming there are $\hat{n}$ communities, where $\hat{n}$ is not greater than N. The modularity function Q of the partition g is defined as:

$$Q(g) = \frac{1}{2m} \sum_{i,j=1}^{N} \left(w_{ij} - \gamma \frac{k_i k_j}{2m} \right) \delta(g_i, g_j), \tag{4.13}$$

where w_{ij} is a weight if the edge between the nodes n_i and n_j, k_i is a degree of the node n_i, $2m = \sum_{i=1}^N k_i$, γ is a resolution parameter and the term $\delta(g_i, g_j) = 1$ if $g_i = g_j$ and $\delta(g_i, g_j) = 0$ otherwise. Intuitively, maximization of the modularity function should yield a good decomposition of the graph G. The maximization of the modularity function is an NP hard problem, and numerous heuristical algorithms are proposed to solve it.

In [34], Hu et. al. propose a reformulation of the modularity function using the TV norm and the l_2 norm of a function defined on a graph. Let $f : G \to \mathbb{R}$

$$\begin{aligned} \mid f \mid_{TV} &= \frac{1}{2} \sum_{i,j=1}^{N} w_{ij} \mid f_i - f_j \mid, \\ \|f\|_{l_2}^2 &= \sum_{i=1}^{N} k_i \mid f_i \mid^2, \\ mean(f) &= \frac{1}{2m} \sum_{i=1}^{N} k_i f_i, \end{aligned} \tag{4.14}$$

where $f_i = f(n_i)$. For a vector function $f = (f^{(1)}, ..., f^{(\hat{n})})$ the TV norm and the l_2 norm are defined as:

$$\begin{aligned} \mid f \mid_{TV} &= \sum_{l=1}^{\hat{n}} \mid f^{(l)} \mid_{TV}, \\ \|f\|_{l_2} &= \sum_{l=1}^{\hat{n}} \|f^{(l)}\|_{l_2}, \end{aligned} \tag{4.15}$$

and $mean(f) = (mean(f^{(1)}), ..., mean(f^{(\hat{n})}))$. Based on the previous discussion community l can be defined as the set $A_l = \{n_i \in G, g_i = l\}$, where $l \in \{1, 2, ..., \hat{n}\}$. Let $f^{(l)} : G \in \{0, 1\}$ be the indicator function of the community l. For each partition g a vector function called the partition function can be defined as $f = (f^{(1)}, ..., f^{(\hat{n})})$. Since the assumption that the communities do not overlap is made, the partition function f can be reformulated as $f : G \to V^{\hat{n}}$, where $V^{\hat{n}}$ is the standard basis of $\mathbb{R}^{\hat{n}}$. The authors of [34] prove the following theorem:

Theorem 4.1 (Hu et. al., 2013)**.** *Maximizing the modularity functional Q over all partitions that have at most $\hat{n}$ communities is equivalent to minimizing*

$$\mid f \mid_{TV} - \gamma \|f - mean(f)\|_{l_2}^2 \tag{4.16}$$

over all functions $f : G \in V^{\hat{n}}$.

Given the established connection between the TV norm and the modularity of the optimal partitioning the authors of [34] propose the Ginzburg-Landau relaxation of the modularity function:

$$H_\epsilon(f) = \frac{1}{2}\sum_{l=1}^{\hat{n}} \langle f^{(l)}, Lf^{(l)}\rangle + \frac{1}{\epsilon^2}\sum_{i=1}^{N} W_{multi}(f_i) - \gamma\|f - mean(f)\|_{l_2}^2, \tag{4.17}$$

where $\epsilon > 0$, $W_{multi} : \mathbb{R}^{\hat{n}} \to \mathbb{R}$ is a multi-well potential and L is the graph Laplacian we previously introduced in 1.6. Hu et. al. prove that the functional H_ϵ Γ-converges to the functional defined in 4.16 on the space $X = \{f | f : G \to \mathbb{R}^{\hat{n}}\}$. The gradient descent equation of 4.17 is

$$\frac{\partial f}{\partial t} = -(Lf^{(1)}, ..., Lf^{(\hat{n})}) - \frac{1}{\epsilon^2}\nabla W_{multi}(f) + \frac{\delta}{\delta f}(\gamma\|f - mean(f)\|_{l_2}^2). \tag{4.18}$$

Similarly to the variations MBO schemes we presented in 2 and 3 the authors of [34] propose a Modularity MBO scheme that alternates between the following two steps to obtain an approximate solution of 4.17:

1. In the diffusion step we apply

$$\frac{\partial f}{\partial t} = -(Lf^{(1)}, ..., Lf^{(\hat{n})}) + \frac{\delta}{\delta f}(\gamma\|f - mean(f)\|_{l_2}^2) \tag{4.19}$$

   on f^n with time τ_n and we repeat it for η time steps.

2. In the thresholding step:

$$f_i^{n+1} = e_{g_i}, \tag{4.20}$$

   where

$$g_i = \arg\max_{1\le l\le \hat{n}}\{\hat{f}_i^{(l)}\}. \tag{4.21}$$

In [34], the authors propose a convex-splitting scheme to solve 4.19. Since 4.19 is the gradient descent of $H_1 + H_2$, they take advantage of the splitting where $H_1(f) = \sum_{l=1}^{\hat{n}}\langle f^{(l)}, Lf^{(l)}\rangle$ is convex and $H_2(f) = -\gamma\|f - mean(f)\|_{l_2}^2$ is concave. In [34] there are two proposed implementation of the MBO scheme. The Recursive Modularity MBO is suitable for the cases when the expected number of communities is very large and the Multiple Input-$\hat{n}$

Modularity is suitable for the cases of graphs with fewer communities.

4.3 A spline method for density estimation

The idea to utilize splines with the polynomial basis functions to create a new version of the MPLE model introduced in Section 1.5 originates from the fact that the valid region can be elegantly incorporated into the model, and no additional penalty terms are needed. In terms of the simplicity of the model, the spline model we will introduce here has clear advantage over the modified TV MPLE models from [39].

Let $\triangle$ be a triangulation of the valid region D, and let us define a family of the polynomial spline functions of the degree m and smoothness r:

$$S_m^r(\triangle) = \{s \in C^r(D) | s \restriction_\tau \in \mathbf{P}_m, \tau \in \triangle\}. \tag{4.22}$$

The TV MPLE model is proposed that will be minimized over S_m^r:

$$\hat{d} = \arg\min_{d \in S_m^r} \left\{ \alpha \int_D \sqrt{\epsilon + \mid \nabla d(\mathbf{x}) \mid^2} + \sum_{j=1}^N \log \frac{1}{d(\mathbf{x}_j)}, d(\mathbf{x}) \geq 0, \int_D d(\mathbf{x}) = 1 \right\} \tag{4.23}$$

where d is a density function and ϵ is a parameter. In [42], the Bernstein-Bezier polynomials,B_{ijk}^τ are proposed to be a basis of the spline functions:

$$d(\mathbf{x}) = \sum_{\tau \in \triangle} \sum_{i+j+k=m} c_{ijk}^\tau B_{ijk}^\tau(\mathbf{x}). \tag{4.24}$$

Using this, the constraint $\int_D d(\mathbf{x}) = 1$ can be written as:

$$\sum_{\tau \in \triangle} \frac{A_\tau}{\binom{m+2}{2}} \sum_{i+j+k=m} c_{ijk}^\tau = 1, \tag{4.25}$$

where A_τ denotes the area of triangle τ. Also $d(\mathbf{x}) \geq 0$ is equivalent to $c_{ijk}^\tau \geq 0,\ i+j+k = m$. The model 4.23 can be written in a matrix form using the vector $\mathbf{c} = (c_{ijk}^\tau,\ i+j+k = m,\ \tau \in$

$\triangle$)

$$\min \Phi_\epsilon(\mathbf{c}) + \frac{1}{2\alpha}\Psi(\mathbf{c}), \tag{4.26}$$

where

$$\mathbf{Sc} = 0, \tag{4.27}$$

$$B\mathbf{c} = G, \tag{4.28}$$

where $\Phi_\epsilon(\mathbf{c}) = \sqrt{\epsilon + \mid \nabla d \mid^2}$ and $\Psi(\mathbf{c}) = \sum_i \log(\frac{1}{T(\mathbf{x}_i)})$. The algorithm to minimize 4.26 proposed in [42] is a variant of the Uzawa algorithm. At each iteration of the algorithm the following minimization problem is solved:

$$^{k+1} = \min_{\mathbf{c}} \Phi_\epsilon(\mathbf{c}) + \frac{1}{2\alpha}\Psi(\mathbf{c}) - \lambda^k \mathbf{c},\ \mathbf{Sc},\ B\mathbf{c} = G, \tag{4.29}$$

where $\lambda^{k+1} \in \mathbb{R}^n_+$ is a parameter. Following the minimization, the parameter λ^k is updated:

$$\lambda^{k+1} = \max\{\lambda^k - \rho \mathbf{c}^{k+1}, 0\}, \tag{4.30}$$

where ρ is a step size. The proof of convergence for this method is given in [42].

In this work we apply the variants of the MBO scheme to minimize the MPLE energy functional and the graph based Ginzburg-Landau function. Comparing to other numerical schemes iterations of the MBO scheme are simple and computationally inexpensive. Our graph based MBO scheme converges in up to ten times fewer iterations than the state-of-the-art nonlocal method. We are proud that our methods opened the way to several new applications of the MBO scheme.

References

[1] Nonlocal discrete regularization on weighted graphs: a framework for image and manifold processing. IEEE Trans. Image Process., 17:1047–1060, 2008.

[2] M. Bertalmio A. Bertozzi and G. Sapiro. Navier-stokes fluid dynamics and image and video inpainting. Proceedings of the International Conference Computer Vision and Pattern Recognition, IEEE, 2001.

[3] S. Esedoglu A. Bertozzi and A. Gillette. Inpainting by the cahn-hilliard equation. IEEE Trans. Image Process., 2007:285–291, 2007.

[4] T. Dupont A. Bertozzi, M. Brenner and L. Kadanoff. Singularities and similarities in interface flows. Trends and Perspectives in Applied Mathematics, Applied Mathematical Sciences, 100:155–208, 1994.

[5] B. Coll A. Buades and J. M. Morel. A non-local algorithm for image denoising. Proc. IEEE Computer Society on Computer Vision and Pattern Recognition, 2:60–65, 2005.

[6] C. Anderson. A raleigh-chebyshev procedure for findinf the smalles eigenvalues and associated eigenvectors of large sparce hermitian matrices. Journal of Computational Physics, 229:7477–7487, 2010.

[7] J. Bence B. Merriman and S. Osher. Diffusion generated motion by mean curvature. Proceedings of the Computational Crystal Growers Workshop, pages 73–83, 1992.

[8] A. Bertozzi and A. Flenner. Diffuse interface models of graphs for classification of high dimensional data. Multiscale Model. and Simul., 10:3:1090–1118, 2012.

[9] T. Buhler and M. Hein. Spectral clustering based on the graph p-laplacian. Proceedings of the 26th International Conference on Machine Learning, pages 81–88, 2009.

[10] F. Chung C. Folwkes, S. Belongie and J. Malik. Efficient spatiotemporal grouping using nyström method. CVPR, 2001.

[11] S. Belongie C. Folwkes and J. Malik. Spectral grouping using the nyström method. IEEE Transactions on Pattern Analysis and Machine Intelligence, 28:469–475, 2006.

[12] A. Bertozzi A. Flenner A. Percus C. Garcia-Cardona, K. Merkurjev. Fast multiclass segmentation using diffuse interface methods on graphs. CAM report.

[13] T. Chan and L. Vese. Active contours without edges. IEEE Transactions on Image Processing, 10:2:266–277, 2001.

[14] F. Chung. Spectral graph theory. CBMS Reg. Conf. Ser. Math. 92, 1997.

[15] G. Chung and L. Vese. Image segmentation using a multilayer level-set approach. Computing and visualizations in science, 12:6:267–285, 2009.

[16] M. Doob D. Cvetkovic and H. Sachs. Spectra of graphs. 3rd revised and enlarged edition. 1995.

[17] B. Schölkopf D. Zhou and T. Hofmann. Semi-supervised learning on directed graphs. 2005.

[18] J. Dobrosotskaya and A. Bertozzi. A wavelet-laplace variational technique for image deconvolution and inpainting. IEEE Trans. Image Process., 17:657–663, 2008.

[19] P. Drineas and M. W. Mahoney. On the nyström method for approximating a gram matrix for improved kernel-based learning. J. Mach. Learn. Res., pages 2153–2175, 2005.

[20] P. P. B. Eggermont and V. N. LaRiccia. Maximum Penalized Likelihood Estimation. Springer, 2001.

[21] T.F. Chan E.S. Brown and X.Bresson. A convex relaxation method for a class of vector-valued minimization problems with applications to mumford-shah segmentation. 2010.

[22] S. Esedoglu and R. March. Segmentation with depth but without detecting junctions. J. Math. Imaging Vision, 18:7–15, 2003.

[23] S. Esedoglu and Y. R. Tsai. Threshold dynamics for the piecewise constant mumford-shah. Journal of Computational Physics, 26:367–384, 2004.

[24] R. A. Fisher. On the mathematical foundations of theoretical statistics. Philos. Trans. Royal. Soc. London, pages 309–360, 1922.

[25] V. Caselles G. Facciolo, P. Arias and G. Sapiro. Exemplar-based interpolation of sparsely sampled images. EMMCVPR, 2009.

[26] T Goldstein G. Mohler, A. Bertozzi and S. Osher. Fast tv regularization for 2d maximum penalized likelihood estimation. Journal of Computational and Graphical Statistics, 20:2:479–491, 2011.

[27] S. Bougleux G. Peyre and L. Cohen. Non-local regularization of inverse problems. ECCV, pages 57–68, 2008.

[28] S. Gao and T.D. Bui. Image segmentation and selective smoothing by using mumford-shah model. IEEE Transactions on Image Processing, 14:10:1537–1549, 2005.

[29] F. Gibou and R. Fedkiw. A fast hybrid k-means level set algorithm for segmentation. Proceedings of the 4th Annual Hawaii International Conference on Statistics, 2005.

[30] G. Gilboa and S. Osher. Nonlocal linear image regularization and supervised segmentation. Multiscale Model. Simul, 6:595–630, 2007.

[31] G. Gilboa and S. Osher. Nonlocal operators with applications to image processing. Multiscale Model. Simul, 7:1005–1028, 2008.

[32] T. Goldstein and S. Osher. The split bregman method for l1-regularized problem. SIAM J. Imaging Sci., 2:323–343, 2009.

[33] L. Grady and E. L. Schwartz. Isoperimetric graph partitioning for image segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence, 28:469–475, 2006.

[34] M. Porter H. Hu, T. Laurent and A. Bertozzi. A method based on total variation for network modularity optimization using the mbo scheme. CAM report.

[35] R. Chan J. F. Cai and Z. Shen. A framelet based image inpainting algorithm. Appl. Comput. Harmon. Anal., pages 131–149, 2008.

[36] M. Elad J. Mairal and G. Sapiro. Sparse representation for color image restoration. IEEE Trans. Image Process., 17:53–69, 2008.

[37] R. Koenker and I. Mizera. Density estimation by total variation regularization, advances in statistical modeling and inference. Essays in Honor of Kjell A. Doksum, World Scientific, pages 613–634, 2006.

[38] R. V. Kohn and P. Sternberg. Local minimizers and singular pertubations. Proc. Roy. Soc. Edinburgh Set. A, 11:69–84, 1989.

[39] T. Wittman G. Mohler L. Smith, M. Keegan and A. Bertozzi. Improving density estimation by incorporating spatial information. URASIP Journal on Advances in Signal Processing, 2010.

[40] S. Osher L.I. Rudin and E. Fatemi. Nonlinear total variation based noise removal algorithm. Physica D, 60:259–268, 1992.

[41] G. Sapiro M. Bertalmio, L. Vese and S. Osher. Simultaneous structure and texture inpainting. IEEE Trans. Image Process., 12:882–889, 2003.

[42] A. Bertozzi M. Lai and T. Kostic. Improving analysis and computation for mathematical crime study.

[43] K. Rothley M. M. Naeini, G. Dutton and G. Mori. Action recognition of insects using spectral clustering. Proceedings of the IAPR Conference on Machine Vision Applications, 2007.

[44] A. Bertozzi M. Moller, T. Wittman and M. Burger. A variational approach for sharpening high dimensional images. SIAM Journal of Imaging Sciences, 5:1:150–178, 2012.

[45] B. Merriman and S. Ruuth. Diffusion generated motion of curves and surfaces. Journal of Computational Physics, 225:2:2267–2282, 2007.

[46] B. Mohar. The laplacian spectrum of graphs. Graph Theory, Combinatorics and Applications, 2:871–898, 1991.

[47] D. Mumford and J. Shah. Optimal approximations by piecewise smooth functions and associated variational problems. Comm. Pure Appl. Math., 42:577–658, 1989.

[48] S. Osher and J. Shen. Digitalized pde method for data restoration. In Analytical-Computational methods in Applied Mathematics, E. G. A. Anastassiou, Ed. New York: Chapman & Hall/CRC, pages 751–771, 2000.

[49] V. Caselles P. Arias and G. Sapiro. A variational framework for nonlocal image inpainting. EMMCVPR 2009, Proceedings, 2009.

[50] C. A. Ratanamahatana and G. Gunopulos. Scaling up naive bayesian clasifier: Using decision trees for feature selection. IEEE International Conference on Data Mining, 2002.

[51] S. Ruuth. Efficient algorithms for diffusion-generated motion by mean curvature. Journal of Computational Physics, 144:603–625, 1998.

[52] F. Chung S. Belongie, C. Fowles and J. Malik. Partitioning with indefinite kernels using the nyström extension. ECCV Copenhagen, 2002.

[53] S. J. Ruuth S. Esedoglu and R. Tsai. Threshold dynamics for high geometric motions. Interfaces Free Bound., 10:263–282, 2008.

[54] S. Ruuth S. Esedoglu and R. Tsai. Diffusion generated motion using signed distance function. Journal of Computational Physics, 229:4:1017–1042, 2010.

[55] S. Sardy and P. Tseng. Density estimation by total variation penalized likelihood driven by the sparsity l1 information criterion. Scandinavian Journal of Statistics, 37:2:321–337, 2010.

[56] H. Schaeffer and S. Osher. A low patch-rank interpretation of texture. SIAM Journal on Imaging Sciences, 6:1:226–262, 2013.

[57] J. Shi and J. Malik. Normalized cuts and image segmentation. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 22:8:888–905, 2000.

[58] B. W. Silverman. Kernel density estimation using the Fast Fourier Transform. Statistical Algorithm, AS176, Applied Statistics, 31:93–97, 1982.

[59] B. W. Silverman. Density Estimation for Statistics and Data Analysis. Chapman & Hall/CRC, April 1986.

[60] B. Song and T. Chan. A fast algorithm for level set based optimization. UCLA CAM report.

[61] A. Szlam and X. Bresson. Total variation-based graph clustering algorithm for cheeger ratio cuts. Proceedings of the 27*th* International Conference on Machine Learning, pages 1039–1046, 2010.

[62] F. Benezit T. Cour and J. Shi. Spectral segmentation with multiscale graph decomposition. IEEE Computer Society Conference on Computer Vision, 2:1124–1131, 2005.

[63] M. Nikolova T.F. Chan and S.Esedoglu. Algorithms for finding global minimizers of image segmentation and denoising models. 66:16–32, 2006.

[64] L.A. Vese and T.F. Chan. A multiphase level set framework for image segmentation using the mumford and shah model. International Journal of Computer Vision, 50:3:271–293, 2002.

[65] U. von Luxberg. A tutorial on spectral clustering. Statist. Comput., 17:395–416, 2007.

[66] Z. Wu and R. Leahy. An optimal graph theoretic approach to data clustering: theory and its application to image segmentation. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 15:11:1101–1113, 1993.

[67] X. Bresson X. Zhang, M. Burger and S. Osher. Bregmanized nonlocal regularization for deconvolution and sparse reconstruction. SIAM Journal on Imaging Sciences, pages 253–276, 2010.

[68] S. Osher Y. Lou, X. Zhang and A.Bertozzi. Image recovery via nonlocal operators. J. Sci. Comput., 42:185–197, 2010.

[69] L. Zelnik-Manor and P. Perona. Self-tuning spectral clustering. Inf. Process. Syst., 17:1601–1608, 2004.

[70] X. Zhang and T. Chan. Wavelet inpainting by nonlocal total variation. Inverse Probl. Imaaging, 4:1:191–210, 2010.

[71] D. Zhou and B. Schölkopf. Learning from labeled and unlabeled data on a directed graph. Proceedings of the 22^{nd} International Conference on Machine Learning, pages 1041–1048, 2005.

[72] D. Zhou and B. Schölkopf. Regularization on discrete spaces. Pattern Recognition, pages 361–368, 2005.

[73] D. Zhou and B. Schölkopf. Descrete regularization. Semi-Supervised Learning, pages 221–232, 2006.