DIFFUSE INTERFACE MODELS ON GRAPHS FOR CLASSIFICATION OF HIGH DIMENSIONAL DATA*

ANDREA L. BERTOZZI[†] AND ARJUNA FLENNER[‡]

Abstract. There are currently several communities working on algorithms for classification of high dimensional data. This work develops a class of variational algorithms that combine recent ideas from spectral methods on graphs with nonlinear edge/region detection methods traditionally used in in the PDE-based imaging community. The algorithms are based on the Ginzburg-Landau functional which has classical PDE connections to total variation minimization. Convex-splitting algorithms allow us to quickly find minimizers of the proposed model and take advantage of fast spectral solvers of linear graph-theoretic problems. We present diverse computational examples involving both basic clustering and semi-supervised learning for different applications. Case studies include feature identification in images, segmentation in social networks, and segmentation of shapes in high dimensional datasets.

Key words. Nyström extension, diffuse interfaces, image processing, high dimensional data

AMS subject classifications. Insert AMS subject classifications.

The work develops a new algorithm for classification of high dimensional data on graphs. The method bridges a gap between the PDE-image processing community, methods related to L1 compressive sensing, and the graph-theoretic community. The algorithm is inspired by diffuse interface methods that have long been used in physical science modeling of problems with free boundaries and interfaces. These same ideas also arise in compressive sensing because of the connection between the diffuse interface energies and TV minimization. Thus it is natural to extend these concepts beyond classical continuum modeling to the discrete graph framework. By implementing various techniques for fast linear algebra and eigenfunction decomposition, we can design methods that are quite good in terms of performance and speed. They are also versatile and we illustrate examples on a variety of datasets including congressional voting records, high dimensional test data, and machine learning in image processing.

This paper is structured as follows: In Section 1 we review diffuse interface methods in Euclidean space and convex splitting methods for minimization. These well-known constructions make heavy use of the classical Laplace operator and our new algorithms involve extensions of this idea to a more general graph Laplacian. Section 2 reviews some of the notation and definitions of the graph Laplacian and this discussion contains a level of detail appropriate for readers less familiar with this machinery. Included in this section is a review of segmentation using spatial clustering and a discussion of various normalization conventions for these linear operators on graphs, in connection to real world problems such as machine learning in image analysis. The rest of the paper explains the main computational algorithm and presents different examples involving both sparse connectivity and non-sparse connectivity of the graph. The algorithms have a multi-scale flavor due to (a) the different scales

*This research is supported by ONR grants N000140810363, N000141010221, N000141210040, and N0001411AF00002, NSF grants DMS-0914856 and DMS-1118971 and AFOSR MURI grant FA9550-10-1-0569.

[†]Department of Mathematics, 520 Portola Plaza, University of California Los Angeles, Los Angeles, CA 90095, `bertozzi@math.ucla.edu`

[‡]Naval Air Weapons Center, Physics and Computational Sciences, China Lake, CA, USA `arjuna.flenner@navy.mil`

inherent in diffuse interface methods and (b) the role of scale in the eigenfunctions and eigenvalues of the graph Laplacian.

1. Background on diffuse interfaces, image processing, and convex splitting methods.

Diffuse interface models in Euclidean space are often built around the Ginzburg-Landau functional

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

where W is a double well potential. For example $W(u) = \frac{1}{4}(u^2-1)^2$ has minimizers at plus and minus one. The operator ∇ denotes the spatial gradient operator and the first term in GL is $\epsilon/2$ times the H^1 semi-norm of u. The small parameter ϵ represents a spatial scale, the diffuse interface scale.

The model is called "diffuse interface" because there is a competition between the two terms in the energy functional. Upon minimization of this functional, the double well will force u to go to either one or negative one, however the H^1 term forces u to have some smoothness, thereby removing sharp jumps between the two minima of W. The resulting minimization leads to regions where u is approximately one, regions where it is approximately negative one, and a very thin, $O(\epsilon)$ scale transition regions between the two. Thus the minimizer appears to have two phases with an interface between them. For this reason such models are often referred to as "phase field" models when one considers dynamic evolution equations built around this energy functional. There are several interesting features of GL minimizers. For example, the transition region between the two phases typically has some length associated with it and the GL functional is roughly proportional to this length. This can be made rigorous by considering the notion of Gamma convergence of the Ginzburg-Landau functional. It is known to converge [36] to the total variation semi-norm,

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

Diffuse interface models with spatial gradient operators are multi-scale because several length scales exist in the model, the smallest being the diffuse interface scale ϵ. The construction of the GL functional requires ϵ to have units of length so that the two terms have balanced units. Also, note that the double well typically restricts u to take on integral order values (between zero and one).

Multiple timescales exist in evolution equations that make use of this functional. The most common examples are the Allen-Cahn equation, which is the L^2 gradient descent of this functional, and the Cahn-Hilliard equation, which is a gradient descent in the H^{-1} inner product. Recent work has gone into developing efficient numerical schemes to track these dynamics [35, 50, 34, 8]; the same ideas can be used to speed up variational algorithms based on these functionals.

1.1. The connection between GL and image processing.

Perhaps the most important motivation for the method proposed in this paper is the existing literature on the use of the GL functional in image processing. We review some of that literature for the reader not directly familiar with it to better motivate the choice of methods proposed here. The Ginzburg-Landau functional is sometimes used in image processing as an alternative or a relative to the TV semi-norm. Because of the gamma convergence, these two functionals can sometimes be interchanged. Moreover, the highest order term in the GL functional is purely quadratic allowing for fast minimization schemes in some problems. Recent advances in TV minimization procedures, e.g. split Bregman and graph cut methods [31, 16], have made this less necessary, nevertheless there are cases where the pure TV case is not enough and the diffuse interface version may be a simpler method.

One example of the GL functional in image processing is the motivation for the Esedoglu-Tsai [23] threshold method for Chan-Vese segmentation [12]. The construction of their method is directly built on the GL functional, rather than the TV method of the original Chan-Vese paper [12]. We also note that this method was inspired by the original Merriman-Bence-Osher paper [40]. Another example is work by Riccardo March [21, 11] and Esedoglu [20, 22]. The work by Ambrosio-Tortorelli [3] is well known in image processing for diffuse interface approximations.

In a typical application we want to minimize an energy functional of the form

$$E(u) = GL(u) + \lambda F(u, u_0)$$

where $F(u, u_0)$ is a fitting term to known data. In the case of denoising, $F(u, u_0)$ is often just an L^2 fit, $\int (u-u_0)^2$. In the case of deblurring it is $\int (K * u - u_0)^2$, or the L^2 of the blurred solution with the data. For inpainting we often have an L^2 fit to known data in the region where the data is known, i.e. $\int_\Omega (u-u_0)^2$. In some instances in the above a different norm is used, e.g. L^1 or other norms. In the case of Cahn-Hilliard based inpainting, the method is not strictly a gradient flow [7, 6], but rather based on gradient flows. In fact the method is a sort of hybrid in which the L^2 least squares fitting term is paired with the Cahn-Hilliard H^{-1} dynamics. The result is a method that achieves both required boundary conditions for inpainting, namely, continuation of both greyscale information and direction of edges across the inpainting domain. The higher order evolution is important for that application and is related to the geometry of the problem. The CH inpainting problem is a good example of a method where new fast algorithms for TV minimization have yet been developed due, in part, to the unusual boundary conditions. The convex splitting method (see below) allows for very fast inpainting using a CH method and the GL functional.

The energy $E(u)$ can be minimized in the L^2 sense using a gradient descent, which gives us a modified Allen-Cahn equation

$$u_t = -\frac{\delta GL}{\delta u} - \lambda \frac{\delta F}{\delta u} = \epsilon \Delta u - \frac{1}{\epsilon} W'(u) - \lambda \frac{\delta F}{\delta u}.$$

This can be evolved to steady state to obtain a local minimizer of the energy E. We note that in general, especially for the GL functional, that E is not convex and thus may have multiple local energy minima. The result is that the long time behavior of the solution of the modified Allen-Cahn equation may depend on the initial condition.

1.2. Convex splitting and time stepping of the GL functional. One of the reasons to choose the GL functional instead of TV is that the minimization procedure for GL often involves the first variation of GL for which the highest order term, involving the Laplace operator, is linear. Thus if one has fast solvers for the Laplace operator or relatives of it, one can take advantage of this in designing convex splitting schemes discussed below. This class of schemes will be developed in the context of graph-based methods later in this paper. To motivate the ideas for the reader unfamiliar with the technique we provide some background on this class of methods in relation to work involving the GL functional.

A particular class of fast solvers are ones in which the Laplacian can be transformed so that the operator diagonalizes. A classical example would be the Fast Fourier Transform which transforms the Laplace operator to multiplication by $-|k|^2$

where k is the wave number of the Fourier mode. The FFT works because the Fourier modes are also eigenfunctions of the Laplace operator. An example of this use in long-time solutions of the Cahn-Hilliard equation is discussed in detail in [50]. Other recent advances for fast Poisson solvers could be used as well (see e.g. [39]). In our graph-based examples we use fast methods for directly diagonalizing the graph Laplacian, either through standard sparse linear agebra routines, or in the case of fully connected weighted graphs, Nyström extension methods.

Convex splitting schemes are based on the idea that an energy functional can be written as the sum of convex and concave parts,

$$E(u) = E_{vex}(u) - E_{cave}(u)$$

where this decomposition is certainly not unique because we can add and subtract any convex function and not change E but certainly change the convex/concave splitting. The idea behind convex splitting for the gradient descent problem is to perform a time stepping scheme in which the convex part is done implicitly and the concave part explicitly. More precisely the convex splitting scheme is

$$\frac{u^{n+1} - u^n}{dt} = -\frac{\delta E_{vex}}{\delta u}(u^{n+1}) + \frac{\delta E_{cave}}{\delta u}(u^n). \tag{1.1}$$

The art then lies in choosing the splitting so that the resulting scheme is stable and also computationally efficient to solve. This method was popularized by a well-known but unpublished manuscript by David Eyre [24]. It has been used to solve the Cahn Hilliard equation on large domains and on long time intervals [50] and also in imaging applications involving Cahn-Hilliard [7] and a wavelet version of the method proposed here for graphs [18]. This same idea has also been directly discussed in the context of general minimization procedures for nonconvex functionals [56].

2. Generalizations of the GL functional to graphs.

One can consider a generalization of the GL functional to graphs. This is in the same spirit as the work of Dobrosotskaya and the first author [18] generalizing the GL functional to wavelets. In their work they construct a linear operator with similar features to the Laplace operator, however the eigenfunctions are the wavelet basis, for some choice of wavelets. The natural choice of eigenvalues are ones that scale like the inverse square of the length scale of the wavelet basis functions, much in the same way that the eigenvalues of the Laplace operator are the inverse square of the period of the corresponding eigenfunction.

In this section we describe how to generalize the Ginzburg Landau functional, or more precisely its L^2 gradient flow, to the case of functions defined on graphs [14]. One challenge is the normalization of the Laplacian due to the fact that we are working with purely discrete functionals that may not have a direct spatial embedding. We include some details for the reader not familiar with the graph-based literature. This section is organized into five subsections discussing how the graph Laplacian can be incorporated into a Ginzburg Landau type functional. First, we introduce the Graph Laplacian using weight functions. We specifically introduce the notation used in this paper. In order to provide insight into our algorithm, we continue in the next subsection with a short discussion on previous segmentation algorithms using the graph Laplacian. Next, the benefits of the normalized graph Laplacian is discussed. The graph based Ginzburg Landau functional for segmentation is then introduced, and the last subsection mentions techniques to create weight functions and thus build interesting Graph Laplacians.

2.1. Graph definitions and notation. Consider an undirected graph $G = (V, E)$ with vertex set $V = \{\nu_n\}_{n=1}^N$ and edge set E. A weighted undirected graph [14] has an associated weight function $w : V \times V \to R$ satisfying $w(\nu, \mu) = w(\mu, \nu)$ and $w(\nu, \mu) \geq 0$. The degree of a vertex $\nu \in V$ is defined as

$$d(\nu) = \sum_{\mu \in V} w(\nu, \mu). \tag{2.1}$$

The degree matrix D can then be defined as the $N \times N$ diagonal matrix with diagonal elements $d(\nu)$.

The size of a subset $A \subset V$ will be important for segmentation using graph theory, and there are two important size measurements. For $A \subset V$ define

$$|A| := \text{ the number of vertices in } A, \tag{2.2}$$

$$vol(A) := \sum_{\nu \in A} d(\nu). \tag{2.3}$$

The topology of the graph also plays a role. A subset $A \subset V$ of a graph is connected if any two vertices in A can be joined by a path such that all the points also lie in A. A subset of A is called a connected component if it is connected and if A and $\overline{A}$ are not connected. The sets $A_1, A_2, \dots, A_k$ form a partition of the graph if $A_i \cap A_j = \emptyset$ and $\cup_k A_k = V$.

The graph Laplacian is the main tool for graph theory based segmentation. Define the graph Laplacian $L(\nu, \mu)$ as

$$L(\nu, \mu) = \begin{cases} d(\nu) & \text{if } \nu = \mu, \\ -w(\nu, \mu) & \text{otherwise.} \end{cases} \tag{2.4}$$

The graph Laplacian can be written in matrix form as $L = D - W$ where W is the matrix $w(\nu, \mu)$. The following definition and property of L are important:

1. (Quadratic Form) For every vector $u \in R^N$

$$\langle u, Lu \rangle = \frac{1}{2} \sum_{\mu, \nu \in V} w(\nu, \mu)(u(\nu) - u(\mu))^2 \tag{2.5}$$

2. (Eigenvalue) L has N non-negative, real valued eigenvalues with $0 = \tilde{\lambda}_1 \leq \tilde{\lambda}_2 \leq \cdots \leq \tilde{\lambda}_N$, and the eigenvector of $\tilde{\lambda}_1$ is the constant N dimensional one vector 1_N.

The Quadratic form is exploited to define a minimization procedure as in the Allen-Chan equation above. The Eigenvalue condition gives limitations on the spectral decomposition of the matrix L. These spectral properties are essential for spectral clustering algorithms discussed below.

There are two popular normalization procedures for the graph Laplacian, and the normalization has segmentation consequences [14, 51]. The normalization that will be used in this work is the symmetric Laplacian L_s defined as

$$L_s = D^{-1/2} L D^{-1/2} = I - D^{-1/2} W D^{-1/2}. \tag{2.6}$$

The symmetric Laplacian is named as such since it is a symmetric matrix. The random walk Laplacian is another important normalization given by

$$L_w = D^{-1} L = I - D^{-1} W. \tag{2.7}$$

The random walk Laplacian is closely related to discrete Markov processes, and we discuss the use of the random walk Laplacian in section 5.2.

The spectrum of the graph Laplacian is used for graph segmentation, and some well known results are collected here for future reference. The spectrum of L_s and L_w are the same, but the eigenvectors are different. The easily verifiable spectral relationships between L_w and L_s are listed below.

1. $\tilde{\lambda}$ is an eigenvalue of L_w if and only if $\tilde{\lambda}$ is an eigenvalue of L_s.
2. ψ is an eigenvector of L_w if and only if $D^{1/2}\psi$ is an eigenvector of L_s.
3. $\tilde{\lambda}$ is an eigenvalue of L_w with eigenvector ψ if and only if $L\psi = \tilde{\lambda} D\psi$.

2.2. Choice of similarity function. Here we discuss how to construct the weight functions $w(x, y)$, sometimes referred to as similarity functions, for specific applications involving high dimensional data. There are two factors to consider when choosing $w(x, y)$. First, the choice of weight function must reflect the desired outcome. For segmentation, this typically involves choosing an appropriate metric on a vector space. Our examples below used the standard Euclidean norm, but other norms may be more appropriate. For example, the angle norm may work better for segmentation of hyperspectral images. A second consideration is algorithm speed. The segmentation algorithms below requires the diagonalization of $w(x, y)$, and this step is often the rate limiting procedure. There are two main methods to obtain speed in the diagonalization. The first method is to use the Nyström extension described in section 3.2. This method does not require a modification of $w(x, y)$, and calculations on large graphs with connections between every vertex is possible.

The second method is to create a sparse graph. A sparse graph can be created by only keeping the N largest values of $w(x, y)$ for each fixed x. Note that such a graph is not symmetric, but it can easily be made symmetric to aid in computation.

We list the two techniques to create the similarity function $w(x, y)$ used in this paper.

1. The Gaussian function
$$w(x, y) = \exp(-||x - y||^2/\tau) \tag{2.8}$$
is a common similarity function. Depending on the choice of metric, this similarity function includes the Yaroslavsky filter [55] and the the non-local means filter [9].
2. Zelnik-Manor and Perona introduced local scaling weights for sparse matrix computations [57]. They start with a metric $d(x_i, x_j)$ between each sample point. The idea is to define a local parameter $\sqrt{\tau(x_i)}$ for each x_i. The choice in [57] is $\sqrt{\tau(x_i)} = d(x_i, x_M)$ where x_M is the M^{th} closest vector to x_i. In [57], $M = 7$, while in this work and [46] $M = 10$. The similarity matrix is then defined as
$$w(x, y) = \exp\left(-\frac{d(x, y)^2}{\sqrt{\tau(x)\tau(y)}}\right). \tag{2.9}$$
This similarity matrix is better at segmentation when there are multiple scales that need to be segmented simultaneously.

2.3. Segmentation, spectral clustering, and the graph Laplacian. In this subsection we review some of the previous literature of spectral clustering - in which one directly uses the spectrum of the graph Laplacian to segment data. The goal of graph clustering is to partition the vertices into groups according to their similarities. Consider the weight function as a measure of the similarities, then the graph problem is equivalent to finding a partition of the vertices such that the sum of the edge weights between the groups are small compared with the sum of the edges within the groups. The weighted graph minimization algorithms in their original form are NP complete problems [51]; therefore a relaxed problem was formulated by Shi and Malik [43] where the minimization function is allowed to be real valued, and such minimization problems are equivalent to the spectral clustering methods.

The segmentation problem naturally generates a graph structure from a set of vertices v_i each of which is assigned a vector $z_i \in \mathbb{R}^K$. For example, when considering voting records of the US House of Representatives, each representative defines a vertex and their voting record defines a vector. A different example arises when considering similarity between regions in image data. Each pixel defines a vertex and one can assign a high dimensional vector to that pixel by comparing similarities between the neighborhood around that pixel and that of any other pixel. Given such an association, a symmetric weight matrix can be created using a symmetric function $\hat{w}(x,y) : \mathbb{R}^K \times \mathbb{R}^K \to \mathbb{R}_+$. In particular, if $\nu_i(y) = z_i$ represents the vector associated with the vertex ν_i, then the weight matrix $w(\nu_i, \mu_j) = \hat{w}(\nu_i(z), \mu_j(z)) = \hat{w}(z_i, z_j)$ is a positive symmetric function. We will abuse notation and not distinguish between these two functions and write $w(\nu_i, \mu_j) = \hat{w}(z_i, z_j) = w(z_i, z_j)$. Similar statements are true for any function $u : V \to \mathbb{R}$. Spectral clustering algorithms for binary segmentation consist of the following steps:

Input: A set of vertices V with the associated set of vectors $Z \subset R^K$, a similarity measure $w(x,y) : R^K \times R^K \to R_+$, and the integer k of clusters to construct.

1. Calculate the weight function $w(x,y)$ for all $x, y \in Z$.
2. Compute the graph Laplacian L.
3. Compute the second eigenvector ψ_2 of L *or* the second eigenvector ψ_2 of the generalized eigenvalue problem $L\psi = \lambda D\psi$.
4. Segment ψ_2 into two clusters using k-means (with $k = 2$).

Output: A partition of V (or equivalently Z) into two clusters A and $\overline{A}$.

Two characteristics of the spectral clustering algorithms should be highlighted. First, the algorithm determines clusters using a k-means algorithm. We note that the k-means algorithm is used to construct a partition of the real valued output, and any algorithm that performs this goal can be substituted for the k-means algorithm. For example, Lang [38] uses separating hyperplanes. A partitioning algorithm is needed since the relaxed problem does not force the final output function f to be binary valued. We address this problem by using the Ginzburg-Landau potential.

The second characteristic is that spectral clustering finds natural clusters through a constrained minimization problem. The constrained minimization problem exploits a finite number of eigenfunctions depending on the *a-priori* chosen number of clusters. A significant difference in our method is that we utilize *all* the eigenfunctions in our variational problem. One can interpret this as an issue of the number of scales that need to be resolved to perform the desired classification. For spectral clustering to work, the eigenfunctions used must capture all the relevant scales in the problem. By using all the eigenfunctions we resolve essentially all the scales in the problem, modulo the choice of ϵ. In the classical differential equation problem ϵ selects a smallest length scale to be resolved for the interfacial problem. An analogous role could occur in the graph problem and thus it would make sense to use this method on large dataset problems rather than relatively small problems, for which other methods might be simpler.

2.4. Proper normalization of the graph Laplacian with scale. An important issue with non-local operators are the behavior of the operators with increased sample size. Increasing sample size for the discrete Laplace operator corresponds to decreasing the grid size, and this operator must be correctly normalized in order to guarantee convergence to the differential Laplacian. We note that in the case of the classical finite difference problem for PDEs the entire matrix is multiplied by N^2 where N is the number of vertices in one of the dimensions, this is essentially $1/dx$, the spatial grid size. Recall that the largest eigenvalue of the operator scales like N^2 or $1/dx^2$, which gives a stiffness constraint for forward time stepping of the heat equation, as a function of grid size. Moreover, with this scaling, the graph Laplacian converges to the continuum differential operator in the limit of large sample size, i.e. as $N \to \infty$ where N is the grid resolution along one of the coordinate axes.

Proper normalization conditions for convergence also exist for the graph Laplacian. The issue of sample size also comes into play but rather than convergence to a differential operator, we consider the density of vertices, in the case of spatial embeddings, which can be measured by the degree of each vertex. The normalized Laplace operator as defined in (2.6) is known to have the correct scaling for spectral convergence of the operator in the limit of large sample size.

We make the following assumptions:

1. The set of k vectors $Z = \{z_i\}_{i=1}^N$ was sampled from a manifold in R^K,
2. each sample is drawn from an unknown distribution $\mu(z)$,
3. the graph Laplacian is a graph representation of the integrating kernel $w(x, y)$ with vertex set V, and
4. each vector in Z is assigned a vertex and weighted edges $w(x, y)$ between every $x, y \in Z$.

Consistency and practicality of the method requires similar and useful solutions as the number of samples increases [53, 51, 52]. Furthermore, the computational methods must be stable. The stability of the computational methods will be discussed first.

Note that the eigenvectors of the discrete Laplacian converge to the eigenvectors of the Laplacian, i.e. the discrete Fourier modes converge to the continuous Fourier modes. Similarly, it has been shown that the spectrum of the graph Laplacian converges (compactly) to the corresponding integral operator [52]. We note that, as stated in [51], there is a dilemma with the convergence for clustering applications. In summary, the unnormalized Laplacian converges to the operator L defined by $(Lu)(x) = d(x)u(x) - \int_\Omega w(x, y)u(y)dy$ while the normalized Laplacian converges to L_s defined by $(L_s u)(x) = u(x) - \int_\Omega (w(x, y)/\sqrt{d(x)d(y)})u(y)dy$. Both operators are a sum of two operators, a multiplication operator and the operator $w(x, y)$ or $w(x, y)/\sqrt{d(x)d(y)}$. The operators with kernels $w(x, y)$ and $w(x, y)/\sqrt{d(x)d(y)}$ are compact and thus have a countable spectrum. The operators $d(x)$ and the identity operator $\mathbf{1}$ are multiplication operators, but the operator $d(x)$ has an *a-priori* unknown value while the identity operator has an isolated eigenvalue. Note that the spectrum of a multiplication is the essential range of the operator $d(x)$; therefore, by perturbation theory results, the essential spectrum of L is the essential spectrum of $d(x)$ [53].

Perturbation theory does not imply anything about the convergence of the eigenvalues inside the essential spectrum of the operator L. Therefore, we do not know if

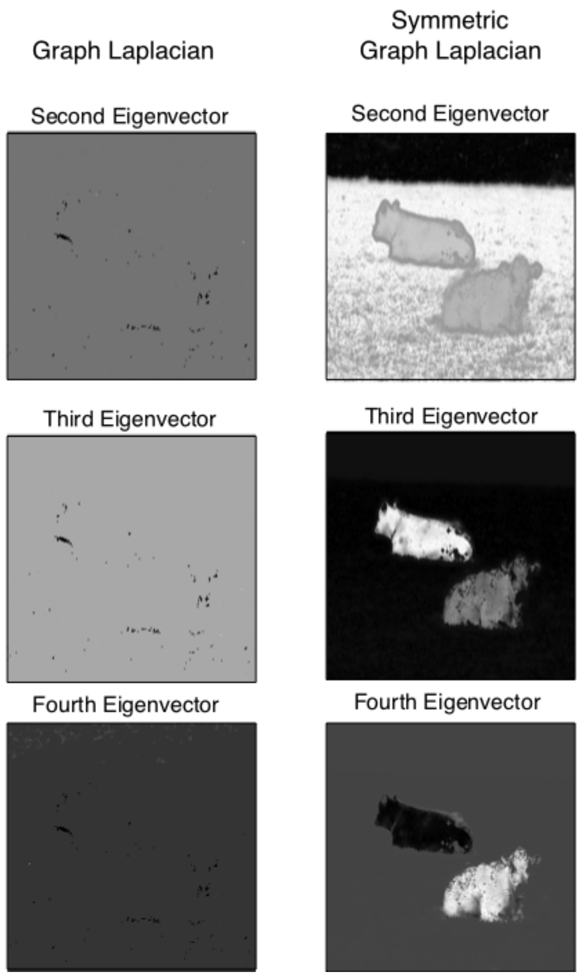

FIG. 2.1. *Eigenfunctions from the graph Laplacian obtained from the cow image in Section 4.3. The left three images are eigenvectors of the unnormalized Laplacian L as in (2.4). The right three images are eigenvectors of the symmetric graph Laplacian L_s as defined in (2.6).*

the function L is consistent if we increase the number of samples. This problem is avoided if the normalized Laplacian is used instead.

This normalization discussion is not pedantic, and the importance of correct normalization is shown in Figure 2.1. The right three images are example eigenvectors of the symmetric graph Laplacian L_s. Notice that the eigenvectors form reasonable segmentation of the images. For example, the second eigenvector distinguishes between the sky and cows, the third eigenvector separates the cows from the background, and the fourth eigenvector separates the two cows. The left three images are examples eigenvalues of the unnormalized Laplacian L. The spectrum of the unnormalized Laplacian (2.4) is dominated by large spikes at a few pixels. In contrast, the eigenfunctions on the normalized symmetric Laplacian (2.6) provide appealing segmentations of the image.

2.5. Semi-supervised Learning (SSL) on Graphs. There are numerous approaches to SSL using graph theory, and we mention a few that are related to this work. The work of Coifman, Szlam and others [15, 47] demonstrate techniques to learn classes using a diffusion framework. Their technique implements the Geometric Diffusion framework with a random walk probability interpretation. Instead of minimizing an energy functional, they find a time s when the marginal between known classes is maximized and then classify the rest of the samples using this diffusion time s. The final segmentation is dominated by the smallest eigenvalues of the random walk graph Laplacian. In contrast our method is based on an extension of a nonlinear geometric segmentation method applied to general graphs rather than lattices embedded in Euclidean space.

The work of Gilboa and Osher [28, 29] is another closely related technique, inspired in part by earlier work of Morel et al [9] for denoising. They use the graph Laplacian with an explicit forward time stepping scheme. The explicit time stepping introduces a stiffness constraint (discussed below) that slows the rate of convergence. Furthermore, their algorithm is stopped at an arbitrary stopping time while the technique proposed here has an automated stopping criteria.

In the paper [29], a nonlinear nonlocal TV based method is developed which has remarkable results for texture-based inpainting, although the computational time is not so fast. Our method is a different way of approaching this problem by using the GL functional instead of TV and by taking advantage of fast algorithms for the minimization problem by using the Nyström extension for the graph Laplacian.

Now we show how to use the Ginzburg-Landau energy on graphs in a semi-supervised learning application. Assume we have data organized in a graphical structure in which each graph node $z_i \in Z$ corresponds to a data vector x_i and the weights between the nodes are constructed using a method such as the ones described in Section 2.2. The goal is to perform a binary segmention of the data on the graph based on a known subset of nodes (possibly very small) which we denote by Z_{data}. We denote by λ the characteristic function of Z_{data} :

$$\lambda(z) = \begin{cases} 1 & \text{if } z \in Z_{data} \\ 0 & \text{otherwise.} \end{cases}$$

The graph segmentation problem automatically finds a decomposition of the vertices Z into disjoint sets $Z_{in} \cup Z_{out}$. These will be computed by assigning ± 1 to each of the nodes using a variational procedure of minimizing a Ginzburg-Landau functional. The known data involves a subset of nodes for which $+1$ or -1 is already assigned, and denoted by u_0 in the variational method.

The Ginzburg-Landau functional for SSL is

$$\boxed{E(u) = \frac{\epsilon}{2}\langle u, L_s u\rangle + \frac{1}{4\epsilon}\sum_{z\in Z}(u^2(z)-1)^2 + \sum_{z\in Z}\frac{\lambda(z)}{2}(u(z)-u_0(z))^2.} \tag{2.10}$$

The fidelity term uses a least-squares fit, allowing for a small amount of misclassification (i.e. noisy data) in the information supplied.

3. Computational algorithm.

In this section we go into greater detail regarding the numerical scheme used to find the minimizers of the variational problem. There are two main components to the algorithm - the choice of splitting schemes (1.1) and the computation of the basis functions as eigenfunctions of the graph Laplacian. We cover both below.

3.1. Convex splitting scheme. Our choice of splitting is motived by prior work on GL-type functionals for image processing with fidelity [18, 7, 6]. First we review the algorithm as it applies to differential operators in the classical Ginzburg-Landau regularization. An efficient convex splitting scheme can be derived by writing the Ginzburg-Landau energy with fidelity as

$$E(u) = E_1(u) - E_2(u)$$

with

$$E_1(u) = \frac{\epsilon}{2}\int |\nabla u(x)|^2 dx + \frac{c}{2}\int |u(x)|^2 dx, \tag{3.1}$$

$$E_2(u) = -\frac{1}{4\epsilon}\int (u(x)^2-1)^2 dx + \frac{c}{2}\int |u(x)|^2 dx - \int \frac{\lambda(x)}{2}(u(x)-u_0(x))^2 dx \tag{3.2}$$

Note that the energy E_2 is not strictly concave, but we can choose the constant c such that it is concave for u near and in between the potential wells of $(u^2-1)^2$. This scheme was chosen so the nonlinear term is in the explicit part of the splitting.

Given the above splitting and since the Fourier transform diagonalizes the Laplace operator, the following numerical scheme solves the Euler-Lagrange equations.

$$\begin{aligned}
a_k^{(n)} &= \int e^{ikx} u^{(n)}(x)\, dx \\
b_k^{(n)} &= \int e^{ikx} (u^{(n)})^3(x)\, dx \\
d_k^{(n)} &= \int e^{ikx} \lambda(x)(u^{(n)}(x) - u_0(x))\, dx \\
\mathcal{D}_k &= 1 + dt\ (\epsilon\ k^2 + c) \\
a_k^{(n+1)} &= \mathcal{D}_k^{-1}\left[\left(1 + \frac{dt}{\epsilon} + c\, dt\right)\ a_k^{(n)} - \frac{dt}{\epsilon} b_k^{(n)} - dt\ \left(d_k^{(n)}\right)\right].
\end{aligned}$$

Note that the H^1 semi norm is convex and thus appeared in the convex part of the energy splitting. The first variation of that yields the Laplace operator which is a stiff operator to have in an evolution equation. The stiffness results because the eigenvalues of the Laplace operator range from order one negative values to minus infinity. Or in the case of a discrete approximation of the Laplace operator, the eigenvalues range from order one to minus one divided by the square of the smallest length scale of resolution (e.g. the spatial grid size in a finite element or finite difference approximation). By projecting onto the eigenfunctions of the Laplacian, we see that there are many different timescales of decay in the spatial operator and all must be resolved numerically in the case of a forward time stepping scheme. However when the Laplace operator is evaluated implicitly, at the new time level, one need not resolve the fastest timescales in the time-stepping scheme.

The same time-stepping scheme can be used if the spectral decomposition of the graph Laplacian is used instead of the Laplacian, and we can use the spectral decomposition for any of the graph Laplacians L, L_w, or L_s. We used the spectrum of L_s due to the convergence and scaling issues discussed above. Here is a summary of the method as used in this paper:

Decompose the solution $u^{(n)}$ at each time step according to the known eigenvectors $\{\phi_k(x)\}$ of L_s:

$$u^{(n)}(x) = \sum_k a_k^{(n)} \phi_k(x).$$

Likewise we need to decompose the pointwise cube of u and the fidelity term,

$$[u^{(n)}(x)]^3 = \sum_k b_k^{(n)} \phi_k(x),$$
$$\lambda(x)\left(u^{(n)}(x) - u_0(x)\right) = \sum_k d_k^{(n)}\ \phi_k(x).$$

Then the algorithm for the next iteration is given in terms of the coefficients for

$$u^{(n+1)}(x) = \sum_k a_k^{(n+1)} \phi_k(x)$$

in terms of its decomposition using the eigenfunctions of L_s again as a basis for the solution. Define $\tilde{\lambda}_k$ to be the eigenvalue associated with the eigenfunction $\phi_k(x)$, i.e. $L_s\phi_k = \tilde{\lambda}_k\phi_k$; then the update equation for $a_k^{(n)}$ is

$$\begin{aligned} \mathcal{D}_k &= 1 + dt\ (\epsilon\ \tilde{\lambda}_k + c), \\ a_k^{(n+1)} &= \mathcal{D}_k^{-1}\left[\left(1 + \frac{dt}{\epsilon} + c\,dt\right)\ a_k^{(n)} - \frac{dt}{\epsilon} b_k^{(n)} - dt\ \left(d_k^{(n)}\right)\right]. \end{aligned} \tag{3.3}$$

```
Convex Splitting for the Graph Laplacian

 1. Input ← an initial function $u_0$ and the eigenvalue-eigenvector pairs
    $(\tilde{\lambda}_k, \phi_k(x))$ for the graph Laplacian $L_s$ from Equation (2.6).
 2. Set convexity parameter $c$ and interface scale $\epsilon$ from Equation (3.2).
 3. Set the time step $dt$.
 4. Initialize $a_k^{(0)} = \int u(x)\phi_k(x)\,dx$.
 5. Initialize $b_k^{(0)} = \int [u_0(x)]^3\phi_k(x)\,dx$.
 6. Initialize $d_k^{(0)} = 0$.
 7. Calculate $\mathcal{D}_k = 1 + dt\ (\epsilon\ \tilde{\lambda}_k + c)$.
 8. For $n$ less than a set number of iterations $M$
      (a) $a_k^{(n+1)} = \mathcal{D}_k^{-1}\left[\left(1 + \frac{dt}{\epsilon} + c\,dt\right)\ a_k^{(n)} - \frac{dt}{\epsilon} b_k^{(n)} - dt d_k^{(n)}\right]$
      (b) $u^{(n+1)}(x) = \sum_k a_k^{(n+1)}\phi_k(x)$
      (c) $b_k^{(n+1)} = \int [u^{(n+1)}(x)]^3\phi_k(x)\,dx$
      (d) $d_k^{(n+1)} = \int \lambda(x)\left(u^{(n+1)}(x) - u_0(x)\right)\phi_k(x)\,dx$
 9. end for
10. Output ← the function $u^{(M)}(x)$.
```

This is a generalization of a classical 'psuedospectral' scheme for PDEs in which one goes back and forth between the spectral domain (the coefficients $a_i^{(n)}$) and the graph domain in which we evaluate u directly at every vertex on the graph. The latter must be done in order to compute the cube $[u^{(n)}(x)]^3$ and the fidelity term $\lambda(x)\left(u^{(n)}(x) - u_0(x)\right)$ which can then be projected back to the spectral domain. Here the convex temporal splitting is very important because it effectively removes the stiffness inherent in the diverse time scales that arise from the range of eigenvalues of the graph Laplacian. Our proposed method is only useful if one has a fast method for determining the eigenfunctions $\phi_k(x)$ and their corresponding eigenvalues. For the case of fully connected graphs we use the Nyström extension reviewed in the next subsection.

3.2. Nyström extension for fully connected graphs. The spectral decomposition of the matrix L_s is related to the spectral decomposition

$$D^{-1/2}WD^{-1/2}\phi = \xi\phi$$

through the relationship

$$\begin{aligned} L_s\phi &= (1 - D^{-1/2}WD^{-1/2})\phi \\ &= (1-\xi)\phi = \tilde{\lambda}\phi. \end{aligned}$$

Therefore, the convex splitting scheme is efficient if the spectral decomposition of the matrix $D^{-1/2}WD^{-1/2}$ can be quickly found. The matrix W, however, is a large matrix and it cannot be assumed that the matrix will be sparse. We use the Nyström extension discussed by Fowlkes *et al.* [26, 5, 25] to address this issue.

The Nyström method is a technique to perform matrix completion that has been used in a variety of image processing applications including spectral clustering [41], kernel principle component analysis [19], and fast Gaussian process calculations. Below we review the Nyström method as used in this paper. Although the method is well-known in the graph theory community, we include a summary of the ideas here for the benefit of readers not familiar with these techniques (including the PDE community who may be interested in extending these ideas to general graph problems).

The Nyström method approximates the eigenvalue equation

$$\int_\Omega w(y,x)\phi(x)\;dx = \gamma\phi(y)$$

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

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

where $E(x)$ is the error in the approximation. Our model, however, does not allow us to choose the interpolation points, but rather the interpolation points are randomly samples from some sample space.

Recall that $Z = \{z_i\}_{i=1}^N$ is the set of nodes on the graph; it also defines an $N-$dimensional vector space with W as a linear operator on that space. In this work, the Nyström method is used to approximate the eigenvalues of the matrix W with components $w(z_i, z_j)$. A key idea used to produce a fast algorithm is to choose a randomly sampled subset $X = \{x_i\}_{i=1}^L$ of the entire graph Z to use as interpolation points, and the interpolation weights are the values of the weight function $a_j(y) = w(y, x_j)$.

Partition Z into two sets X and Y with $Z = X \cup Y$ and $X \cap Y = \emptyset$. Furthermore, create the set X by randomly sampling L points from Z. Let $\phi_k(x)$ be the k^{th} eigenvector for W. The Nyström extension approximates the value of $\phi_k(y_i)$, up to a scaling factor, using the system of equations

$$\sum_{x_j\in X} w(y_i,x_j)\phi_k(x_j) = \gamma\phi_k(y_i). \tag{3.4}$$

This equation cannot be calculated directly since the eigenvectors $\phi_k(x)$ are not initially known. This problem is overcome by first approximating the eigenvectors of W

with the eigenvectors of a sub-matrix of W. These approximate eigenvalues, however, may not be orthogonal. The approximate eigenvectors will then be orthogonalized, and this final set of eigenvectors will serve as an approximation to the eigenvectors of the complete matrix W. Note that since only a subset of the matrix W is initially used, only a subset of the eigenvectors can be approximated using this method.

The notation

$$W_{XY} = \begin{bmatrix} w(x_1, y_1) & \dots & w(x_1, y_{N-L}) \\ \vdots & \ddots & \vdots \\ w(x_L, y_1) & \dots & w(x_L, y_{N-L}) \end{bmatrix} \tag{3.5}$$

will be used in this section. Similarly, define the matrices W_{XX} and W_{YY} and the vectors ϕ_X and ϕ_Y. The matrix $W \in \mathbb{R}^K \times \mathbb{R}^K$ and vectors $\phi \in \mathbb{R}^K$ can be written as

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

and $\phi = \begin{bmatrix} \phi_X^T & \phi_Y^T \end{bmatrix}^T$ with ϕ^T denoting the transpose operation.

The spectral decomposition of W_{XX} is $W_{XX} = B_X \Gamma B_X^T$ where B_X is the eigenvector matrix of W_{XX} with each column an eigenvector and $\Gamma = diag(\gamma_1, \gamma_2, \dots, \gamma_L)$ are the corresponding eigenvalues. The Nyström extension of Equation 3.4 in matrix form using the interpolation points X is

$$B_Y = W_{YX} B_X \Gamma^{-1}. \tag{3.6}$$

In short, the n eigenvectors of W are approximated by $B = \begin{bmatrix} B_X^T & (W_{YX} B_X \Gamma^{-1})^T \end{bmatrix}^T$. The associated approximation of $W = B \Gamma B^T$ is

$$W = \begin{bmatrix} W_{XX} & W_{XY} \\ W_{YX} & W_{YX} W_{XX}^{-1} W_{XY} \end{bmatrix}.$$

From this equation, it can be shown that the large matrix W_{YY} is approximated by

$$W_{YY} \approx W_{YX} W_{XX}^{-1} W_{XY}.$$

As mentioned in [26], the quality of the approximation to W_{YY} is given by the norm $||W_{YY} - W_{YX} W_{XX}^{-1} W_{XY}||$, and this is determined by how well W_{YY} is spanned by the columns of W_{XY}.

This decomposition is unsatisfactory since the approximate eigenvectors $\phi_i(x)$ defined above are not orthogonal. This deficiency can be fixed using the following trick. For arbitrary unitary A and diagonal matrix Ξ then if

$$\Phi = \begin{bmatrix} W_{XX} \\ W_{YX} \end{bmatrix} (B_X \Gamma^{-1/2} B_X^T)(A \Xi^{-1/2}) \tag{3.7}$$

the matrix W can be written as $W = \Phi \Xi \Phi^T$. We are therefore free to choose A unitary such that $\Phi^T \Phi = \mathbf{1}$. If such a matrix A can be found, then the matrix W will be diagonalized using the unitary matrix Φ. Define the operator $Y = A \Xi^{-1/2}$, then the proper choice of A is given through the relationship

$$\Phi^T \Phi = (Y^T)^{-1} W_{XX} Y^{-1} + (Y^T)^{-1} W_{XX}^{-1/2} W_{XY} W_{YX} W_{XX}^{-1/2} Y^{-1}.$$

If $\Phi^T\Phi = \mathbf{1}$ then after multiplying the last equation on the right by $\Xi^{1/2}A$ and on the left by the transpose we have

$$A^T \Xi A = W_{XX} + W_{XX}^{-1/2} W_{XY} W_{YX} W_{XX}^{-1/2}. \tag{3.8}$$

Therefore, if the matrix $W_{XX} + W_{XX}^{-1/2} W_{XY} W_{YX} W_{XX}^{-1/2}$ is diagonalized, then its spectral decomposition can be used to find an approximate orthogonal decomposition of W with eigenvectors Φ given by Equation 3.7.

The matrix W must also be normalized in order to use L_s for segmentation. Normalization of the matrix is a straightforward application of Equation 3.7. In particular, let $\mathbf{1}_K$ be the K dimensional unit vector, then define $[d_X^T\, d_Y^T]^T$ as

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

Let $A./B$ denote component-wise division between two matrices A and B and $x\,y^T$ the outer product of two vectors, then the matrices W_{XX} and W_{XY} can be normalized by

$$\begin{aligned}\overline{W}_{XX} &= W_{XX}./(s_X s_X^T), \\ \overline{W}_{XY} &= W_{XY}./(s_X s_Y^T),\end{aligned} \tag{3.9}$$

where $s_X = \sqrt{d_X}$ and $s_Y = \sqrt{d_Y}$.

The Nyström extension can be summarized by the following pseudo code.

```
Nyström Extension for Symmetric Graph Laplacian
 1. Input ← a set of features $Z = \{x_i\}_{i=1}^N$
 2. Partition the set $Z$ into $Z = X \cup Y$ where $X$ consists of $L$ randomly
    selected elements.
 3. Calculate $W_{XX}$ and $W_{XY}$ using Equation 3.5.
 4. $d_X = W_{XX}\mathbf{1}_L + W_{XY}\mathbf{1}_{N-L}$.
 5. $d_Y = W_{YX}\mathbf{1}_L + (W_{YX}W_{XX}^{-1}W_{XY})\,\mathbf{1}_{N-L}$.
 6. $s_X = \sqrt{d_X}$ and $s_Y = \sqrt{d_Y}$.
 7. $W_{XX} = W_{XX}./(s_X s_X^T)$.
 8. $W_{XY} = W_{XY}./(s_X s_Y^T)$.
 9. $B_X \Gamma B_X^T = W_{XX}$ (using the SVD).
10. $S = B_X \Gamma^{-1/2} B_X^T$.
11. $Q = W_{XX} + S(W_{XY}W_{YX})S$.
12. $A \Xi A^T = Q$ (using the SVD).
13. $\Phi = \begin{bmatrix} B_X\Gamma^{1/2} \\ W_{YX}B_X\Gamma^{-1/2} \end{bmatrix} B_X^T(A\Xi^{-1/2})$ diagonalizes $W$.
14. Output ← the $L$ eigenvalue-eigenvector pairs $(\phi_i, \tilde{\lambda}_i)$ where $\phi_i$ is the $i^{th}$
    column of $\Phi$ and $\tilde{\lambda}_i = 1 - \xi_i$ with $\xi_i$ the $i^{th}$ diagonal element of $\Xi$.
```

4. Classification on graphs.

The Ginzburg-Landau energy functional can be used for unsupervised and semi-supervised classification learning on graphs. This section gives examples of three classifications problems. In particular, we investigate the house voting records of 1984 from the UCI machine learning database [27], the Two Moons example dataset of Bühler and von Luxberg [10], and an image segmentation problem.

Each of the classification examples follows the same general procedure. Given a set of vertices $V = \{\nu_i\}_{i=1}^N$, the general procedure consists of the following SSL steps:

1. **Determine Features:** For each vertex ν_i, determine a feature vector z_i.
2. **Build Graph:** Determine edge weights using either formula (2.8) or (2.9) and build an undirected graph based on these weights.
3. **Initialization:** Initialize a function $u(z_i)$ based on any *a-priori* knowledge.
4. **Minimization:** Minimize the Ginzburg-Landau energy functional with appropriate constraints and fidelity term(s). Note that for all experiments we use the normalized Laplacian L_s.
5. **Segmentation:** Segment the vertices into two classes according to $f(z_i) = sgn(u(z_i))$.

Each of the vertices represent the objects that we want to segment, and the feature vectors provide distinguishing characteristics between the objects.

4.1. House voting records from 1984. The US House or Representatives voting records data set consists of 435 individuals where each individual represents a vertex of the graph. The goal is to use SSL to segment the data into the two party affiliation Democrat or Republican. The SSL algorithm was performed by assuming a party affiliation of five individuals, two Democrats and Three Republicans, and segmenting the rest. The votes were taken in 1984 from the 98th United States Congress, 2nd session.

A 16 dimensional feature vector was created using 16 votes recorded for each individual in the following manner. A yes vote was set to one, while a no vote was set to negative one. The voting records had a third category, a did not vote category. Each did not vote recording was represented by a zero in the feature vector. A fully weighted graph was then created using Gaussian similarity function (2.8) with $\tau = .3$.

The function $u(z)$ was initialized to one for the two Democrats, negative one for the three Republicans, and zero for the rest of the classes. The Ginzburg-Landau function with fidelity, equation (2.10), was then minimized using the convex splitting algorithm with parameters $c = 1$, $dt = 0.1$, $\epsilon = 2$, and 500 iterations. In the fidelity terms, we chose $\lambda = 1$ for each of the five known individuals and $\lambda = 0$ for the rest. This segmentation yielded 95.13% correct results. Note that due to the small size of this graph we did not use the Nyström extension to compute the spectrum.

The probability of the party affiliation given the vote was above 90% for some of the votes. We investigated the accuracy of the segmentation when these votes were removed. Figure 4.1 shows the accuracy of the method when the 14, 12, 10, and 8 least predictive votes were used for the analysis, and we obtained 91.42%, 90.95%, 85.92%, and 77.49% respective accuracy.

The work of Ratanamahatana and Gunopulos [42] studies this dataset using a naive Bayesian decision tree method. They obtained 96.6% classification accuracy using 80% of the data for training and 20% for classification. In contrast, our method uses only 1.15% of the data (5 samples out of 435) to obtain a classification accuracy of 95.13%. The work of Gionis et al. [30] uses clustering aggregation to automatically determine the number of classes and class membership. Their method obtains 89% correct classification in contrast to our 95.13%, in which we specify two clusters.

4.2. Two moons. The two moon dataset was used by Bühler *et al.* [10] and Szlam *et al.* [45, 46] in connection with spectral clustering using the p-Laplacian. This

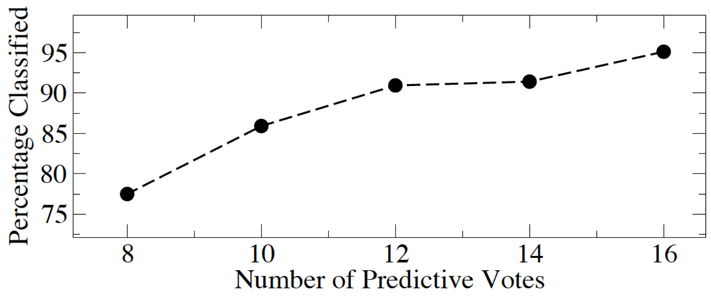

FIG. 4.1. *The error rate of segmenting the house votes. We tested the accuracy of the segmentation when the most predictive votes were removed. The segmentation procedure was reproduced where we removed the top two, four, six, and eight most predictive votes to investigate the robustness of the algorithm.*

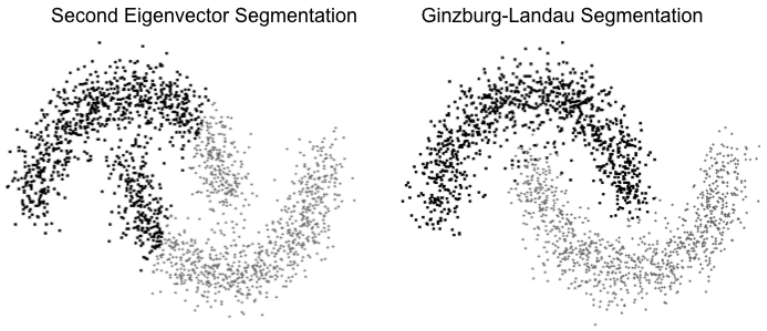

FIG. 4.2. *The left hand figure is the segmentation achieved by thresholding the second eigenvector of the graph Laplacian. The right hand image is the segmentation obtained from the algorithm presented in this paper. This algorithm segmented this data set with an error rate of 2.1%.*

data set is constructed using two half circles in R^2 with radius one. The first half circle is contained in the upper half plane with a center at the origin, while the second half circle is created by taking the half circle in the lower half plane and shifting it to $(1, .5)$. The two half circles are then embedded in R^{100}. Two thousand data points are sampled from the circles and independent and identically distributed Gaussian noise with standard deviation .02 is added to each of the 100 dimensions. The goal is to segment the two half circles using unsupervised segmentation. The unsupervised segmentation is accomplished by adding a mean zero constraint to the variational problem.

In order to make quantitative comparisons, we build the graph following the procedure described in Szlam and Bresson [45, 46] and Bühler [10]. They created a 10 nearest neighbor weighted graph from the data using the self-tuning weights of Zelnik-Manor and Perona [57] discussed in Section 2.2 where M was set to 10. This is a difficult segmentation problem since the embedding and noise creates a complex graphical structure.

We initialize the function $u(z)$ using the second eigenfunction of the Laplacian. More specifically, we set $u(z) = sgn(\phi_2(z) - \overline{\phi_2})$ where $\phi_2(z)$ is the second eigenfunction and $\overline{\phi_2}$ is the mean of the second eigenfunction. This initialization was used since the second eigenfunction approximates the relaxed graph cut solution (see Sec-

ϵ	c	no. iterations
10	0.2	500
2	1	200
1.5	1.33	200
1	2	200

TABLE 4.1
Table of parameter values for the GL functional for the two moons segmentation. The parameter $dt = 0.1$ was used throughout along with the formula (2.9) to construct the weighted graph.

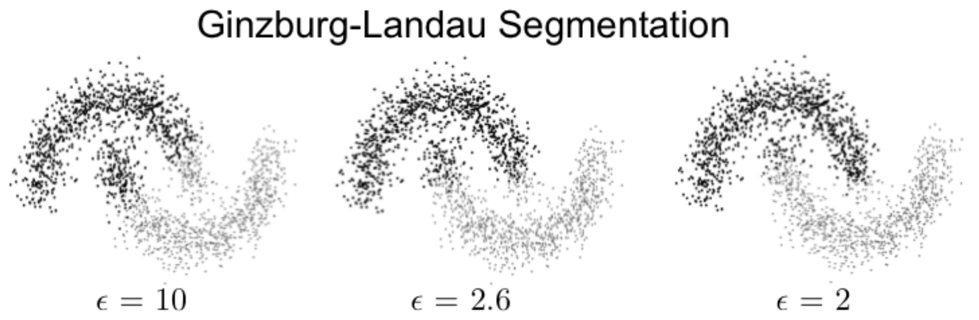

FIG. 4.3. *The parameter ϵ determines a scale for the Ginzburg-Landau energy functional. A more accurate segmentation is obtained as the ϵ scale parameter decreases. The percentage of correct classification was 82.85%, 90.75%, and 94.55% for $\epsilon = 10, 2.6,$ and 2 respectively.*

tion 5.1) We minimize the Ginzburg-Landau energy (2.10) with the mean constraint $\int u(x)dx = 0$, but without any fidelity terms. The Nyström extension is ineffective for sparse graphs. Instead, we used the first 20 eigenvectors using Matlab's sparse matrix algorithms.

Figure 4.2 compares the classical spectral clustering method with our method. Parameters for the Ginzburg-Landau minimization problem are shown in Table 4.1 and its caption. The left hand figure is the segmentation achieved by thresholding the eigenvector of the two moons data set. Clearly, spectral clustering using the second eigenvector of the Laplacian does not segment the two half moons accurately. The right hand image is the segmentation obtained from the algorithm presented in this paper. This algorithm segmented this data set with an error rate of 2.1%.

Reducing the Ginzburg-Landau energy parameter ϵ raises the potential barrier between the two states in the Ginzburg-Landau potential function and reduces the effect of the graph weights. Reducing ϵ corresponds to reducing the scale of the graph and allows for a sharper transition between the two states. The change in scale is shown in figure 4.3 where better segmentation was achieved with reduced ϵ. The $\epsilon = 10$ case is essentially the spectral clustering solution, while the $\epsilon = 2$ case closely resembles the 1-Laplacian solution of Bühler and Hein [10]. A high quality segmentation in which 94.55% of the samples were classified correct occurs when the parameter ϵ was set to two. This is contrasted with the second eigenvector spectral clustering technique that obtained 82.85% correct classification, essentially equivalent to the large $\epsilon = 10$ case.

Better segmentation can be achieved if the algorithm is repeated while reducing ϵ using the last segmentation as the initialization. The method of successive reductions in ϵ was used for image inpainting via the Cahn-Hilliard equation [6,

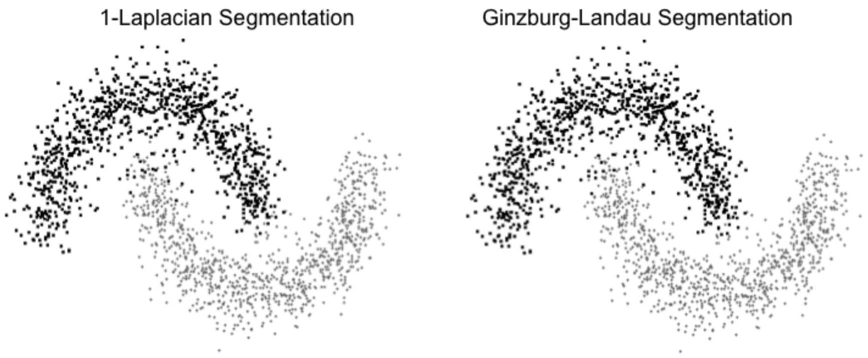

FIG. 4.4. *The 1-Laplacian and the Ginzburg-Landau clustering methods obtains nearly identical results for unsupervised clustering on the Half Moon dataset. The 1-Laplacian and Ginzburg-Landau percentage error was 97.3% and 97.9% respectively.*

7]. In [6] the authors carefully study the space of steady states for a stripe inpainting example in which the problem exhibits an incomplete supercritical pitchfork birfurcation as the scale parameter ϵ is varied. Different methods of reducing ϵ could lead to different local minima of the energy functional, along the two stable branches of the pitchfork. Such a detailed study is beyond the scope of this paper, however we can examine a segmentation, shown in Figure 4.4, where ϵ is reduced from 2 to 1 in steps of .5. The final segmentation gives 97.7% accuracy. We compared this segmentation to the 1-Laplacian Inverse Power Method (IPM) of Bühler and Hein [33]. (The code is freely obtainable from www.ml.uni-saarland.de/code/oneSpectralClustering/oneSpectralClustering.html.) The Normalized 1-Laplacian algorithm of Bühler and Hein with 10 initializations and 5 inner loops was used to obtain 97.3% accuracy for this data set. The computational time and accuracy of the 1-Laplacian method and the Ginzburg-Landau technique is shown in figure 4.5. The Ginzburg-Landau technique of this paper was able to obtain more accurate results in less computational time. No additional effort was made in our numerical tests to reduce run time - for example the large number of iterations may not be necessary with an adaptive dt or a better initialization. Speedup in other problems can easily be an order of magnitude when making such adjustments. Even so the run time is very fast.

4.3. Image labeling. The objective of image segmentation is to partition an image into multiple components where each component is a set of pixels. Furthermore, each component represents a certain object or class of objects. We are interested in the binary segmentation problem where each pixel can belong to one of two classes.

Most image segmentation algorithms assume that a segmented region is connected in the image. We need not make this assumption. Instead, we build a graph based on feature vectors derived from a neighborhood of each pixel, and segment the image based on a partition of the graph. The graph based segmentation allows us to label the unknown content of one image based on the known content of another image. As input to our segmentation algorithm we take two images where one of the images has been hand segmented into two classes. The goal is to automatically segment the second image based on the segmentation of the first image.

Each pixel y represents a vertex of the graph. The features vectors associated to each vertex y is defined using a pixel neighborhood $N(y)$ around y. For example, a

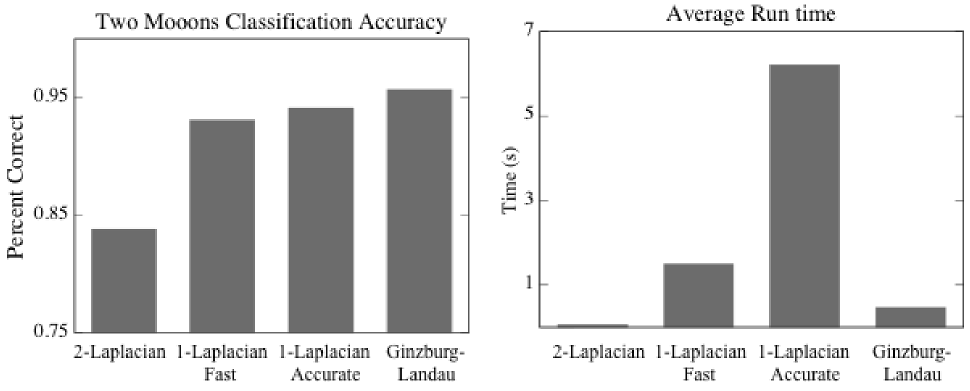

FIG. 4.5. *The accuracy of the Ginzburg-Landau unsupervised segmentation procedure presented in this paper is compared with spectral clustering and the 1-Laplacian code of Hein and Bühler [33]. These two graphs were created using 100 runs of the two moon data set and averaging the results.*

typical choice for a pixel neighborhood on a Cartesian grid $\Omega = Z^2$ is the set

$$N(y) = \{z \in \Omega : |z_1 - y_1| + |z_2 - y_2| \leq M\},$$

for some integer M. A feature vector derived from a finite sized neighborhood of a pixel is called a pixel neighborhood feature.

Let I be an image, then an example of a pixel neighborhood feature is the set of image pixel values $z(y) = I(N(y))$ chosen in a consistent order. Another example is to calculate a collection of filter responses for each pixel, i.e. $z(x) = ((g_1 * I)(x), (g_2 * I)(x), \ldots, (g_j * I)(x))$ where g_i represents a filter for each i, and $*$ is the convolution operator. The proper choice of neighborhood and features are application dependent. Note that a neighborhood system is equivalent to an edge system in graph theory [13], but the neighborhood system used to determine pixel neighborhood features is *not* the same as the graph used to generate the graph Laplacian.

A fully connected graph is generated using the pixels from two images as vetices and the weight matrix $w(x, y)$ for edge weights. This graph construction is very general and can be used to segment many different types of objects based on their determining features. For example, color and texture features are appropriate for segmenting trees and grass from other objects. We also note that the metric used in the weight matrix can be modified depending on the data set. For example, the spectral angle may be more appropriate for segmentation of hyperspectral images.

We demonstrate the image labeling technique using cow images from the Microsoft image database[2], and on face segmentation. The feature vectors used for the Microsoft image database and the face segmentation were respectively the Varma-Zisserman MR8 texture features [49] and the Weijer-Schmid Hue features [48]. The MR8 texture feature are robust to rotation and are translational invariant. The Hue features are invariant to photometric transformations.

On the hand labeled image, the function $u(z)$ was initialized to one for one of the classes and negative one for the other class. The function $u(z)$ was initialized to zero on the unlabeled image. The graph Laplacian is constructed using (2.8). The Ginzburg-Landau energy with fidelity was then minimized. The parameter values were as follows: $\tau = .1$, $dt = .01$, $\epsilon = .1$, $c = 21$, and 500 iterations. The fidelity term Λ was set to one on the initial image and to zero on the unknown image. The

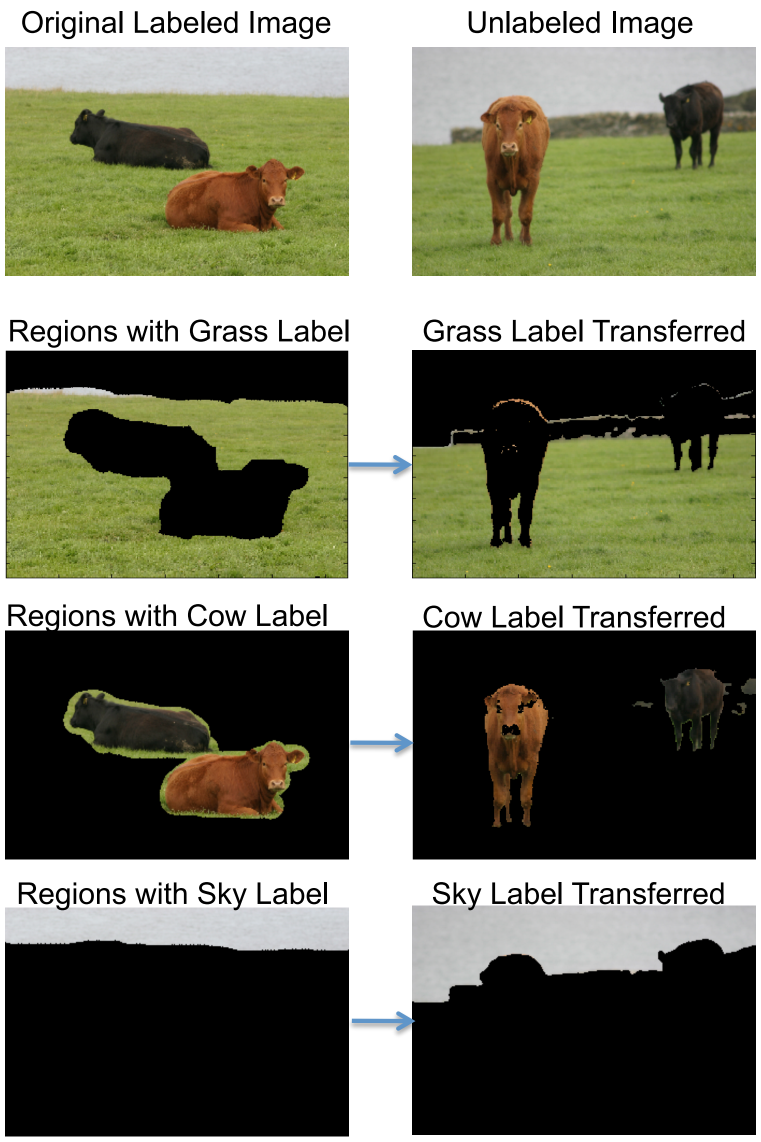

FIG. 4.6. *The labels from the original upper left hand image was transferred to the upper right hand image. The individual results for each region is shown in the lower images. Note that the algorithm is robust to mislabeled sections. Furthermore, the algorithm can identify regions that we do not know a label for such as the wall in the right hand images.*

Nyström extension was used to determine the spectral decomposition of the weight matrix. The labels of the second image was then determined by the sign of $u(z)$ on the second image. Results of the segmentation for the Microsoft database is shown in Figure 4.6. Note that the algorithm is robust to mislabeling in the initial image. Additional examples are shown in Figure 4.7. This figure demonstrates that cows of another color are not labeled as a cow. Note that the white is predominately classified as not a black or brown cow. Furthermore, the algorithm can handle more complex scenes where there are no cows. In the cluttered airplane scene, no consolidated

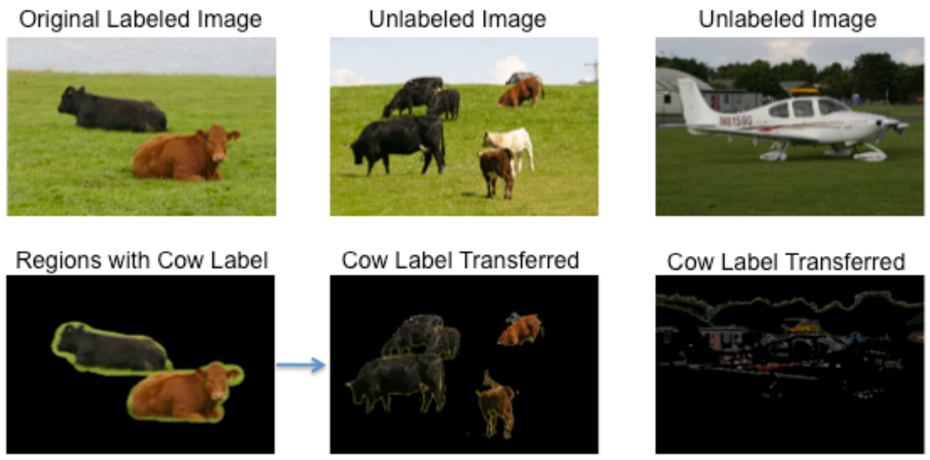

FIG. 4.7. *This figure uses the same labeled image as in Figure 4.6 and tries to classify similar features in additional images. We include one image with a white colored cow and another image with clutter and no cows.*

features are identified.

Another example is given in Figure 4.8 where a face from the Computational Vision database [1] was segmented. In both examples, the predominant features were identified, and some of the pixels with few representative features were removed. For example, the nose and eye of the brown cow were removed from the segmentation and the eyes and eyebrows of the face was removed. In some other notable image segmentation algorithms a postprocessing filter is often needed. The results shown here use no postprocessing filters.

5. Connection to previous methods in the literature.

We discuss the connection between our algorithm and related methods from the literature. In the final section we present some open problems.

5.1. Graph cuts.

Spectral clustering and graph segmentation methods are related through the *graph cut* objective function. For two disjoint subsets $A, B, \subset V$, define the graph cut of two sets as

$$cut(A,B) = \sum_{x\in A, y\in B} w(x,y).$$

The function $cut(A,B)$ is smaller when there are less weights on the connections between the sets A and B.

The mincut problem involves finding a partition $\{A, \overline{A}\}$ of V that minimizes $cut(A, \overline{A})$, where $\overline{A}$ is the complement of A. Stoer and Wagner [44] have an efficient algorithm for this problem. The mincut problem, however, leads to poor classification performance for many problems since isolated points often form one cluster and the rest of the points form another cluster. Modifications to the min-cut problem include a normalization of either $|A|$ or $vol(A)$ as a measure of the size. This procedure leads

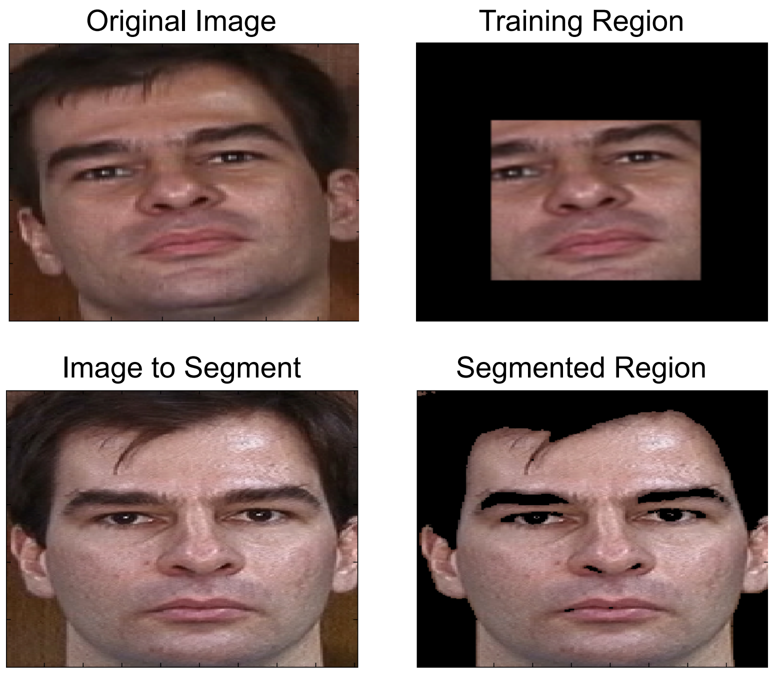

Fig. 4.8. *The robustness of the algorithm to lighting conditions and changes in texture is shown. The upper left hand image is the original image with the upper right hand the training region. The lower left hand image was segmented using the training region in the upper right hand image, and the segmentation is shown in the lower right hand image.*

to minimization of one of the following [51]

$$RatioCut(A, \overline{A}) = \frac{cut(A, \overline{A})}{|A|} + \frac{cut(A, \overline{A})}{|\overline{A}|},$$

$$Ncut(A, \overline{A}) = \frac{cut(A, \overline{A})}{vol(A)} + \frac{cut(A, \overline{A})}{vol(\overline{A})}.$$

The sum is minimized when either $|A|$ or $vol(A)$ is the same as $|\overline{A}|$ or $vol(\overline{A})$ respectively. In other words, the number of vertices or sum of the edge weights must be close to the same in each partition. This balance turns the mincut problem into an NP hard problem [54]. Spectral clustering techniques relaxes the balancing conditions to approximately solve a simpler version of the mincut problem.

The relaxed minimization of the *RatioCut* is

$$\min_{A \subset Y} \langle u, Lu \rangle \text{ such that } u \perp \mathbf{1} \text{ and } ||u||^2 = |Y|.$$

The relaxed problem is a norm minimization with two constraints, an L^2 constraint and a subspace constraint.

Similar to the *RatioCut* segmentation procedure the relaxed *NCut* problem is

$$\min_{A \subset Y} \langle u, L_s u \rangle \text{ such that } u \perp D^{1/2}\mathbf{1}, \text{ and } ||u||^2 = vol(Y).$$

This minimization problem is in the form of the Raleigh-Ritz theorem, and the solution is again given by the second eigenvector of L_s. We emphasize that the difference between the relaxed problem and the true graph cut solution is that the relaxed problem determines a real valued solution while the graph cut problem finds a binary solution. The relaxation from the discrete problem to the real valued problem does not always yield an approximation to the *Ncut* or *RatioCut* problem even for the binary segmentation problem. See for example [32]. The relaxed problem has been used for many segmentation problems and it produces appealing results [43].

Minimizing the Ginzburg-Landau energy functional with the mass constraint $u \perp \mathbf{1}$ produces a graph cut problem that is different than the other spectral clustering methods and it reintroduces a nearly binary valued solution. Recall that the Ginzburg-Landau potential term encourages the variational solution u to take values ± 1. Assume that these are the only two values for the variational solution, then the normalized graph Laplacian is

$$\langle u, L_s u\rangle = C + 4 \sum_{x\in \overline{A}\, y\in A} \frac{w(x,y)}{\sqrt{d(x)d(y)}} - 2\left(\sum_{x\in A,\, y\in A} \frac{w(x,y)}{\sqrt{d(x)d(y)}} + \sum_{x\in \overline{A},\, y\in \overline{A}} \frac{w(x,y)}{\sqrt{d(x)d(y)}} \right),$$

where C is a constant that depends on the graph but not the partition. Note that a mass constraint (or a fidelity constraint) will prevent the trivial solution with every element in a single set. It is clear from this representation that the graph cut is minimized when the normalized weights between the partitions is small, while the normalized weights within the partition remains large.

The graph p-Laplacian is a generalization of the graph Laplacian due to Ambhibech [4]. The graph p-Laplacian is the operator L_p that satisfies the equation

$$\langle u, L_p u\rangle = \frac{1}{2}\sum_{i,j=1}^{N} w_{ij}|u_i - u_j|^p.$$

Spectral clustering was accomplished by Bühler and Hein using the graph p-Laplacian [10]. They defined the eigenvectors of the graph p-Laplacian using the Rayleigh-Ritz principle where the functional to be minimized is

$$F_p(u) = \frac{\langle u, L_p u\rangle}{||u||_p^p}.$$

The work of Szlam and Bresson [45, 46] demonstrated that the solution of the relaxed version of the 1-Laplacian is identical to the unrelaxed version. They then derived a Split-Bregman algorithm to find an approximate solution to the 1-Laplacian. Another approximation method was derived by Bühler and Hein to solve the 1-Laplacian [33]. Their algorithm was used in this work to compare to the Ginzburg-Landau segmentation procedure.

The work by Zhu, Gharhramani, and Lafferty [58] is another early publication in this area. Their work connects elements of graph theory, Gaussian random fields, and random walks to provide an outline for supervised learning algorithms using the 2-Laplacian. Their approach constrains u to be fixed at some of the nodes, but they do not use the double well potential or a fidelity term as is done in this work.

5.2. Non-local means. Buades, Coll, and Morel described the following non-local filtering, non-local means (NLM), procedure for noise removal in images. Define the non-local operator

$$NL(u)(x) = \frac{1}{d(x)}\int_{\Omega} w(x,y)u(y)\, dy,$$

with

$$||u(x)-u(y)||_a = \int_\Omega G_a(t)|u(x+t)-u(y+t)|^2 dt,$$

$$w(x,y) = \exp\left(-\frac{||u(x)-u(y))||_a}{h^2}\right), \quad d(x) = \int_\Omega w(x,y)dy, \tag{5.1}$$

and $G_a(t)$ a Gaussian with standard deviation a.

Similar to the segmentation in section 4.3, the norm $||\cdot||_a$ is defined using an image neighborhood. Unlike section 4.3, the NLM algorithm uses a Gaussian weighted norm so the values of pixels closer to the center pixel has a larger influences on the similarity between two neighborhoods.

Given this analog with graph theory, the NLM weight matrix can be related to the random walk graph Laplacian. A comparison of equations (2.7) and (5.1) demonstrates that $NL(u)(x) = D^{-1}W$. Substituting $NL(u)(x) = D^{-1}W$ into (2.7) gives the relationship

$$L_w = \mathbf{1} - NL(u)(x).$$

The operator L_w, and therefore the NLM operator, naturally occurs in the gradient flow of a weighted L^2 norm. To see this, consider the weighted L^2 inner product

$$\langle u,v\rangle_{d(x)} = \int_\Omega u(x)\, v(x)\; d(x)dx,$$

where $d(x)$ is the degree function. With this inner product we can write

$$\begin{aligned}\langle u, L_w u\rangle_{d(x)} &= \int_\Omega u(x)\left(\int_\Omega (u(x)-u(y))\frac{1}{d(x)}w(x,y)dy\right)d(x)dx\\ &= \int_\Omega u(x)\int_\Omega (u(x)-u(y))w(x,y)dy\; dx\\ &= \langle u, Lu\rangle.\end{aligned}$$

Therefore, there is a natural relationship between the weighted inner product and the non-weighted inner product. This last equation is symmetric when x and y are interchanged, therefore we can write the energy functional

$$E(u) = \int_\Omega\int_\Omega (u(x)-u(y))^2 w(x,y)\; dy\, dx = \frac{1}{2}\langle u, (L_w u)\rangle_{d(x)}.$$

Note that the energy $\langle u, L_w u\rangle_{d(x)}$ is a well defined energy functional. The gradient flow in the weighted norm $\langle\cdot\rangle_{d(x)}$is

$$\frac{\partial u}{\partial t} = -L_w u = -(u(x) - NL(u)(x)).$$

This equation describes a diffusion process using the NLM operator.

Zhou and Scholkopf [59] and Gilboa and Osher [28, 29] derive a calculus based on the nonlocal operators. The former for the discrete graph case and the latter in a continuum setting that was subsequently discretized in computational examples. Zhou and Scholkopf mainly study graph versions of the Poisson equation and its variants. In the continuum case, Gilboa and Osher define the nonlocal derivative for $y, x \in \Omega$ as

$$\partial_y u(x) = (u(y) - u(x))\sqrt{w(x,y)}, \tag{5.2}$$

where $0 \leq w(x,y) < \infty$ is the symmetric weight matrix in (2.8). Their nonlocal gradient $\nabla_w u : \Omega \to \Omega \times \Omega$has the form

$$(\nabla_w u)(x,y) = (u(x) - u(y))\sqrt{w(x,y)}.$$

The nonlocal divergence $\mathrm{div}_w \vec{v}(x) : \Omega \times \Omega \to \Omega$ is

$$(\mathrm{div}_w \vec{v})(x) = \int_\Omega (v(x,y) - v(y,x))\sqrt{w(x,y)}dy,$$

which is the adjoint of the nonlocal gradient using the above inner product. Finally, the nonlocal Laplacian can be written as

$$\nabla_w^2 u(x) = \frac{1}{2}\mathrm{div}_w(\nabla_w u(x)) = -\int_\Omega (u(x) - u(y))w(x,y)dy. \tag{5.3}$$

This equation is the continuous equivalent of the standard graph Laplacian, which is a different normalization from the one we use. Gilboa and Osher use this calculus to establish a nonlocal total variation energy functional which proves to be highly effective for problems in image inpainting and semi-supervised learning. It would be interesting to establish a formal connection between our Ginzburg-Landau algorithm and the NL-TV method.

5.3. Geometric diffusion. Coifman *et. al* in [15] discuss a diffusion map formulation to investigate the inherent structure in data, and to segment high dimensional data sets. Their construction consists of defining a weight matrix $w(x,y)$ with admissibility properties satisfied by the Gaussian similarity function $w(x,y) = \exp(-||x-y||^2/\tau)$ used in this paper. A major contribution of geometric diffusion is the observation that a correctly normalized graph Laplacian operator converges to the Laplace-Beltrami operator on a manifold.

A data segmentation procedure was introduced by Coifman *et al.* using Geometric Diffusion approach [15]. The technique was adapted to images by Szlam, Maggioni and Coifman in [47]. Let Ω_1 be the set of points in class one, Ω_2 be the set of points in class two, and Ω_3 be the unlabeled points. Their procedure for a two class segmentation problem consists of the following steps:

1. Initialize the functions
$$u_0^{(i)}(x) = \begin{cases} 1 & x \in \Omega_i \\ 0 & \text{otherwise.} \end{cases} \tag{5.4}$$
2. Create the similarity function $w_{LB}(x,y)$ using feature vectors derived from a neighborhood of each pixel.
3. Diagonalize the matrix $w_{LB}(x,y) = \sum_j \lambda_j \phi_j(x)\phi_j(y)$. This step can be performed using the Nystöm extension.
4. Calculate $u_t^{(i)}(x) = \sum_j \lambda_j^t \phi_j(x) \int \phi_j(y) u_0^{(i)}(y)dy$, where the parameter t is chosen by cross-validation with the initial labels.
5. At each point x, assign the class according to $argmax_i \, \{u_t^{(i)}(x)\}$.

This equation exploits the result that w_{LB} is an approximation to the Laplace-Beltrami operator, and therefore w_{LB} is an approximation to the fundamental solution of the Laplace-Beltrami operator [37].

6. Conclusion. In summary, this paper develops a class of algorithms for approximating $L1$ (TV) regularization for classification of high dimensional data. The algorithms are inspired by classical physical models for diffuse interfaces involving multiple scales, including a diffuse interface scale typically smaller than the bulk features of the problem. Such models have recently been introduced to the image processing literature and have been rigorously connected to methods based on total variation. These models are known to produce reasonably sharp edges in image problems provided the diffuse interface scale is smaller than the features of interest in the image. By analogy we develop a graph-based method in which the graph Laplacian takes on the role of the spatial Laplace operator in the physical problem. Fast methods can be derived for solving the minimization problem provided that a reasonably fast algorithm exists for diagonalizing the graph Laplacian. For the classical physics problem there are well-known methods based on the fast Fourier transform which diagonalizes the Laplace operator. For our problem we consider standard sparse matrix methods in the case of sparse graphs and the Nyström extension in the case of highly connected graphs. In all cases we find the results to be comparable to state of the art L1-methods but with a faster compute time.

This paper is the first step in developing the Ginzburg-Landau functional for classification of graph-based data. Many interesting open problems remain. One simple observation is that our iterative method is based on numerical solution of a gradient descent for a nonlinear functional. We use fixed time steps for this method and one would expect possibly significant speed up with an efficient variable time step method. One also can exploit variation in the scale parameter ϵ during this minimization procedure. This idea was used to great advantage in earlier work on image inpainting using the Cahn-Hilliard equation [7, 6]. While the theory of gamma convergence is well-known for the classical Ginzberg-Landau operator and for its wavelet-based cousin [17], no such theory exists for the graph-based problem and this is a very interesting and important problem. Our results suggest that the two should be connected but no rigorous results exist to date. Finally we mention that the GL functional leads to a simplified algorithm for piecewise-constant image segmentation using a carefully designed alternation between evolution of the heat equation and thresholding [23]. The same kind of procedure could be extended to our method although again it would be important to develop a theoretical framework for this idea and its best practice.

Acknowledgments. We thank Lawrence Carin, John Greer, Gary Hewer, Stanley Osher and Yves van Gennip for helpful comments.

REFERENCES

[1] http://www.vision.caltech.edu/html-files/archive.html.
[2] Microsoft Research Cambridge object recognition image database, version 1.0., 2005. http://research.microsoft.com/downloads.
[3] Luigi Ambrosio and V. M. Tortorelli. On the approximation of free discontinuity problems. *Boll. Un. Mat. Ital. B (7)*, 6(1):105–123, 1992.
[4] S Amghibech. Eigenvalues of the discrete p-Laplacian for graphs. *Ars Combin.*, 67:283–302, 2003.
[5] S. Belongie, C. Fowlkes, F. Chung, and J. Malik. Partitioning with indefinite kernels using the Nyström extension. *ECCV Copenhagen*, 2002.
[6] Andrea Bertozzi, Selim Esedoḡlu, and Alan Gillette. Analysis of a two-scale Cahn-Hilliard model for binary image inpainting. *Multiscale Model. Simul.*, 6(3):913–936, 2007.
[7] Andrea L. Bertozzi, Selim Esedoḡlu, and Alan Gillette. Inpainting of binary images using the Cahn-Hilliard equation. *IEEE Trans. Image Process.*, 16(1):285–291, 2007.
[8] Andrea L. Bertozzi, Ning Ju, and Hsiang-Wei Lu. A biharmonic modified forward time stepping

method for fourth order nonlinear diffusion equations. *Discrete and Continuous Dynamical Systems*, 29(4):1367–1391, 2011.
[9] A. Buades, B. Coll, and J. M. Morel. A review of image denoising algorithms, with a new one. *Multiscale Modeling and Simulation*, 4:490530, 2005.
[10] Thomas Bühler and Matthias Hein. Spectral clustering based on the graph p-Laplacian. In *Proceedings of the 26th International Conference on Machine Learning*, pages 81–88, 2009.
[11] Italo Capuzzo Dolcetta, Stefano Finzi Vita, and Riccardo March. Area-preserving curve-shortening flows: from phase separation to image processing. *Interfaces Free Bound.*, 4(4):325–343, 2002.
[12] T. F. Chan and L. A. Vese. Active contours without edges. *IEEE Trans. Imag. Proc.*, 10(2):266–277, 2001.
[13] Tony Chan and Jianhong Shen. *Image Processing And Analysis: Variational, Pde, Wavelet, And Stochastic Methods.* Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2005.
[14] Fan R. K. Chung. *Spectral graph theory*, volume 92 of *CBMS Regional Conference Series in Mathematics.* Published for the Conference Board of the Mathematical Sciences, Washington, DC, 1997.
[15] R. R. Coifman, S. Lafon, A. B. Lee, M. Maggioni, B. Nadler, F. Warner, and S. W. Zucker. Geometric diffusions as a tool for harmonic analysis and structure definition of data: Diffusion maps. *Proc. Nat. Acad. Sci.*, 102(21):7426–7431, 2005.
[16] J. Darbon and M. Sigelle. A fast and exact algorithm for total variation minimization. *Pattern recognition and image analysis*, 3522:351–359, 2005.
[17] J. Dobrosotskaya and A. L. Bertozzi. Wavelet analogue of the Ginzburg-Landau energy and its gamma-convergence. *Interfaces and Free Boundaries*, 12(2):497–525, 2010.
[18] Julia A. Dobrosotskaya and Andrea L. Bertozzi. A wavelet-Laplace variational technique for image deconvolution and inpainting. *IEEE Trans. Image Process.*, 17(5):657–663, 2008.
[19] Petros Drineas and Michael W. Mahoney. On the nystrm method for approximating a gram matrix for improved kernel-based learning. *The Journal of Machine Learning Research*, 6:2153 – 2175, 2005.
[20] Selim Esedoglu. Blind deconvolution of bar code signals. *Inverse Problems*, 20(1):121–135, 2004.
[21] Selim Esedoglu and Riccardo March. Segmentation with depth but without detecting junctions. *J. Math. Imaging Vision*, 18(1):7–15, 2003. Special issue on imaging science (Boston, MA, 2002).
[22] Selim Esedoglu and Jianhong Shen. Digital inpainting based on the Mumford-Shah-Euler image model. *European J. Appl. Math.*, 13(4):353–370, 2002.
[23] Selim Esedoḡlu and Yen-Hsi Richard Tsai. Threshold dynamics for the piecewise constant Mumford-Shah functional. *J. Comput. Phys.*, 211(1):367–384, 2006.
[24] David J. Eyre. An unconditionally stable one-step scheme for gradient systems, 1998. Department of Mathematics, Univ. of Utah.
[25] C. Fowlkes, S. Belongie, and J. Malik. Efficient spatiotemporal grouping using the Nyström method. *CVPR, Hawaii*, 2001.
[26] Charles Fowlkes, Serge Belongie, Fan Chung, and Jitendra Malik. Spectral grouping using the Nyström method. *IEEE Trans. Patt. Anal. Mach. Int.*, 26(2), 2004.
[27] A. Frank and A. Asuncion. UCI machine learning repository, 2010.
[28] Guy Gilboa and Stanley Osher. Nonlocal linear image regularization and supervised segmentation. *Multiscale Modeling and Simulation*, 6(2):595–630, 2007.
[29] Guy Gilboa and Stanley Osher. Nonlocal operators with applications to image processing. *Multiscale Modeling and Simulation*, 7(3):1005–1028, 2008.
[30] Aristides Gionis, Heikki Mannila, and Panayiotis Tsaparas. Clustering aggregation. *ACM Trans. Knowl. Discov. Data*, 1, March 2007.
[31] T. Goldstein and S. Osher. The split Bregman method for $L1-$regularized problems. *SIAM J. Imaging Sci.*, 2(2):323–343, 2009.
[32] S Guattery and G. L. Miller. On the quality of spectral separators. *SIAM Journal of Matrix Anal. Appl.*, 19(3):701–719, 1998.
[33] H. Hein and T. Buehler. An inverse power method for nonlinear eigenproblems with applications in 1-spectral clustering and sparse PCA. In *Advances in Neural Information Processing Systems 23 (NIPS 2010)*, 2010.
[34] David Kay and Richard Welford. A multigrid finite element solver for the Cahn-Hilliard equation. *J. Comp. Phys.*, 212(1):288–304, 2006.
[35] J. Kim, K. Kang, and J. Lowengrub. Conservative multigrid methods for Cahn-Hilliard fluids. *J. Comp. Phys.*, 193(2):551–543, 2004.

[36] Robert V. Kohn and Peter Sternberg. Local minimisers and singular perturbations. *Proc. Roy. Soc. Edinburgh Sect. A*, 111(1-2):69–84, 1989.

[37] S. S. Lafon. *Diffusion Maps and Geometric Harmonics.* PhD dissertation, Yale University, May 2004.

[38] K. Lang. Fixing two weaknesses of the spectral method. In *Advances in Neural Information Processing Systems 28 (NIPS 2006)*, 2006.

[39] A. McAdams, E. Sifakis, and J. Teran. A parallel multigrid Poisson solver for fluids simulation on large grids. In M. Otaduy and Z. Popovic, editors, *ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA)*, pages 1–10, 2010.

[40] Barry Merriman, James K. Bence, and Stanley J. Osher. Motion of multiple functions: a level set approach. *J. Comput. Phys.*, 112(2):334–363, 1994.

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

[42] Chotirat Ann Ratanamahatana and Dimitrios Gunopulos. Scaling up the naive Bayesian classifier: Using decision trees for feature selection. *Proceedings of the IEEE Workshop on Data Cleaning and Preprocessing (DCAP)*, 2002. at IEEE International Conference on Data Mining (ICDM 2002), Maebashi, Japan.

[43] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[44] M. Stoer and F. Wagner. A simple min-cut algorithm. *Journal of the ACM*, 44(4):585–591, 1997.

[45] Arthur Szlam and Xavier Bresson. A total variation-based graph clustering algorithm for Cheeger ratio cuts, 2009. UCLA CAM Preprint number 09-68.

[46] Arthur Szlam and Xavier Bresson. Total variation and Cheeger cuts. In Johannes Fürnkranz and Thorsten Joachims, editors, *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 1039–1046, Haifa, Israel, June 2010. Omnipress.

[47] Arthur D. Szlam, Mauro Maggioni, and Ronald R. Coifman. Regularization on graphs with function-adapted diffusion processes. *Journal of Machine Learning Research*, 9:1711–1739, 2008.

[48] Joost van de Weijer and Cordelia Schmid. Coloring local feature extraction. In Ale Leonardis, Horst Bischof, and Axel Pinz, editors, *Computer Vision ECCV 2006*, volume 3952 of *Lecture Notes in Computer Science*, pages 334–348. Springer Berlin / Heidelberg, 2006.

[49] Manik Varma and Andrew Zisserman. A statistical approach to texture classification from single images. *International Journal of Computer Vision*, 62:61–81, 2005. 10.1007/s11263-005-4635-4.

[50] Benjamin P. Vollmayr-Lee and Andrew D. Rutenberg. Fast and accurate coarsening simulation with an unconditionally stable time step. *Phys. Rev. E*, 68(6):066703, Dec 2003.

[51] Ulrike von Luxburg. A tutorial on spectral clustering, 2006. Technical Report No. TR-149, Max Planck Institute for Biological Cybernetics.

[52] Ulrike von Luxburg, Mikhail Belkin, and Olivier Bousquet. On the convergence of spectral clustering on random samples: the normalized case. In John Shawe-Taylor and Yoram Singer, editors, *17th Annual conference on Learning Theory, COLT 2004, Banff, Canada, July 1-4, 2004*, volume 3120 of *Lecture Notes in Computer Science*, pages 457–471, Berlin Heidelberg, 2004. Springer-Verlag.

[53] Ulrike von Luxburg, Mikhail Belkin, and Olivier Bousquet. Consistency of spectral clustering. *Annals of Statistics*, 36(2):555–586, 2008.

[54] D. Wagner and F. Wagner. Between min cut and graph bisection. *Mathematical Foundations of Computer Science: Lecture Notes in Computer Science*, 711:744–750, 1993.

[55] Yaroslavsky. *Digital Picture Processing, an Introduction.* Springer-Verlag, Berlin, 1985.

[56] A.L. Yuille and A. Rangarajan. The concave-convex procedure (CCCP). *Neural Computation*, 15(4):915–936, 2003.

[57] L. Zelnik-Manor and P. Perona. Self-tuning spectral clustering. In *Advances in Neural Information Processing Systems 17 (NIPS 2004)*, volume 17, pages 1601–1608, 2004.

[58] Xiaojin Zhu, Zoubin Ghahramani, and John. Lafferty. Semi-supervised learning using Gaussian fields and harmonic functions. *Proceedings of the Twentieth International Conference on Machine Learning (ICML)*, 20:912–919, 2003.

[59] D. Zou and B. Scholkopf. A regularization framework for learning from graph data. *ICML workshop on statistical learning and its connection to other fields*, 2004.