Published in Image Processing On Line on 2017–08–19.
Submitted on 2017–02–18, accepted on 2017–06–13.
ISSN 2105–1232 © 2017 IPOL & the authors CC–BY–NC–SA
This article is available online with supplementary materials,
software, datasets and online demo at
https://doi.org/10.5201/ipol.2017.204

Hyperspectral Image Classification Using Graph Clustering Methods

Zhaoyi Meng[1], Ekaterina Merkurjev[2], Alice Koniges[3], Andrea L Bertozzi[4]

[1] Department of Mathematics, UCLA (mzhy@ucla.edu)
[2] Department of Mathematics, MSU (kmerkurev@math.msu.edu)
[3] Lawrence Berkeley National Laboratory (aekoniges@lbl.gov)
[4] Department of Mathematics, UCLA (bertozzi@math.ucla.edu)

Communicated by Miguel Colom *Demo edited by* Nelson Monzón

Abstract

Hyperspectral imagery is a challenging modality due to the dimension of the pixels which can range from hundreds to over a thousand frequencies depending on the sensor. Most methods in the literature reduce the dimension of the data using a method such as principal component analysis, however this procedure can lose information. More recently methods have been developed to address classification of large datasets in high dimensions. This paper presents two classes of graph-based classification methods for hyperspectral imagery. Using the full dimensionality of the data, we consider a similarity graph based on pairwise comparisons of pixels. The graph is segmented using a pseudospectral algorithm for graph clustering that requires information about the eigenfunctions of the graph Laplacian but does not require computation of the full graph. We develop a parallel version of the Nyström extension method to randomly sample the graph to construct a low rank approximation of the graph Laplacian. With at most a few hundred eigenfunctions, we can implement the clustering method designed to solve a variational problem for a graph-cut-based semi-supervised or unsupervised classification problem. We implement OpenMP directive-based parallelism in our algorithms and show performance improvement and strong, almost ideal, scaling behavior. The method can handle very large datasets including a video sequence with over a million pixels, and the problem of segmenting a data set into a pre-determined number of classes.

Source Code

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

Keywords: multi-class classification; hyperspectral video; energy minimization; MBO scheme; Nyström extension method; parallel computing

[1] https://doi.org/10.5201/ipol.2017.204

1 Introduction

Multi-class classification is one of the fundamental problems in machine learning and computer vision. In this paper, we outline two very efficient methods to classify data sets, so that the similarity between nodes in one class is much larger than the similarity between nodes of different classes. One application of this work is the class detection of hyperspectral images, where each pixel contains many channels. We use the graphical framework, where we consider each pixel as a node on a graph, and take the values in the channels to form the feature vectors [60]. To make our codes even more efficient, we implement OpenMP directive-based parallelism in our procedures and observe strong scaling behavior.

A traditional way of solving the hyperspectral image classification problem is to first use dimension reduction and then a classifier. Many feature extraction and dimension reduction techniques have been developed for hyperspectral image classification, such as principal component analysis (PCA) [29], independent component analysis [77], signal subspace identification [7], discrete wavelet transform [45], band reduction based on rough sets [66], projection pursuit algorithm [44], and clonal selection feature-selection algorithm [79]. The Fuzzy C-Means (FCM) algorithm is a well-known tool to find proper clusters, which can be used for hyperspectral image classification [43], and can be further enhanced by the Support Vector Domain Description [65].

Support vector machines (SVM) [9, 73] are another popular approach to the supervised classification problem, including one involving hyperspectral data [38, 55, 18]. Similar approaches include transductive SVM [15] and SVM with composite kernels [19]. Some of these methods use kernels, which have been shown to improve performance [18, 48], especially when the data is not linearly separable. Examples of kernel methods can be found in [75, 32, 78]. Other successful techniques include multinomial logistic regression [8, 47], which was applied to hyperspectral image segmentation in [49, 50], and sparse representation, also applied to such images in [23, 22]. The reader is referred to [52, 70] for more approaches.

As a basis for both of our algorithms, we use the graphical framework, described in more detail in [60]. This framework provides several advantages; for example, it allows for a general incorporation of information of any kind of data set- video data, text, images, etc. It also brings forth a way to work with nonlinearly separable classes [34], i.e. when the data is not linearly separable. Moreover, in the case of image processing, the framework allows one to easily capture texture and patterns throughout the image [16, 60, 37, 36] using the non-local means feature vector. The graphical approach has appeared in many recent works for different kinds of applications: semi-supervised learning [21, 57, 2, 34, 60, 59, 61, 3], image processing [26, 80, 25], machine learning [81], graph cuts [10, 39, 67, 69, 11, 13, 12, 58].

In this work, we introduce two novel graph-based classification algorithms. The first one is semi-supervised [61], which requires some known labels as input, and the second one is unsupervised [42]. Both methods make use of the MBO scheme, which is a well-established PDE method for evolving an interface by mean curvature [62, 63], which is adapted to the graph setting [34, 33]. The supervised algorithm minimizes an energy functional that is a sum of a graph cut term and a least squares fit to the training data. The unsupervised algorithm is derived from the Chan-Vese method [20, 74] likewise adapted to the graph setting. It has an energy functional that is the sum of a graph cut term and a least squares fit to the average value of pixels in that class.

For hyperspectral images, the pixel dimension can be very large. This motivated us to develop parallel implementations and optimizations of these two new algorithms [56]. In particular, for computations, we use an optimized implementation of the Nyström extension eigensolver on high performance computing systems. Moreover, after analyzing the computational hotspots, we implement directive-based OpenMP parallelization. In practice, we obtain strong scaling results and extremely fast implementations.

Main Contributions

The main contributions of the paper are the following:

- We describe two efficient data classification algorithms to cluster a data set into a pre-determined number of classes. The methods are applied specifically to hyperspectral data. While the first algorithm is semi-supervised (requiring a subset of the pixels to have known labels), the second method is unsupervised, allowing us to proceed when no ground truth is available.
- The new parallel methods provide performance and accuracy advantages over traditional data classification algorithms in serial mode. The efficiency is largely due to new implementations of our spectral solver built on the Nyström extension method. For the 40 frames of the plume video data, the entire classification process takes only 1 minute using 32 threads on a supercomputer!
- We implement OpenMP directive-based parallelism in our algorithms, and show performance improvement and strong (almost ideal) scaling behavior due to the parallelization. We optimize the OpenMP implementations and show performance results on IPOL server Purple (Intel X7560 Nehalem) and traditional supercomputer nodes (Intel Haswell).
- These methods have been introduced in earlier conference papers [61, 42] as serial implementations and as a parallel implementation [56] on a supercomputer, without online code. Here we develop new parallel versions for real time implementations on the IPOL server for both hyperspectral imagery and the non local means functional for segmentation of RGB images.

The paper is organized as follows. In Section 2, we describe the graph model and provide some background information. In Section 3, the details of our algorithms and the Nyström extension technique are presented. In Section 4, we elaborate on our numerical results using hyperspectral and RGB data sets. Section 5 contains details on parallelization and scaling. Section 6 concludes the paper.

2 Background

2.1 Graphical Representation

The two classification algorithms are based on the graph setting [24, 76]. The framework involves an undirected graph $G = (V, E)$, where $V$and E are sets of nodes and edges, respectively. Pairs of nodes of the graph are connected by edges, each of which is assigned a weight, which describes the similarity between the nodes the edge is connecting. A high-valued weight indicates that the two vertices are similar, and a low-valued weight indicates they are dissimilar.

We embed our data into a graphical framework, and use the weight function given by the formula

$$w_{ij} = \exp(-||x_i - x_j||^2/\tau), \tag{1}$$

where x_i and x_j are feature vectors of nodes i and j, and τ is a parameter to be determined. More details about how to choose τ can be found in [4]. In the case of hyperspectral images, each node in a graph is a pixel in the image, and the feature vector of a pixel is a vector of intensity values in its many bands. We use the cosine norm to compare two feature vectors.

The degree of a node $i \in V$ is defined as $d_i = \sum_{j \in V} w_{ij}$. If D is the diagonal matrix with elements d_i and $W = \{w_{ij}\}$ is the weight matrix, the graph Laplacian is defined as $L = D - W$. For scaling purposes, we use the normalized symmetric Laplacian L_s, which is defined as

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

Another common version of the normalization, related to the discrete Markov process, is the random walk Laplacian given by

$$L_w = D^{-1}L = I - D^{-1}W. \tag{3}$$

We note that the graph Laplacians satisfy the following properties [24, 76]:

1) L and L_s are symmetric.

2) L, L_s and L_w are positive semi-definite matrices.

3) λ is an eigenvalue of L_w with eigenvector u if and only if λ is an eigenvalue of L_s with eigenvector $w = D^{\frac{1}{2}}u$.

4) L, L_s and L_w have non-negative, real-valued eigenvalues $0 = \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_N$. Note that 0 is an eigenvalue of L and L_w with a constant one eigenvector $\mathbb{1}$. Due to 3), 0 is an eigenvalue of L_s with eigenvector $D^{1/2}\mathbb{1}$.

In this paper, we use a fully connected graph, and also use non-local operators, which allow the algorithm to capture patterns and texture in the image by using non-local information [37, 36].

2.2 Spectral Clustering and K-means

Our algorithms involve computing eigenvalues and eigenvectors of the graph Laplacian. This information is also used in the technique of spectral clustering, so we review it here.

Spectral clustering [76] is a popular approach for clustering a data set into several classes. The method requires the data set to be embedded in a graph framework and the eigenvectors of the graph Laplacian (or the random walk Laplacian) to be computed. The procedure is described in Algorithm 1.

Algorithm 1: Spectral Clustering

Input: Graph Laplacian L (or L_w), number K of clusters to construct.
Output: Clusters $A_1, \ldots, A_K$ with $A_i = \{j | y_j \in C_i\}$.
1 Compute first K eigenvectors $v_1, \ldots, v_K$ of L (or L_w).
2 Let $V \in \mathbb{R}^{N \times K}$ be the matrix containing the vectors $v_1, \ldots, v_K$ as columns.
3 For $i = 1, \ldots, N$, let $y_i \in \mathbb{R}^K$ be the vector corresponding to the i^{th} row of V.
4 Cluster the points $(y_i)_{i=1,\ldots,N}$ in $\mathbb{R}^K$ with the K-means algorithm into clusters $C_1, \ldots, C_K$.

The K-means Algorithm [53] for finding K clusters proceeds iteratively by first choosing K centroids and then assigning each point to the cluster of the nearest centroid. The centroid of each cluster is then recalculated and the iterations continue until there is little change from one iteration to the next. A generalized version of spectral clustering using the p-Laplacian is proposed in [17].

3 Graph-based Classification Algorithms

3.1 Semi-supervised Algorithm

In this section, we describe how to use eigenvectors of the Laplacian to minimize a semi-supervised graph model. In semi-supervised learning, the fidelity, or a small amount of "ground truth", is known and the rest of the data set needs to be classified according to the categories of the known data [61].

We approach the semi-supervised classification problem using energy minimization techniques. Similar approaches have been used in [4, 5], where the problem is formulated as a minimization of the Ginzburg-Laudau (GL) functional (in graph form) with a fidelity term. In [60], the authors propose an MBO scheme to solve the binary classification problem; a multi-class extension of that algorithm is described in [34, 59].

The problem is to classify a data set with N elements into $\hat{n}$ classes, where $\hat{n}$ is to be provided to the algorithm in advance. We work with an assignment matrix u, which is an $N \times \hat{n}$ matrix, where each row is an element of the Gibbs simplex $\Sigma^{\hat{n}}$, defined as

$$\Sigma^{\hat{n}} := \left\{ (x_1, \dots, x_{\hat{n}}) \in [0,1]^{\hat{n}} \,\middle|\, \sum_{k=1}^{\hat{n}} x_k = 1 \right\}. \tag{4}$$

Therefore, each row of u is a probability distribution; in fact, the k^{th} component of the i^{th} row of u is the probability the i^{th} node belongs to class k. In the text that follows, we denote the i^{th} row of u by u_i. Let us also denote by e_k the k^{th} vertex of the simplex, where all the entries are zero, except the k^{th} one, which is equal to one.

The optimization problem we consider consists of minimizing the following energy

$$E(u) = \epsilon \langle u, L_s u \rangle + \frac{1}{\epsilon} \sum_i W(u_i) + \sum_i \frac{\mu}{2} \lambda(x_i) ||u_i - \hat{u_i}||_{L_2}^2, \tag{5}$$

encountered also in [34, 60, 59]. The first two terms of (5) comprise the graph form of the Ginzburg-Landau functional, where L_s is the symmetric Laplacian, ϵ is a small positive constant, and $W(u_i)$ is the multi-well potential in $\hat{n}$ dimensions, where $\hat{n}$ is the number of classes

$$W(u_i) = \prod_{k=1}^{\hat{n}} \frac{1}{4} ||u_i - e_k||_{L_1}^2. \tag{6}$$

The last term of (5) is the regular L_2 fit to known data with some constant μ, while $\lambda(x)$ takes the value of 1 on fidelity nodes, and 0 otherwise. The variable $\hat{u}$ is the initial value for u with randomly chosen labels for non-fidelity data points and the "ground truth" for the fidelity points. Lastly, in (5), for matrices A and B, $\langle A, B \rangle = \text{trace}(A^T B)$, where A^T indicates A transpose.

Minimizing $E(u)$ by the gradient descent method, one obtains

$$\frac{\partial u}{\partial t} = -\epsilon L_s u - \frac{1}{\epsilon} W'(u) - \mu \lambda(x)(u - \hat{u}). \tag{7}$$

This is the Allen-Cahn Equation [1, 30] with fidelity term with the differential operator Δu replaced by a more general graph operator $-L_s$ [51]; when $\epsilon \to 0$, the solution to the Allen-Cahn equation approximates motion by mean curvature [62]. Note that in the last term of (7), the product is meant to be calculated on each node.

In [34], the authors propose an MBO scheme to solve (7). We slightly modify this scheme to solve (7) in the formulation of our semi-supervised method.

We now present the semi-supervised algorithm, detailed in Algorithm 2, and which is based on that of [34]. For initialization, which we denote by $\hat{u}$, we use the known labels for the fidelity points and random class labels for non-fidelity points. To obtain the next iterate of u, one proceeds with the following two steps:

- Step 1: Heat equation with forcing term:

$$\frac{u^{n+\frac{1}{2}} - u^n}{dt} = -L_s u^n - \mu \lambda(x)(u^n - \hat{u}), \tag{8}$$

- Step 2: Threshold
$$u_i^{n+1} = e_r, r = \arg\max u_i^{n+\frac{1}{2}}, \tag{9}$$
for all $i \in \{1, 2, \ldots, N\}$, where e_r is the r^{th} standard basis in $\mathbb{R}^{\hat{n}}$.

For a stopping criterion, we compute the norm of the difference between the label matrix u of two consecutive iterations and stop the iteration when the norm is below a threshold value. Let us denote the final u by u_f. To obtain the final classification of node i, we find the largest value in the i^{th} row of u_f and assign the corresponding index as the class label of node i. For a more thorough discussion about the MBO scheme and motion by mean curvature on graphs, the reader is referred to [72].

Step 1 can be computed very efficiently and simply by using the eigendecomposition of L_s, which is
$$L_s = X\Lambda X^T, \tag{10}$$
where X is the eigenvector matrix and Λ is a diagonal matrix containing the eigenvalues. We approximate X by a truncated matrix retaining only a small number of the leading eigenvectors. If we write
$$u^n = Xa^n, \quad \mu\lambda(x)(u^n - \hat{u}) = Xd^n, \tag{11}$$
and equate coefficients, we can formulate Step 1 in the MBO scheme as solving for the coefficients a_k^{n+1}
$$a_k^{n+1} = (1 - dt\lambda_k) \cdot a_k^n - dt \cdot d_k^n, \tag{12}$$
where λ_k is the k^{th} eigenvalue of L_s, in ascending order.

Due to the fact that, in practice, only the leading eigenvalues and eigenvectors (in ascending order) need to be calculated to obtain good accuracy, (12) is an efficient way to compute Step 1 of the algorithm, even in the case when the number of classes is very large. This feature of the method makes the procedure very fast.

Empirically, the algorithm converges after a small number of iterations. Note that the iterations stop when a purity score between the partitions from two consecutive iterations is greater than 99.99%. The purity score, as used in [41], measures how "similar" two partitions are. Intuitively, it can be viewed as the fraction of nodes of one partition that have been assigned to the correct class with respect to the other partition.

3.2 Unsupervised Algorithm

In this section, we formulate an unsupervised algorithm to handle the case when there is no knowledge of the class of any part of the data set. Our method is based on the Mumford-Shah model [64], which is a famous model used for multi-class segmentation problems. One simplified version of the Mumford-Shah model tailored for images is the piecewise constant model [74, 28]
$$E^{MS}(\Phi, \{c_r\}_{r=1}^{\hat{n}}) = |\Phi| + \mu \sum_{r=1}^{\hat{n}} \int_{\Omega_r} (f - c_r)^2, \tag{13}$$
where the contour Φ segments an image region Ω into $\hat{n}$ disjoint sub-regions Ω_r, $|\Phi|$ is the length of the contour, f is the observed image data, μ is a constant, and $\{c_r\}_{r=1}^{\hat{n}}$ is a set of constant values which represent the local centroids.

The graph version of the multi-class piecewise constant Mumford-Shah energy was introduced in [42] for hyperspectral images
$$MS(u, \{c_r\}_{r=1}^{\hat{n}}) = \frac{1}{2}|u|_{TV} + \mu \sum_{r=1}^{\hat{n}} \langle ||f - c_r||^2, u_{\star,r} \rangle, \tag{14}$$

Algorithm 2: Semi-Supervised Algorithm

Input: A data set of N points (embedded in a graphical framework $G = (V, E)$) to be classified into $\hat{n}$ classes, parameters dt and μ and threshold value δ.

Output: matrix C denoting the final class of all nodes.

Initialize u^0 and compute d^0.

Calculate $m << N$ smallest eigenvectors and eigenvalues of the symmetric graph Laplacian L_s. Let X denote the eigenvector matrix.

Set $n = 0$.

while $purity(u^n, u^{n-1}) < 99.99\%$ **do**

  $a^n = X^T \cdot u^n$.

  $a_k^{n+1} = (1 - dt\lambda_k) \cdot a_k^n - dt \cdot d_k^n$ for $k = 1$ to $k = m$.

  Compute $u^{n+\frac{1}{2}}$ via $u^{n+\frac{1}{2}} = Xa^{n+1}$.

  Compute d^{n+1} via (11).

  $u_i^{n+1} = e_r, r = \arg\max u_i^{n+\frac{1}{2}}$ for $i = 1$ to $i = N$.

  $n \leftarrow n + 1$.

If u_f is the final iterate of u, let $C_i = \arg\max(u_f)_i$.

where u is the class assignment matrix (described in the previous section) in which each row is an element of the Gibbs simplex (4). The length of the contour is estimated by the total variation (TV) of the assignment matrix u. In (14), the term $||f - c_r||^2$ denotes an $N \times 1$ vector $(||f(x_1) - c_r||^2, \ldots, ||f(x_N) - c_r||^2)^T$ and the x_i $(i = 1, \ldots, N)$ are the N pixels of the data set. In addition, the term $u_{\star,r}$ indicates the r^{th} column of u; the vector $u_{\star,r}$ is a $N \times 1$ vector which contains the probabilities of every node belonging to class r. Lastly, in (14), $\langle , \rangle$ indicates the usual inner product.

The problem is to classify a data set with N elements into $\hat{n}$ classes, where $\hat{n}$ is to be provided to the algorithm in advance. We work with an assignment matrix u, described in the previous section. For the purpose of segmentation, we need to minimize equation (14). This problem is essentially equivalent to the K-means method when μ approaches $+\infty$.

Minimizing the variation in c yields the following formula for the optimal constants c_r

$$c_r = \frac{\langle f, u_r \rangle}{\sum_{i=1}^{N} u_{\star,r}(x_i)}, \tag{15}$$

where $u_{\star,r}(x_i)$ indicates the i^{th} entry of $u_{\star,r}$.

To motivate our algorithm, we note that the GL functional converges to the TV seminorm [46, 6]; thus, we modify (14) using a diffuse interface approximation

$$E(u, c_r) = \epsilon \langle u, L_s u \rangle + \frac{1}{\epsilon} \sum_i W(u_i) + \mu \sum_{r=1}^{\hat{n}} \langle ||f - c_r||^2, u_{\star,r} \rangle. \tag{16}$$

Similarly to the procedure in Section 3.1, using gradient descent yields

$$\frac{\partial u}{\partial t} = -\epsilon L_s u - \frac{1}{\epsilon} W'(u) - \mu(||f - c_1^n||^2, \ldots, ||f - c_{\hat{n}}^n||^2). \tag{17}$$

We can use the MBO scheme, described in Section 3.1, to solve this minimization problem. A similar thresholding procedure can be found derived in [28]. A class of algorithms for the high order geometric motion of planar curves following a similar thresholding procedure can be found in [27].

We now present our unsupervised algorithm, detailed in Algorithm 3. For initialization of u, we use random labels. To obtain the next iterate of u, one proceeds with the following three steps:

- Step 1: Compute

$$\frac{u^{n+\frac{1}{2}} - u^n}{dt} = -L_s u^n - \mu(||f - c_1^n||^2, \ldots, ||f - c_{\hat{n}}^n||^2). \tag{18}$$

- Step 2: Threshold

$$u_i^{n+1} = e_r, r = \arg\max u_i^{n+\frac{1}{2}}, \tag{19}$$

for all $i \in \{1, 2, \ldots, N\}$, where e_r is the r^{th} standard basis in $\mathbb{R}^{\hat{n}}$.

- Step 3: Update c

$$c_r^{n+1} = \frac{\langle f, u_{\star,r}^{n+1} \rangle}{\langle 1, u_{\star,r}^{n+1} \rangle}. \tag{20}$$

The stopping criteria for this scheme is the same as the one in Section 3.1. The final classification of the nodes is also obtained in the same manner as in Section 3.1.

As in the case of the semi-supervised algorithm, Step 1 can be computed very efficiently and simply by using the eigendecomposition of L_s. Let X be the matrix containing the first $m << N$ orthogonal leading eigenvectors of L, Λ be the diagonal matrix containing the corresponding eigenvalues, and write u^n as $u^n = Xa^n$. Then Step 1 of the algorithm can be approximately computed as

$$u^{n+\frac{1}{2}} = X(1 - dt \cdot \Lambda)a^n - dt \cdot \mu(||f - c_1^k||^2, \ldots, ||f - c_{\hat{n}}^k||^2). \tag{21}$$

Due to the fact that, in practice, only the leading eigenvalues and eigenvectors need to be computed to obtain a good accuracy, (21) is an efficient way to compute Step 1 of the algorithm, even when the number of classes is large. This feature makes this method very fast.

The algorithm also converges after a small number of iterations empirically. Note that the iterations stop when a purity score between the partitions from two consecutive iterations is greater than 99.99%.

Algorithm 3: Unsupervised Algorithm

Input: A data set of N points (embedded in a graphical framework $G = (V, E)$) to be classified into $\hat{n}$ classes, and parameters dt and μ.

Output: matrix C denoting the final class of all nodes.

1 Initialize u^0 and compute a^0 via (11).
2 Calculate $m << N$ smallest eigenvectors and eigenvalues of the symmetric graph Laplacian L_s. Let X= the eigenvector matrix, Λ= the diagonal matrix containing the eigenvalues.
3 Set $n = 0$.
4 **while** $purity(u^n, u^{n-1}) < 99.99\%$ **do**
5 Compute c_r^{n+1} via (20) for $r = 1$ to $r = \hat{n}$.
6 Compute $u^{n+\frac{1}{2}}$ via (21).
7 $u_i^{n+1} = e_r, r = \arg\max(u^{n+\frac{1}{2}})_i$ for $i = 1$ to $i = N$.
8 Compute a^{n+1} via (11).
9 $n \leftarrow n + 1$.
10 If u_f is the final iterate of u, let $C_i = \arg\max(u_f)_i$.

3.3 Background on the Nyström Extension Technique

In the procedure of both the supervised and unsupervised algorithms, we use spectral methods to make the computations more efficient. Due to the fact that we calculate several leading eigenvectors and eigenvalues of the graph Laplacian matrix and project all vectors onto this sub-eigenspace, Step 1 becomes simply updating coefficients.

An approximation to the eigendecomposition of the graph Laplacian matrix can be computed very efficiently by the Nyström extension method [31]. This is achieved by calculating an eigendecomposition of a smaller system of size $m << N$ and then expanding the results back up to N dimensions. The computational complexity is almost $O(N)$. We can set $m << N$ without any significant decrease in the accuracy of the solution. In practice, we also see that only the leading eigenvectors and eigenvalues are needed to obtain an accurate answer.

The method proceeds as follows. Suppose $Z = \{z_i\}_{i=1}^N$ is the whole set of nodes on the graph. By randomly selecting a small subset X, we can partition Z as $Z = X \bigcup Y$. The weight matrix W can be written as

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

where W_{XX} denotes the weights of nodes in set X, W_{XY} denotes the weights between set X and set Y, $W_{YX} = W_{XY}^T$ and W_{YY} denotes the weights of nodes in set Y.

It can be shown that the large matrix W_{YY} can be approximated by $W_{YY} \approx W_{YX} W_{XX}^{-1} W_{XY}$, and the error is determined by how many of the rows of W_{XY} span the rows of W_{YY}. We only need to compute W_{XX}, $W_{XY} = W_{YX}^T$, and it requires only $(|X| \cdot (|X| + |Y|))$ computations versus $(|X| + |Y|)^2$ when the whole matrix is used. The major overhead is computing W_{XY} and we have developed a new parallel code for the first time in the literature. See Section 5 for details.

Based on the definition in formula (2), if ξ is an eigenvalue of $\hat{W} = D^{-1/2} W D^{-1/2}$, then $1 - \xi$ is an eigenvalue of L_s. Therefore, we first calculate the eigendecomposition of $\hat{W}$, and then easily extend it to one of L_s. However, to calculate the eigenvalues and eigenvectors of $\hat{W}$, we need to first calculate normalizations of W_{XX} and W_{XY}.

Let 1_K be the K-dimensional unit vector, and matrices d_X and d_Y be defined as

$$\begin{aligned} d_X &= W_{XX} 1_L + W_{XY} 1_{N-L}, \\ d_Y &= W_{YX} 1_L + (W_{YX} W_{XX}^{-1} W_{XY}) 1_{N-L}. \end{aligned} \tag{22}$$

Let $A./B$ denote component-wise division between matrices A and B, and v^T denote the transpose of vector v; then, the matrices W_{XX} and W_{XY} can be normalized in the following manner to obtain $\hat{W}_{XX}$ and $\hat{W}_{XY}$:

$$\begin{aligned} \hat{W}_{XX} &= W_{XX}./(s_X s_X^T), \\ \hat{W}_{XY} &= W_{XY}./(s_X s_Y^T), \end{aligned} \tag{23}$$

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

It is shown in [4] that if we have the eigendecomposition of two small matrices

$$\hat{W}_{XX} = B_X \Gamma B_X^T, \tag{24}$$

and

$$\hat{W}_{XX} + \hat{W}_{XX}^{-1/2} \hat{W}_{XY} \hat{W}_{YX} \hat{W}_{XX}^{-1/2} = A^T \Xi A, \tag{25}$$

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

$$\Phi = \begin{bmatrix} B_X \Gamma^{1/2} B_X^T A \Xi^{-1/2} \\ \hat{W}_{YX} B_X \Gamma^{-1/2} B_X^T A \Xi^{-1/2} \end{bmatrix}.$$

The diagonal components of $I - \Xi$ contain the corresponding eigenvalues of the symmetric graph Laplacian L_s.

4 Numerical Results

4.1 Urban Data

The Urban data set, available at http://www.tec.army.mil/Hypercube, is one of the most widely used hyperspectral data sets in the hyperspectral image study. It was recorded in October 1995 by the Hyperspectral Digital Imagery Collection Experiment (HYDICE), whose location is an urban area in Copperas Cove, TX, U.S.. The data consists of an image with dimension of 307 x 307 pixels, each of which corresponds to a 2 square meters area. For each pixel, there are 210 channels with wavelengths ranging from 400 nm to 2500 nm, resulting in a spectral resolution of 10 nm. After removing certain channels due to dense water vapor and atmospheric effects, the common clean data set contains 162 channels. We use the ground truth from http://www.escience.cn/people/feiyunZHU/Dataset_GT.html [85], which contains 4 classes with end members corresponding to asphalt, grass, tree and roof, respectively. The goal is to accurately segment the image into the 4 classes.

We apply both the semi-supervised and unsupervised algorithms to this data set. For the semi-supervised algorithm, we randomly select 10% of the ground truth to have known labels.

The classification results are shown in Figure 1. There are four classes in the ground truth; asphalt, grass, trees and roof are labeled in blue, red, green and yellow, respectively. We see that both algorithms are able to obtain a result very close to the ground truth and one that is much more accurate than that of spectral clustering.

4.2 Kennedy Space Center Data

The Kennedy Space Center Data Set was acquired by NASA AVIRIS (Airborne Visible/Infrared Imaging Spectrometer) instrument at the Kennedy Space Center (KSC) in Florida. The data set consists of an image, the dimension of which is 512×614 pixels. The are 224 bands having a width of 10 nm with center wavelengths ranging from 400 to 2500 nm. After removing water absorption and low SNR bands, 176 bands were used for the analysis.

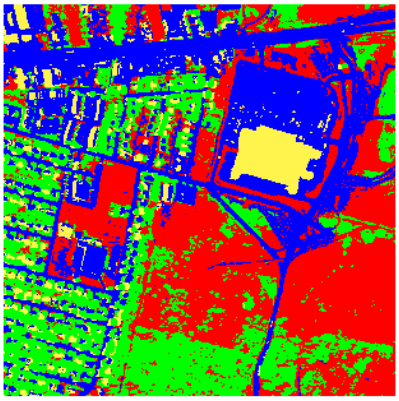
For classification purposes, 13 classes representing the various land cover types that occur in this environment were defined for the site; therefore, the goal is to accurately detect all 13 classes. The public ground truth, which contains only 1.66% of the total pixels, is shown in Figure 2(a). The image of band 50 is shown in Figure 2(b).

We chose 10 pixels per class for fidelity and applied our semi-supervised algorithm on this data set. The classification result representing the whole image is shown in Figure 2(c). The overall accuracy, calculated using known labels, is 80.37%.

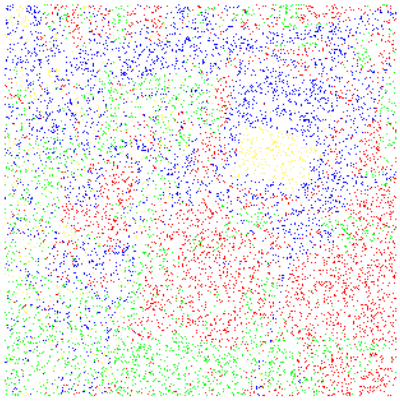
4.3 DC Mall Data

The DC Mall data set was collected with an airborne sensor system located over the Washington DC Mall. The data set consists of an image with dimension of 1280×307 pixels with 210 spectral bands, each of which contains a wavelength in the $400 - 2400$ nm region. However, after elimination of water absorption and noisy bands, only 191 spectral bands remain, and the modified data set is available at http://cobweb.ecn.purdue.edu/~biehl/.

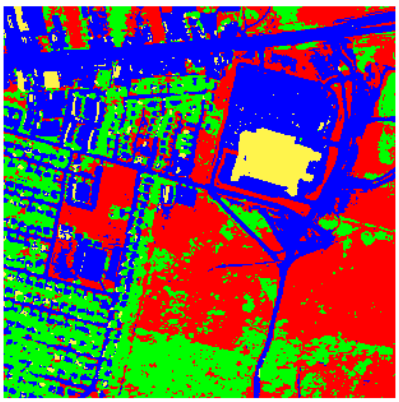
The data set contains 7 classes; thus, the problem becomes to accurately segment the 7 classes. However, we only have 2.06% of the ground truth available and it is shown in Figure 3(a).



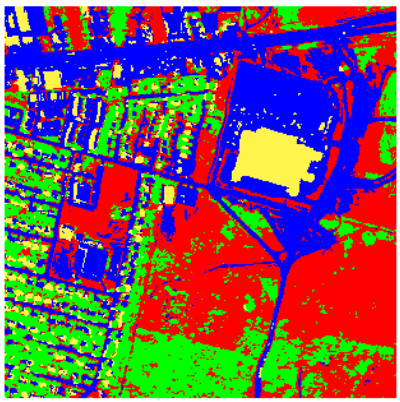
(a) the ground truth with four classes: asphalt (blue), grass (red), trees (green), roof (yellow).



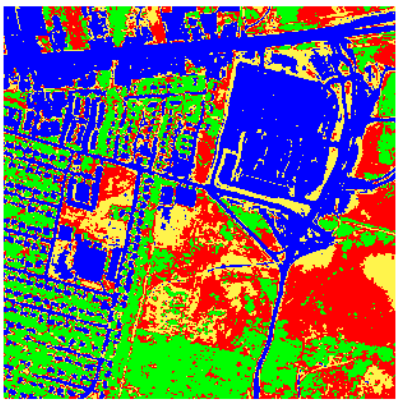
(b) pixels selected to have known labels (10% of the data) for the semi-supervised algorithm.



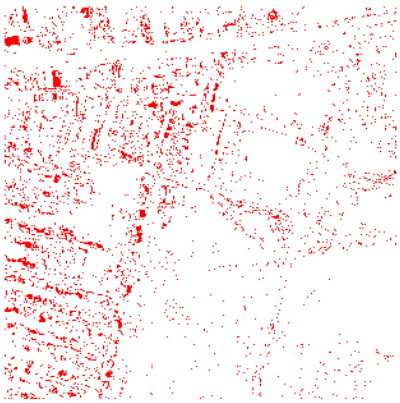
(c) the classification result of the semi-supervised algorithm with the accuracy of 93.48%.



(d) the classification result of the unsupervised algorithm with the accuracy of 92.35%.



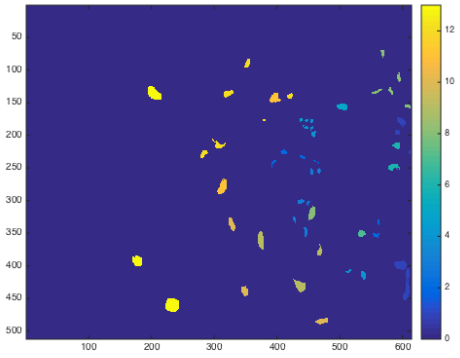
(e) the result of spectral clustering with K-means with the accuracy of 75.06%.



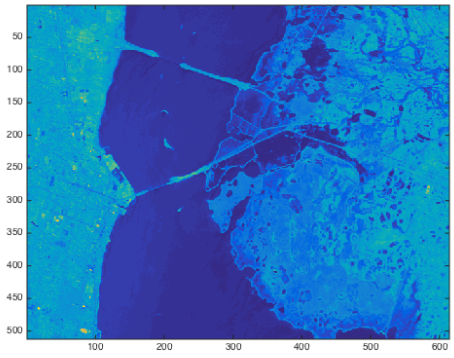
(f) the error of the semi-supervised algorithm.

Figure 1: Urban Data.

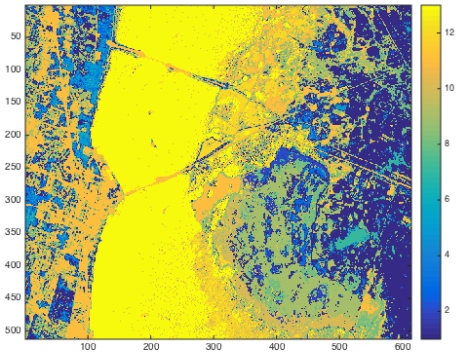
We use 10 pixels per class for fidelity and apply our semi-supervised algorithm on this data set. The result is shown in Figure 3(b), and the overall accuracy is 98.11%. We note that when 20 pixels



(a) Ground truth.



(b) Band 50.



(c) Classification result.

Figure 2: Kennedy Space Center data.

per class are used for fidelity, an even better accuracy of 99.17% is achieved. The algorithm clearly performs very well in clustering the data set into seven classes.

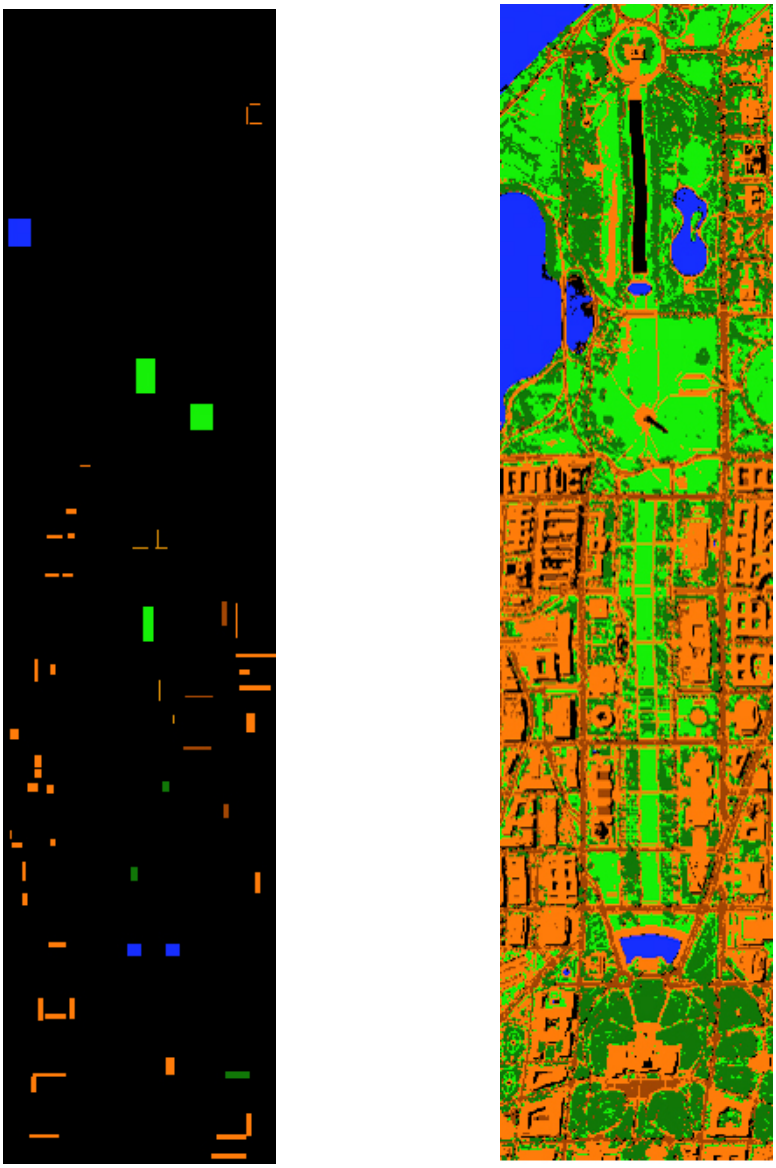
4.4 Face Data

We consider the face data of the Stanford Center for Image System Engineering. We choose an image with dimension 1372 ×1183. It has 148 spectral bands with wavelengths ranging from 415 to 950 nm in steps of 4 nm. The data set is available at `https://scien.stanford.edu/index.php/faces`.

The data does not have ground truth but we can observe the different classes using the RGB image shown in Figure 4(a). The testing is performed using the unsupervised algorithm and we vary the number of classes from three to five. The classification results are shown in Figure 4(b), (c), (d). We see that the algorithm accurately detects the different regions of the image.

4.5 Plume Video Data

We also consider the data set of hyperspectral video sequences recording the release of chemical plumes at the Dugway Proving Ground [14]. The data set we use here is the aa12 Victory data set from Algorithms for Threat Detection Data Repository, which is a video sequence documenting the release of a plume, and it has 329 frames in total. Each frame of the video sequence is a 3D image of dimension $128 \times 320 \times 129$, where the last dimension indicates the number of channels. Each channel depicts a particular frequency starting at 7,830 nm and ending at 11,700 nm with a spacing of 30



(a) Ground truth (b) Classification result

Figure 3: DC mall data.

(a) Ground truth. (b) Three classes. (c) Four classes. (d) Five classes.

Figure 4: Face data.

nm. The videos in this repository were captured by Johns Hopkins Applied Physics Lab using three long wave infrared (LWIR) spectrometers, each placed at a different location about 2 km away from the release of plume at an elevation of around 1300 feet. Frames were captured every five seconds.

The goal, of course, is to track the plume as it moves in the video sequence, but before the data is directly inputed into the algorithm, we perform some preprocessing of the raw data from the repository. We first convert the data to spectral emissivity values. Then, we locate the pixels with values greater than a certain threshold and replace their values with the mean values of their neighborhood.

Due to the temperature fluctuation during the day, there is a flicking inconsistency throughout the video and the pixel values vary from frame to frame. We show the different values from different frames of one fixed pixel in Figure 5. To eliminate the flicker between frames, the Midway equalization method is used in [35]. In this paper, we do not perform this preprocessing and the flicker problem does not affect the classification result.

This plume video dataset has been studied in several papers. The approach in [35] uses a combination of dimension reduction and histogram equalization to prepare the hyperspectral video data for segmentation. Principal Component Analysis (PCA) is used for dimension reduction of the hyperspectral video data, and a Midway method for histogram equalization is used to redistribute the intensity values in order to reduce flicker between frames. Then, the preprocessed data is classified using some traditional methods including K-means, spectral clustering, and the Ginzburg-Landau functional. In [71], a binary partition tree method is used to retrieve the real location and the extent of the plume. Moreover, the author of [68] proposes two ways to compute meaningful eigenvectors of the graph Laplacian. Other detection methods for hyperspectral plumes include [40] (MWIR) and [54] (HYDICE).

We select the 17^{th} frame to the 56^{th} frame, which contain most of the plume scenes, for our tests and we classify all the pixels simultaneously, not frame by frame. Note that when the number of frames is very large, the number of pixel values we are dealing with may exceed the maximum allowed value for a 32-bit signed binary integer (2,147,483,647) in many programming languages and an overflow can cause its value to wrap and become negative. For example, if we are dealing with 500 frames of the plume data, the overall number of pixel values is $500 \times 320 \times 128 \times 129 = 2.6e+09$, which is larger than 2,147,483,647. In this case, we need to process the video frames in batches. Instead of having just two classes of the plume and background, we choose to segment each frame in the video sequence into four classes: plume, sky, foreground, and mountain.

For the semi-supervised algorithm, since we do not have the ground truth of the plume video, we choose the "ground truth" by identifying the relevant eigenvectors and then thresholding, similarly to the procedure in [61]. We show the 2^{nd} to 5^{th} eigenvectors of frame 30 in aa12 Victory video set in Figure 6. We threshold the largest values in the 2^{nd} eigenvector to obtain the "known" labels for the sky and the smallest values to obtain "known" labels for the foreground. Similarly, we use the smallest values in the 3^{rd} eigenvector for the plume and the smallest values in the 5^{th} eigenvector for the mountain region. We select 2% of the "known" labels for each of the 4 classes. The selected fidelity regions are shown in Figure 7.

For the unsupervised algorithm, we set the number of classes to be five. This is due to the fact that there are a few noisy pixels in frame 22, which is the frame when the explosion of the plume started, and which would be classified as one class. The total number of the noisy pixels in this class is 68, which is not noticeable in the classification result.

The classification result (29^{th} frame to the 37^{th} frame) is shown in Figure 8 for the semi-supervised algorithm and Figure 9 for the unsupervised method. We see that the algorithms are able to accurately detect the plume as it moves in the video sequence. These results can be compared to that of spectral clustering, shown in Figure 10, in which the plume is separated into two classes. Therefore, spectral clustering does not do well in detecting the plume. We note that for the spectral clustering

experiment, we calculate the first 100 leading eigenvectors of the 40 frames using the Nyström extension method and then use the K-means algorithm on these 100 leading eigenvectors to find the 4 classes.

We also perform experiments using 329 frames. Since most of the frames are just background frames without any plume, the plume class contains a much smaller amount of pixels compared to the other three classes and becomes harder to detect. For the semi-supervised algorithm, in order to get similar results as that of the 40 frames, we need to increase the value of μ before the fidelity term. In a certain range, when μ is increased, the plume becomes thicker, and when μ is decreased, the plume becomes thinner. For the unsupervised algorithm, since we do not have any known labels, the plume detected is thinner than that of the result of the 40 frames.

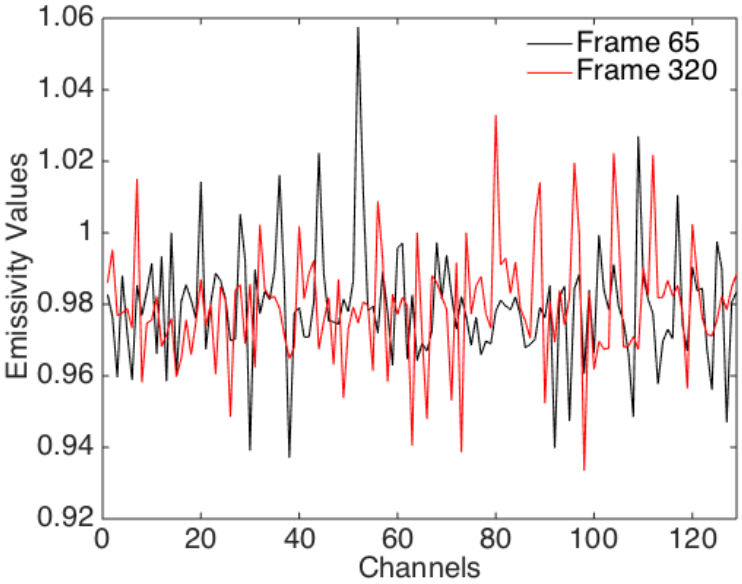


Figure 5: Emissivity value of one fixed pixel of frame 65 and frame 320.

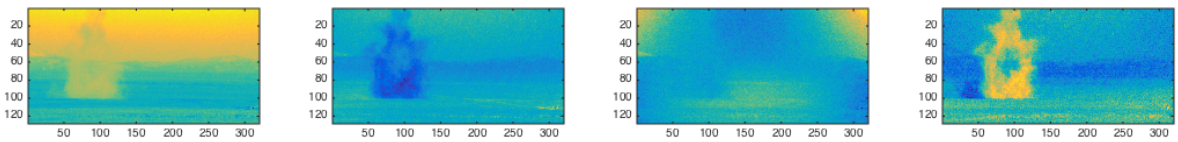


Figure 6: 2nd to 5th eigenvectors of the 30th frame.

4.6 Non-local Means Method for RGB image

We have applied the semi-supervised and unsupervised algorithms on several hyperspectral data sets and we also consider RGB images. RGB images have three bands on each pixel, which is much less than the number of bands of the hyperspectral images. We use the non-local means method to compute a feature vector for each pixel based on the RGB values of this pixel and that of its neighbors.

The non-local means method is widely used in image processing. Zhou and Schölkopf in their papers [84][83][82] formulated a theory of non-local operators that is related to the discrete graph Laplacian described in Section 2.1. Buades, Coll, and Morel applied this non-local theory to denoising algorithms in their work [16]. Osher and Gilboa proposed using non-local operators to define functionals involving the TV semi-norm for various image processing applications in their work [37, 36].

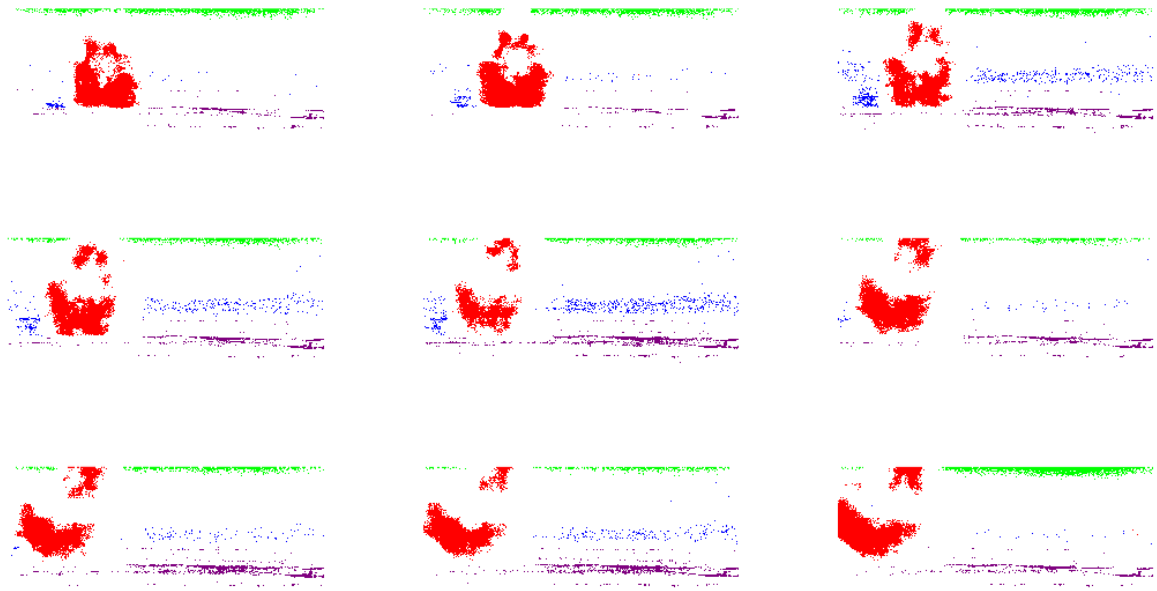


Figure 7: 8% of data selected to be the fidelity region by thresholding the eigenvectors, frames 29-37.

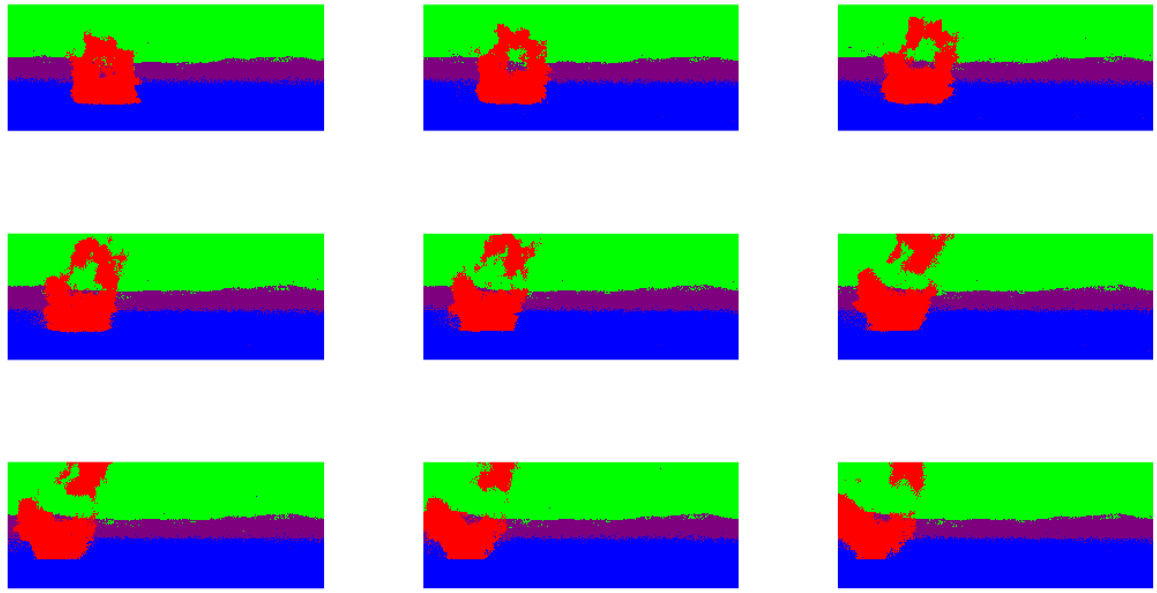


Figure 8: Classification result of the semi-supervised algorithm (frames 29-37 of the aa12 Victory video).

In our work, we use the non-local means method to compute the feature vector of each pixel. The procedure is described in Algorithm 4.

We apply a Gaussian kernel on a patch for each pixel i since we would like to give more importance to points closest to the center of the patch (pixel i). The points farther away from the main pixel

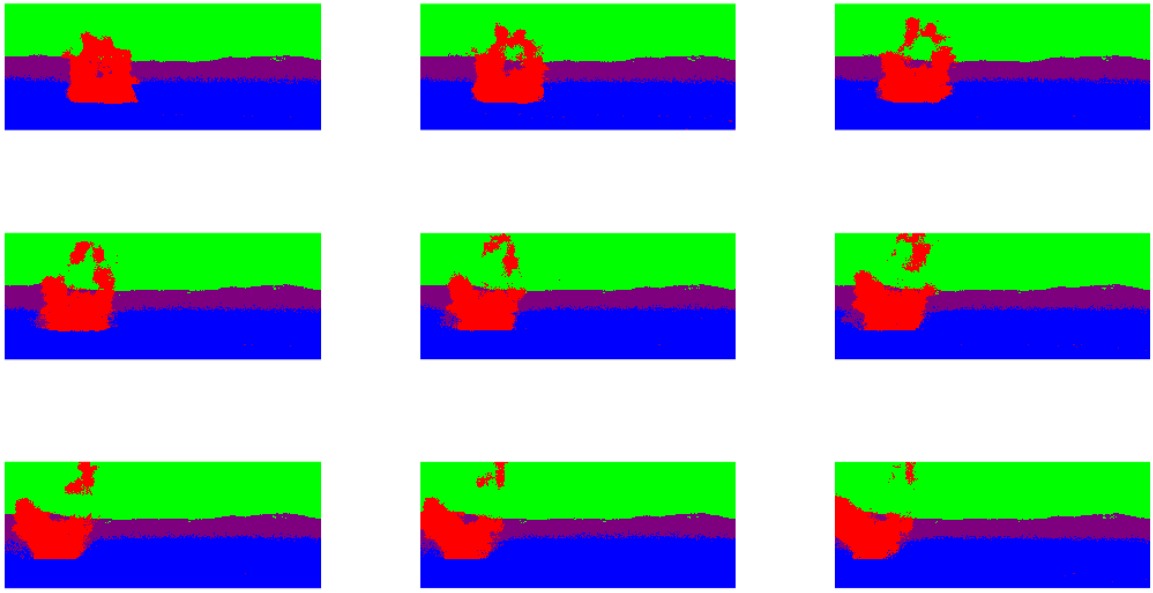


Figure 9: Classification result of the unsupervised algorithm (frames 29-37 of the aa12 Victory video).

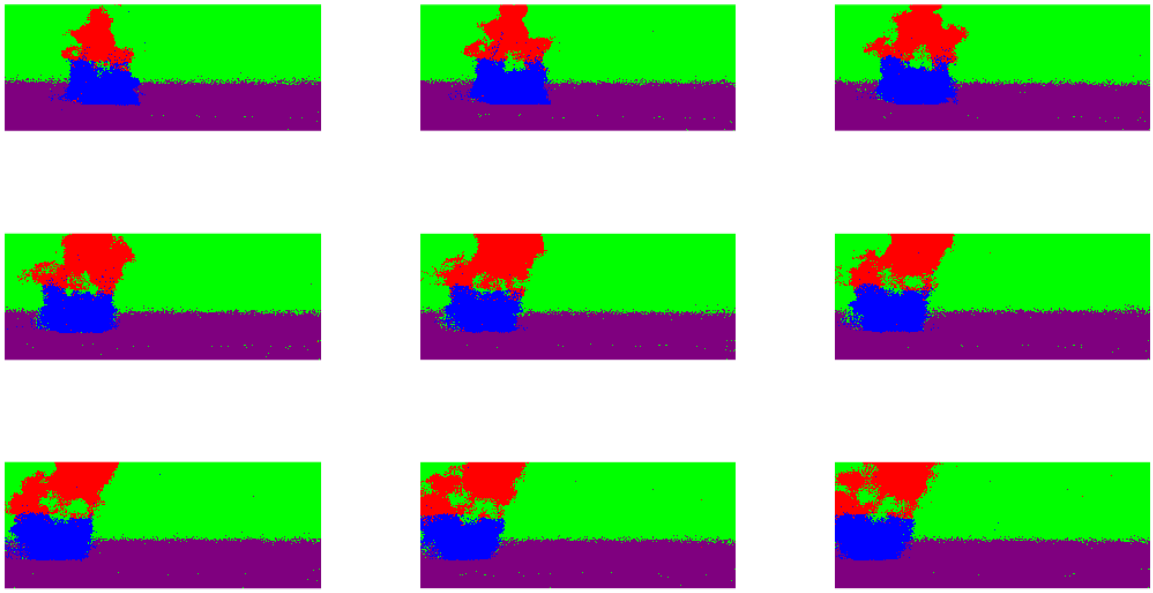


Figure 10: Classification result of spectral clustering with K-means (frames 29-37 of the aa12 Victory video).

should be weighted less than those closest to it.

We apply the semi-supervised and unsupervised algorithm on the RGB image of two cows and show the results in Figure 11. Figure 11(a) is the original image and there are four classes by observation: the grass, the river, one brown cow and one black cow. To use the semi-supervised

Algorithm 4: Non-local Means Method

Input: RGB image I, window size d.
Output: a $(m \times n)$ by $(2f+1)^2 \times 3$ feature matrix F.

1 Pad the image with mirror reflections of itself with a width d.
2 For the i^{th} pixel, make a $(2d+1)$ by $(2d+1)$ patch centered at pixel i.
3 For the $j^{th}, (j = 1; 2; 3)$ band, apply a Gaussian kernel on this patch and straighten it to a vector v_{ij}.
4 Concatenate the three vectors v_{i1}, v_{i2}, and v_{i3} together to form the feature vector v_i at pixel i.
5 Form the feature matrix F by letting each row of F be a feature vector of a pixel.

algorithm, we select some pixels with known labels to be the fidelity. The fidelity is shown in Figure 11(b) and the classification result of the semi-supervised algorithm is shown in Figure 11(c). The river, the grass and the brown cow are very accurately segmented, and some parts of the black cow get mixed with the river or the grass. The classification result of the unsupervised algorithm is shown in Figure 11(d). The river, the grass and the brown cow are also accurately segmented here. The black cow gets mixed with the river because the ratio of the RGB values of the black color (the black cow) and the grey color (the river) are the same. Since we use the cosine distance when building the weight matrix, the grey color and the black color are very close to each other and can be easily classified as one class. While when dealing with the hyperspectral images, the cosine distance is more meaningful because the same materials have same spectrum but may have different intensities.

In practice, we use $d = 2$ in our source code. The length of the feature vector of each pixel is $3 \times (2d+1)^2 = 75$, which is a reasonable dimension for the feature vector. We also consider a corner case. We consider an image with a dark region where the pixel values of all the pixels in this region are 0. Then when we apply the non-local means method on this image, the feature vector of pixel which is in the center of the dark region would be a zero vector. In this case, when we calculate the weight between this pixel with other pixels, using formula (1), we get an invalid value since the cosine distance between each vector to a zero vector is infinity. To avoid this kind of case, we add a small value ϵ (we use $\epsilon = 0.1$ in the codes) to all the pixel values in the RGB image to make sure all the weights are valid numbers.

4.7 Result Summary

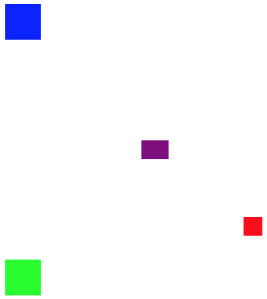
4.7.1 Accuracy Summary

We apply our the semi-supervised and unsupervised algorithms on various data sets. The accuracy summary is shown in Table 1. We show the overall accuracy of the data sets with ground truth. For the other data sets without ground truth (the face, the plume and the cow), the classification results are shown in figures in the previous subsections. The semi-supervised algorithm achieves high accuracy with very low portion of known labels for all data sets. We also compared the unsupervised algorithm with the spectral clustering with K-means. The spectral clustering with K-means achieves only 75.06% overall accuracy for the urban data set and it gives wrong classification results for the plume data set which is shown in Figure 10.

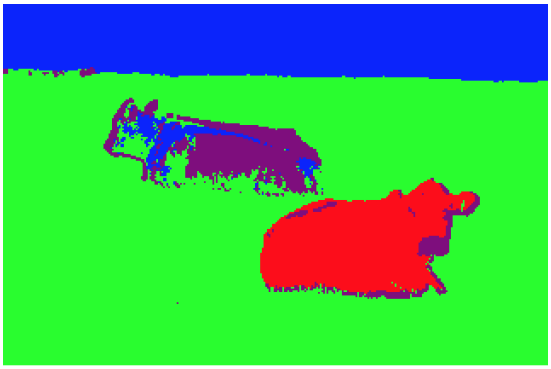
4.7.2 Run Time Summary

We apply our the semi-supervised and unsupervised algorithms on various hyperspectral data sets. The run time summary is shown in Table 2. We show the run time of Nyström extension method

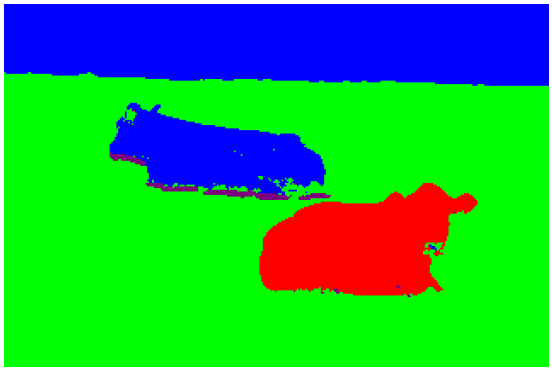
(a) the image of two cows on the grass near the river.



(b) pixels selected to have known labels for the semi-supervised algorithm.



(c) the classification result of the semi-supervised algorithm.



(d) the classification result of the unsupervised algorithm.

Figure 11: Classification result of the non-local method on the RGB Image of Two Cows.

	Urban(semi)	Urban(un)	KSC(semi)	DC(semi)
Number of classes	4	4	13	7
Percentage of fidelity	10%	NA	2.5%	1.73%
Overall Accuracy	93.48%	92.35%	80.37%	99.17%

Table 1: Accuracy summary of hyperspectral images.

and graph MBO methods tested on the IPOL server Purple. The run time of Nyström extension is greatly reduced by the parallelization which we will discuss in the following section. The run time of the MBO (both semi-supervised and unsupervised) method is highly dependent on the number of iterations. The MBO algorithm converges after around 10 iterations for the plume, face, and urban data sets. While for KSC and DC data set, it takes about 100 iterations to reach the convergence. The number of iterations it takes to get to convergence varies for different data sets and different numbers of classes.

5 Parallelization

In general, hyperspectral video sequences have hundreds of frames. For example, the Victory data set we use in this paper has 40 frames, each of which is a hyperspectral image with dimension $128 \times 320 \times 129$, where 129 is the number of channels of each pixel. Thus, the total number of pixels is 1,638,400! Therefore, in order to apply the algorithms to a large data set very efficiently,

	Urban(semi)	Urban(un)	KSC(semi)	DC(semi)	Face(un)	Plume(semi)	Plume(un)
Number of pixels	94,249	94,249	314,368	392,960	1,623,076	1,638,400	1,638,400
Number of bands	162	162	176	191	148	129	129
Number of classes	4	4	13	7	5	4	4
Nyström (serial)	5.72	5.72	19.97	26.44	93.92	112.00	112.00
Nyström (32 threads)	1.30	1.30	4.35	5.51	22.43	23.57	23.57
MBO	12.13	2.07	145.62	141.79	35.92	59.57	41.02

Table 2: Run time summary of hyperspectral images and videos (in seconds).

we parallelize our codes.

As a first step, we analyze the serial codes of the two algorithms, locate hotspots, and evaluate cache miss rates, potential for vectorization, locality improvement and other performance factors such as contention for shared resources. Here, the part that consumes the most run time is a big "For" loop building the matrix W_{XY} in the Nyström extension procedure. A simple parallelization for this problem is to use OpenMP directive-based parallelism. We apply that procedure to the code here, and present results on the IPOL testing server and a single node of a supercomputer at the National Energy Research Scientific Computing Center (NERSC).

First, we present parallelization scaling results on the IPOL test server named Purple. The IPOL server is comprised of Intel X7560 Nehalem, running at 2.27GHz and configured into a PowerEdge R910 server. With four sockets, and each one containing 8 cores, we have 32 cores of shared memory that are available. The scaling of the optimized code indicates almost ideal scaling, as shown in Figure 12.

Second, we describe parallelization scaling results on the NERSC machine Cori Phase I. The Cori Phase 1 system is based on Intel Haswell processors. It has two 2.3 GHz 16-core Haswell^{TM} processors per node. Each core has its own L1 and L2 caches, with 64 KB (32 KB instruction cache, 32 KB data) and 256 KB, respectively; there is also a 40-MB shared L3 cache per socket. According to Figure 12, the scaling of the optimized code is almost ideal as well.

In Figure 12, we also show the scaling result of the total time of the Nyström extension procedure; it includes both the serial and the parallelized part. Since we are using library BLAS in the serial part, the operations in the BLAS library are also parallelized when given more threads.

The results relating to the speedup factor are further shown in Table 3. According to the table, the speedup factor of the OpenMP part on IPOL server Purple shows an ideal scaling result! In fact, it runs 30.96 times faster when using 32 threads, while on Cori, it runs 23.74 times faster when using 32 cores; the difference of scaling is due to the different architecture of the machines. With regards to the speedup of the total time of Nyström extension procedure, Cori Phase I gives better results than Purple. The procedure runs 7.98 times faster when using 32 threads on Cori and 4.75 times faster on Purple.

6 Conclusion

We have presented one semi-supervised and one unsupervised graph-based classification algorithm and have applied these methods to hyperspectral video data, hyperspectral images and RGB images. The results show that these procedures can very accurately and efficiently segment a data set into a number of classes. To make the algorithms very efficient, we use the Nyström extension method to calculate the eigendecomposition of the graph Laplacian; in practice, only a small portion of the eigenvectors are needed to obtain good results. Moreover, we use OpenMP directive-based parallelism in our algorithms and observe strong (almost ideal) scaling behavior and very fast implementation

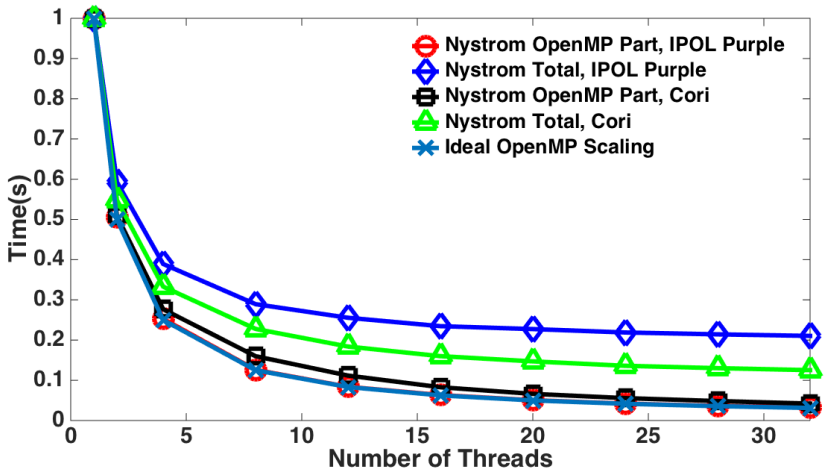


Figure 12: Normalized scaling results of the IPOL server Purple and NERSC machine Cori Phase I.

# of Threads	Nyström OpenMP(Purple)	Nyström Total(Purple)	Nyström OpenMP(Cori)	Nyström Total(Cori)
2	1.97	1.69	1.97	1.82
4	3.95	2.57	3.61	3.00
8	7,94	3.46	6.25	4.39
12	11.92	3.91	8.95	5.44
16	15.79	4.26	12.08	6.25
20	19.80	4.40	15.09	6.80
24	23.75	4.57	17.96	7.35
28	27.57	4.67	20.66	7.68
32	30.96	4.75	23.74	7.98

Table 3: Speedup factors of using multiple cores.

times. In fact, the entire process takes only 1 minute using 32 threads on a supercomputer for 40 frames of the plume video data!

Acknowledgements

The work was supported by the NSF grant DMS-1417674, UC Lab Fees Research grant 12-LR-236660, ONR grant N00014-16-1-2119 and NSF grant DMS-1118971, and U.S. Department of Energy under Contract No. DE-AC02-05CH11231. This research used resources of the National Energy Research Scientific Computing Center, a DOE Office of Science User Facility supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231. We would like to thank Dr. Huiyi Hu for her contribution for the unsupervised algorithm and we would like to thank Dr. Da Kuang for his suggestions on optimizing the codes. We also would like to thank Dr. Helen He for her help with the parallelization.

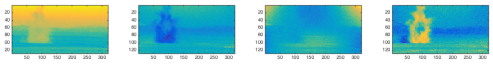
Image Credits


Urban Data: US Army Corps of Engineers, Ground truth: Feiyun Zhu


KSC: NASA AVIRIS

 DC: Larry L. Biehl

 Face: Stanford Center for Image System Engineering

 Plume: Johns Hopkins Applied Physics Lab

References

[1] S.M. Allen and J.W. Cahn, *A microscopic theory for antiphase boundary motion and its application to antiphase domain coarsening*, Acta Metallurgica, 27 (1979), pp. 1085–1095. https://doi.org/10.1016/0001-6160(79)90196-2.

[2] E. Bae and E. Merkurjev, *Convex variational methods on graphs for multiclass segmentation of high-dimensional data and point clouds*, Journal of Mathematical Imaging and Vision, 58 (2017), pp. 468–493. https://doi.org/10.1007/s10851-017-0713-9.

[3] M. Belkin, P. Niyogi, and V. Sindhwani, *Manifold regularization: A geometric framework for learning from labeled and unlabeled examples*, The Journal of Machine Learning Research, 7 (2006), pp. 2399–2434.

[4] A.L. Bertozzi and A. Flenner, *Diffuse interface models on graphs for classification of high dimensional data*, Multiscale Modeling & Simulation, 10 (2012), pp. 1090–1118. https://doi.org/10.1137/11083109X.

[5] ——, *Diffuse interface models on graphs for classification of high dimensional data*, SIAM Review, 58 (2016), pp. 293–328. https://doi.org/10.1137/16M1070426.

[6] A.L. Bertozzi and Y. van Gennip, *Gamma-convergence of graph Ginzburg-Landau functionals*, Advances in Differential Equations, 17 (2012), pp. 1115–1180.

[7] J.M. Bioucas-Dias and J.M.P. Nascimento, *Hyperspectral subspace identification*, IEEE Transactions on Geoscience and Remote Sensing, 46 (2008), pp. 2435–2445. https://doi.org/10.1109/TGRS.2008.918089.

[8] D. Böhning, *Multinomial logistic regression algorithm*, Annals of the Institute of Statistical Mathematics, 44 (1992), pp. 197–200. https://doi.org/10.1007/BF00048682.

[9] B.E. Boser, I.M. Guyon, and V.N. Vapnik, *A training algorithm for optimal margin classifiers*, in Proceedings of the Fifth Annual Workshop on Computational Learning Theory, ACM, 1992, pp. 144–152. https://doi.org/10.1145/130385.130401.

[10] Y. Boykov, O. Veksler, and R. Zabih, *Fast approximate energy minimization via graph cuts*, IEEE Transactions Pattern Analysis and Machine Intelligence, 23 (2001), pp. 1222–1239. https://doi.org/10.1109/34.969114.

[11] X. Bresson, T. Laurent, D. Uminsky, and J.H. von Brecht, *Convergence and energy landscape for Cheeger cut clustering*, in Advances in Neural Information Processing Systems, 2012, pp. 1385–1393. https://doi.org/10.21236/ada612749.

[12] ——, *An adaptive total variation algorithm for computing the balanced cut of a graph*, arXiv preprint arXiv:1302.2717, (2013).

[13] ——, *Multiclass total variation clustering*, in Advances in Neural Information Processing Systems, 2013, pp. 1421–1429.

[14] J.B. Broadwater, D. Limsui, and A.K. Carr, *A primer for chemical plume detection using LWIR sensors*, tech. report, National Security Technology Department, 2011.

[15] L. Bruzzone, M. Chi, and M. Marconcini, *A novel transductive SVM for semisupervised classification of remote-sensing images*, IEEE Transactions on Geoscience and Remote Sensing, 44 (2006), pp. 3363–3373. https://doi.org/10.1109/TGRS.2006.877950.

[16] A. Buades, B. Coll, and J-M Morel, *A non-local algorithm for image denoising*, in 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), vol. 2, IEEE, 2005, pp. 60–65. https://doi.org/10.1109/CVPR.2005.38.

[17] T. Bühler and M. Hein, *Spectral clustering based on the graph p-Laplacian*, in Proceedings of the 26th Annual International Conference on Machine Learning, ACM, 2009, pp. 81–88. https://doi.org/10.1145/1553374.1553385.

[18] G. Camps-Valls and L. Bruzzone, *Kernel-based methods for hyperspectral image classification*, IEEE Transactions on Geoscience and Remote Sensing, 43 (2005), pp. 1351–1362. https://doi.org/10.1109/TGRS.2005.846154.

[19] G. Camps-Valls, L. Gomez-Chova, J. Muñoz-Marí, J. Vila-Francés, and J. Calpe-Maravilla, *Composite kernels for hyperspectral image classification*, IEEE Geoscience and Remote Sensing Letters, 3 (2006), pp. 93–97. https://doi.org/10.1109/LGRS.2005.857031.

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

[21] O. Chapelle, B. Schölkopf, and A. Zien, eds., *Semi-Supervised Learning*, MIT Press, Cambridge, MA, 2006. http://doi.org/10.7551/mitpress/9780262033589.001.0001.

[22] Y. Chen, N.M. Nasrabadi, and T.D. Tran, *Hyperspectral image classification using dictionary-based sparse representation*, IEEE Transactions on Geoscience and Remote Sensing, 49 (2011), pp. 3973–3985. https://doi.org/10.1109/TGRS.2011.2129595.

[23] ——, *Sparse representation for target detection in hyperspectral imagery*, IEEE Journal of Selected Topics in Signal Processing, 5 (2011), pp. 629–640. https://doi.org/10.1109/JSTSP.2011.2113170.

[24] F. Chung, *Spectral graph theory*, vol. 92, American Mathematical Society, 1997. http://doi.org/10.1090/cbms/092.

[25] C. Couprie, L. Grady, L. Najman, and H. Talbot, *Power watershed: A unifying graph-based optimization framework*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 33 (2011), pp. 1384–1399. https://doi.org/10.1109/TPAMI.2010.200.

[26] A. Elmoataz, O. Lezoray, and S. Bougleux, *Nonlocal discrete regularization on weighted graphs: a framework for image and manifold processing*, IEEE Transactions on Image Processing, 17 (2008), pp. 1047–1060. https://doi.org/10.1109/TIP.2008.924284.

[27] S. Esedoglu, S.J. Ruuth, and R. Tsai, *Threshold dynamics for high order geometric motions*, Interfaces and Free Boundaries, 10 (2008), pp. 263–282. https://doi.org/10.4171/IFB/189.

[28] S. Esedoglu and Y-H. Richard Tsai, *Threshold dynamics for the piecewise constant Mumford-Shah functional*, Journal of Computational Physics, 211 (2006), pp. 367–384. https://doi.org/10.1016/j.jcp.2005.05.027.

[29] M.D. Farrell Jr and R.M. Mersereau, *On the impact of PCA dimension reduction for hyperspectral detection of difficult targets*, IEEE Geoscience and Remote Sensing Letters, 2 (2005), pp. 192–195. https://doi.org/10.1109/LGRS.2005.846011.

[30] X. Feng and A. Prohl, *Numerical analysis of the Allen-Cahn equation and approximation for mean curvature flows*, Numerische Mathematik, 94 (2003), pp. 33–65. https://doi.org/10.1007/s00211-002-0413-1.

[31] Ch. Fowlkes, S. Belongie, F. Chung, and J. Malik, *Spectral grouping using the Nystrom method*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 26 (2004), pp. 214–225. https://doi.org/10.1109/TPAMI.2004.1262185.

[32] S. Gao, I.W-H. Tsang, and L-T. Chia, *Kernel sparse representation for image classification and face recognition*, in European Conference on Computer Vision, Springer, 2010, pp. 1–14. http://doi.org/10.1007/978-3-642-15561-1_1.

[33] C. Garcia-Cardona, A. Flenner, and A.G. Percus, *Multiclass diffuse interface models for semi-supervised learning on graphs*, in Proceedings of the 2th International Conference on Pattern Recognition Applications and Methods, SciTePress, 2013. https://doi.org/10.1007/978-3-319-12610-4_8.

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

[35] T. Gerhart, J. Sunu, L. Lieu, E. Merkurjev, J-M. Chang, J. Gilles, and A.L. Bertozzi, *Detection and tracking of gas plumes in LWIR hyperspectral video sequence data*, in SPIE Defense, Security, and Sensing, International Society for Optics and Photonics, 2013, pp. 87430J–87430J. http://doi.org/10.1117/12.2015155.

[36] G. Gilboa and S. Osher, *Nonlocal linear image regularization and supervised segmentation*, Multiscale Modeling & Simulation, 6 (2007), pp. 595–630. https://doi.org/10.1137/060669358.

[37] ——, *Nonlocal operators with applications to image processing*, Multiscale Modeling & Simulation, 7 (2008), pp. 1005–1028. https://doi.org/10.1137/070698592.

[38] J.A. Gualtieri and R.F. Cromp, *Support vector machines for hyperspectral remote sensing classification*, in 27th AIPR Workshop: Advances in Computer-Assisted Recognition, 1999, pp. 221–232.

[39] M. Hein and S. Setzer, *Beyond spectral clustering - tight relaxations of balanced graph cuts*, in Advances in Neural Information Processing Systems 24, J. Shawe-Taylor, R.S. Zemel, P. Bartlett, F.C.N. Pereira, and K.Q. Weinberger, eds., 2011, pp. 2366–2374.

[40] M. Hinnrichs, J.O. Jensen, and G. McAnally, *Handheld hyperspectral imager for stand-off detection of chemical and biological aerosols*, in Optical Technologies for Industrial, Environmental, and Biological Sensing, International Society for Optics and Photonics, 2004, pp. 67–78. https://doi.org/10.1117/12.519163.

[41] H. Hu, T. Laurent, M.A. Porter, and A.L. Bertozzi, *A method based on total variation for network modularity optimization using the mbo scheme*, SIAM Journal on Applied Mathematics, 73 (2013), pp. 2224–2246. https://doi.org/10.1137/130917387.

[42] H. Hu, J. Sunu, and A.L. Bertozzi, *Multi-class graph Mumford-Shah model for plume detection using the MBO scheme*, in Energy Minimization Methods in Computer Vision and Pattern Recognition, Springer, 2015, pp. 209–222. https://doi.org/10.1007/978-3-319-14612-6_16.

[43] C-C. Hung, S. Kulkarni, and B-C. Kuo, *A new weighted fuzzy c-means clustering algorithm for remotely sensed image classification*, IEEE Journal of Selected Topics in Signal Processing, 5 (2011), pp. 543–553. https://doi.org/10.1109/JSTSP.2010.2096797.

[44] L.O. Jimenez and D.A. Landgrebe, *Hyperspectral data analysis and supervised feature reduction via projection pursuit*, IEEE Transactions on Geoscience and Remote Sensing, 37 (1999), pp. 2653–2667. https://doi.org/10.1109/36.803413.

[45] S. Kaewpijit, J. Le Moigne, and T. El-Ghazawi, *Automatic reduction of hyperspectral imagery using wavelet spectral analysis*, IEEE Transactions on Geoscience and Remote Sensing, 41 (2003), pp. 863–871. http://doi.org/10.1109/TGRS.2003.810712.

[46] R.V. Kohn and P. Sternberg, *Local minimisers and singular perturbations*, Proceedings of the Royal Society of Edinburgh: Section A Mathematics, 111 (1989), pp. 69–84. https://doi.org/10.1017/S0308210500025026.

[47] B. Krishnapuram, L. Carin, M.A.T. Figueiredo, and A.J. Hartemink, *Sparse multinomial logistic regression: Fast algorithms and generalization bounds*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 27 (2005), pp. 957–968. https://doi.org/10.1109/TPAMI.2005.127.

[48] H. Kwon and N.M. Nasrabadi, *A comparative analysis of kernel subspace target detectors for hyperspectral imagery*, EURASIP Journal on Advances in Signal Processing, 2007 (2006), pp. 1–13.

[49] J. Li, J.M. Bioucas-Dias, and A. Plaza, *Semisupervised hyperspectral image segmentation using multinomial logistic regression with active learning*, IEEE Transactions on Geoscience and Remote Sensing, 48 (2010), pp. 4085–4098. https://doi.org/10.1109/TGRS.2010.2060550.

[50] ——, *Spectral–spatial hyperspectral image segmentation using subspace multinomial logistic regression and Markov random fields*, IEEE Transactions on Geoscience and Remote Sensing, 50 (2012), pp. 809–823. https://doi.org/10.1109/TGRS.2011.2162649.

[51] X. Luo and A.L. Bertozzi, *Convergence analysis of the graph Allen-Cahn scheme*. submitted, 2016.

[52] L. Ma, M.M. Crawford, and J. Tian, *Local manifold learning-based k-nearest-neighbor for hyperspectral image classification*, IEEE Transactions on Geoscience and Remote Sensing, 48 (2010), pp. 4099–4109. https://doi.org/10.1109/TGRS.2010.2055876.

[53] J. MacQueen, *Some methods for classification and analysis of multivariate observations*, in Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, vol. 1, Oakland, CA, USA., 1967, pp. 281–297.

[54] D. Manolakis, T. Siracusa, and G. Shaw, *Adaptive matched subspace detectors for hyperspectral imaging applications*, in IEEE International Conference on Acoustics, Speech, and Signal Processing, vol. 5, 2001, pp. 3153–3156. https://doi.org/10.1109/ICASSP.2001.940327.

[55] F. Melgani and L. Bruzzone, *Classification of hyperspectral remote sensing images with support vector machines*, IEEE Transactions on Geoscience and Remote Sensing, 42 (2004), pp. 1778–1790. https://doi.org/10.1109/TGRS.2004.831865.

[56] Z. Meng, A. Koniges, Y.(Helen) He, S. Willianms, T. Kurth, B. Cook, J. Deslippe, and A.L. Bertozzi, *OpenMP parallelization and optimization of graph-based machine learning algorithms*, in Maruyama N., de Supinski B., Wahib M. (eds) OpenMP: Memory, Devices, and Tasks. IWOMP. Lecture Notes in Computer Science, vol 9903. Springer, 2016. https://doi.org/10.1007/978-3-319-45550-1_2.

[57] E. Merkurjev, E. Bae, A.L. Bertozzi, and X-C. Tai, *Global binary optimization on graphs for classification of high-dimensional data*, Journal of Mathematical Imaging and Vision, 52 (2015), pp. 414–435. https://doi.org/10.1007/s10851-015-0567-y.

[58] E. Merkurjev, A. Bertozzi, X. Yan, and K. Lerman, *Modified Cheeger and Ratio cut methods using the Ginzburg-Landau functional for classification of high-dimensional data*, Inverse Problems, 33 (2017). https://doi.org/10.1088/1361-6420/33/7/074003.

[59] E. Merkurjev, C. Garcia-Cardona, A.L. Bertozzi, A. Flenner, and A.G. Percus, *Diffuse interface methods for multiclass segmentation of high-dimensional data*, Applied Mathematics Letters, 33 (2014), pp. 29–34. https://doi.org/10.1016/j.aml.2014.02.008.

[60] E. Merkurjev, T. Kostic, and A.L. Bertozzi, *An MBO scheme on graphs for classification and image processing*, SIAM Journal on Imaging Sciences, 6 (2013), pp. 1903–1930. https://doi.org/10.1137/120886935.

[61] E. Merkurjev, J. Sunu, and A.L. Bertozzi, *Graph MBO method for multiclass segmentation of hyperspectral stand-off detection video*, in IEEE International Conference on Image Processing (ICIP), 2014, pp. 689–693. https://doi.org/10.1109/ICIP.2014.7025138.

[62] B. Merriman, J.K. Bence, and S. Osher, *Diffusion generated motion by mean curvature*, AMS Selected Lectures in Mathematics Series: Computational Crystal Growers Workshop, 8966 (1992), pp. 73–83.

[63] B. Merriman, J.K. Bence, and S.J. Osher, *Motion of multiple junctions: A level set approach*, Journal of Computational Physics, 112 (1994), pp. 334–363. https://doi.org/10.1006/jcph.1994.1105.

[64] D. Mumford and J. Shah, *Optimal approximations by piecewise smooth functions and associated variational problems*, Communications on Pure and Applied Mathematics, 42 (1989), pp. 577–685. https://doi.org/10.1002/cpa.3160420503.

[65] S. Niazmardi, S. Homayouni, and A. Safari, *An improved FCM algorithm based on the SVD for unsupervised hyperspectral data classification*, IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, 6 (2013), pp. 831–839. https://doi.org/10.1109/JSTARS.2013.2244851.

[66] H. Shi, Y. Shen, and Z. Liu, *Hyperspectral bands reduction based on rough sets and fuzzy c-means clustering*, in Proceedings of the 20th IEEE Instrumentation and Measurement Technology Conference, vol. 2, 2003, pp. 1053–1056.

[67] J. Shi and J. Malik, *Normalized cuts and image segmentation*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 22 (2000), pp. 888–905. https://doi.org/10.1109/34.868688.

[68] J. Sunu, J-M. Chang, and A.L. Bertozzi, *Simultaneous spectral analysis of multiple video sequence data for LWIR gas plumes*, in SPIE conference on Defense, Security, and Sensing, 2014. https://doi.org/10.1117/12.2050149.

[69] A. Szlam and X. Bresson, *Total variation and Cheeger cuts*, in Proceedings of the 27th International Conference on Machine Learning, Johannes Fürnkranz and Thorsten Joachims, eds., Haifa, I., 2010, Omnipress, pp. 1039–1046.

[70] Y. Tarabalka, J.A. Benediktsson, J. Chanussot, and J.C. Tilton, *Multiple spectral–spatial classification approach for hyperspectral data*, IEEE Transactions on Geoscience and Remote Sensing, 48 (2010), pp. 4122–4132. https://doi.org/10.1109/TGRS.2010.2062526.

[71] G. Tochon, J. Chanussot, J. Gilles, M. Dalla Mura, J-M. Chang, and A. Bertozzi, *Gas plume detection and tracking in hyperspectral video sequences using binary partition trees*, in IEEE Workshop on Hyperspectral Image and Signal Processing: Evolution in Remote Sensing (WHISPERS 2014), 2014.

[72] Y. Van Gennip, N. Guillen, B. Osting, and A.L. Bertozzi, *Mean curvature, threshold dynamics, and phase field theory on finite graphs*, Milan Journal of Mathematics, 82 (2014), pp. 3–65. https://doi.org/10.1007/s00032-014-0216-8.

[73] V. Vapnik, *The nature of statistical learning theory*, Springer Science & Business Media, 2013. http://doi.org/10.1007/978-1-4757-3264-1.

[74] 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 (2002), pp. 271–293. https://doi.org/10.1023/A:1020874308076.

[75] P. Vincent and Y. Bengio, *Kernel matching pursuit*, Machine Learning, 48 (2002), pp. 165–187.

[76] U. Von Luxburg, *A tutorial on spectral clustering*, Statistics and Computing, 17 (2007), pp. 395–416. https://doi.org/10.1007/s11222-007-9033-z.

[77] J. Wang and C-I. Chang, *Independent component analysis-based dimensionality reduction with applications in hyperspectral image analysis*, IEEE Transactions on Geoscience and Remote Sensing, 44 (2006), pp. 1586–1600. https://doi.org/10.1109/TGRS.2005.863297.

[78] X-T. Yuan, X. Liu, and S. Yan, *Visual classification with multitask joint sparse representation*, IEEE Transactions on Image Processing, 21 (2012), pp. 4349–4360. https://doi.org/10.1109/TIP.2012.2205006.

[79] L. Zhang, Y. Zhong, B. Huang, J. Gong, and P. Li, *Dimensionality reduction based on clonal selection for hyperspectral imagery*, IEEE Transactions on Geoscience and Remote Sensing, 45 (2007), pp. 4172–4186. https://doi.org/10.1109/TGRS.2007.905311.

[80] X. Zhang, M. Burger, X. Bresson, and S. Osher, *Bregmanized nonlocal regularization for deconvolution and sparse reconstruction*, SIAM Journal on Imaging Sciences, 3 (2010), pp. 253–276. https://doi.org/10.1137/090746379.

[81] D. Zhou, O. Bousquet, T.L. Lal, J. Weston, and B. Schölkopf, *Learning with local and global consistency*, in Advances in Neural Information Processing Systems 16, Sebastian Thrun, Lawrence K. Saul, and Bernhard Schölkopf, eds., MIT Press, Cambridge, MA, 2004, pp. 321–328.

[82] D. Zhou, T. Hofmann, and B. Schölkopf, *Semi-supervised learning on directed graphs*, in Advances in neural information processing systems, 2004, pp. 1633–1640.

[83] D. Zhou, J. Huang, and B. Schölkopf, *Learning from labeled and unlabeled data on a directed graph*, in Proceedings of the 22nd international conference on Machine learning, ACM, 2005, pp. 1036–1043. http://doi.org/10.1145/1102351.1102482.

[84] D. Zhou and B. Schölkopf, *Regularization on discrete spaces*, in Joint Pattern Recognition Symposium, Springer, 2005, pp. 361–368. https://doi.org/10.1007/11550518_45.

[85] F. Zhu, Y. Wang, S. Xiang, B. Fan, and C. Pan, *Structured sparse method for hyperspectral unmixing*, ISPRS Journal of Photogrammetry and Remote Sensing, 88 (2014), pp. 101–118. http://doi.org/10.1016/j.isprsjprs.2013.11.014.