Published in Image Processing On Line on YYYY–MM–DD.
Submitted on YYYY–MM–DD, accepted on YYYY–MM–DD.
ISSN 2105–1232 © YYYY IPOL & the authors CC–BY–NC–SA
This article is available online with supplementary materials,
software, datasets and online demo at
http://www.ipol.im/

Hyperspectral Video Analysis Using Graph Clustering Methods

Zhaoyi Meng [13], Ekaterina Merkurjev[2], Alice Koniges[3], Andrea L Bertozzi[4]

[1] Department of Mathematics, UCLA (mzhy@ucla.edu)
[4] Department of Mathematics, UCLA (bertozzi@math.ucla.edu)

PREPRINT April 20, 2016

Abstract

Perhaps the most challenging imaging modality to analyze in terms of the vast size of the data are videos taken from hyperspectral cameras. We consider an example involving standoff detection of a gas plume involving long wave infrared spectral data with 128 bands. Rather than using PCA or a similar dimension reduction method we treat this as a "big data" classification problem and simultaneously process all pixels in the entire video using novel new graph clustering techniques. Computation of the entire similarity graph is prohibitive for such data so we use the Nyström extension to randomly sample the graph and compute a modest number of eigenfunctions of the graph Laplacian. A very small part of the spectrum allows for spectral clustering of the data. However with a larger but still modest number of eigenfunctions we can solve a graph-cut based semi-supervised or unsupervised machine learning problem to sort the pixels into classes. We discuss the challenges of running such code on both desktops and supercomputers. With only 12 cores, we are able to process 329 frames of hyperspectral video in three minutes.

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

1 Introduction

Multi-class classification is one of the fundamental problems in machine learning and computer vision. In this paper, we outline methods to classify data sets, such that the similarity between nodes in one class is much larger than the similarity between nodes of different class. 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.

A traditional way of solving the hyperspectral image classification problem is by dimension reduction and the most common approach is principal component analysis (PCA). Signal subspace identification is another way which enables a correct dimensionality reduction. A new minimum mean square error-based approach to infer the signal subspace in hyperspectral imagery is introduced in [3]. Fuzzy C-Means (FCM) algorithm is an essential tool to find proper clusters and can be used for hyperspectral image classification. A new weighted fuzzy C-Means clustering algorithm was proposed in [16] to improve the performance of FCM. FCM can be also enhanced by the Support Vector Domain Description [24].

In this work, we introduce two novel graph-based classification algorithms. The first one is semi-supervised [20], which requires some known labels as input, and the other one is unsupervised [15]. Both algorithms are derived from PDE methods and are modified to apply to discrete data sets. The semi-supervised algorithm uses the Ginzburg-Laudau (GL) functional as described in [12, 11]. The functional is formulated in a graphical setting, and is then minimized. The unsupervised algorithm is derived from the Chan-Vese method [5, 27]. We modify the Chan-Vese model on the graph and formulate the classification problem as an energy minimization problem. Both methods apply the MBO scheme, which is a well-established PDE method for evolving an interface by mean curvature [21, 22]. It can be generalized to the graph setting. By alternating between diffusion and thresholding techniques, it can efficiently solve the minimization problems. For computational efficiency, we also implement the Nyström Extension method for calculating eigenvalues and eigenvectors, which uses a small sample of the nodes to create an approximation in order to overcome the computational cost of constructing the whole weight matrix [9, 1, 10]. The two algorithms are very fast and give accurate classification results.

For hyperspectral images, the channel of values of each pixel is usually of hundreds of dimensions. Thus, hyperspectral video data sets can be very large. This motivated us to develop parallel implementations and optimizations of these two new algorithms. 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 super fast implementations.

The paper is organized as follows. In Section 2, we present the graph model, the details of the two algorithms and the Nyström Extension method. In Section 3 we show some numerical results of the hypersectral data sets. In Section 4, we describe our parallelization technique.

2 Graph-based Classification Algorithms

2.1 Graphical Representation

The two classification algorithms are based on the graph setting [7, 28]. Suppose $G = (V, E)$ is an undirected graph, where $V$and E are sets of nodes and edges respectively. We represent the data as nodes on a weighted graph and the similarity (weight) between two nodes is 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 two nodes and τ is a parameter to be determined. More details about how to choose τ can be found in [2]. For hyperspectral image, the feature vector is the vector of channels of each pixel and we use the cosine norm to compare two feature vectors.

The degree of a node $i \in V$ is defined as $d_i = \sum_{j \in V} w(i,j)$. 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 purpose, 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 is the random walk Laplacian given by:

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

It is related to the discrete Markov process and more details about these two normalizations can be found in [7] and [28].

2.2 Semi-supervised Algorithm

One type of a classification algorithm is that of semi-supervised learning. In such a situation, the fidelity (a small amount of "ground truth") is known and the rest needs to be classified according to the categories of the known data [20].

We approach the classification problem using the ideas of energy minimization. In [2], the classification problem is approached by minimizing the Ginzburg-Laudau (GL) functional in graph form with a fidelity term. In [19], the authors propose a MBO scheme to solve this problem.

We work with an assignment matrix u. The i^{th} row u_i indicates the class label of node i. The k^{th} component of u_i is the probability the i^{th} node belongs to class k. The minimization problem is formulated as following:

$$E(u) = \epsilon u \cdot L_s u + \frac{1}{\epsilon}\int W(u)dx + \frac{1}{2}C\lambda(x)(u-\hat{u})^2. \tag{4}$$

The first two terms are the graph form of the GL functional where L_s is the symmetric Laplacian and $W(u)$ is a double-well potential. For example $W(u) = (u^2-1)^2$ in a binary partitioning and multi-well potential in $\hat{n}$ dimensions, where $\hat{n}$ is the number of classes. The last term is the regular L^2 fit with some constant C, and $\lambda(x)$ takes the value of 1 on fidelity terms, and 0 otherwise. $\hat{u}$ is the initial data with randomly chosen labels for non-fidelity data points and the "ground truth" for the fidelity points.

Minimizing $E(u)$ by gradient decent method, one obtains:

$$\frac{\partial u}{\partial t} = -\epsilon L_s u - \frac{1}{\epsilon}W'(u) - C\lambda(x)(u-\hat{u}). \tag{5}$$

This is the Allen-Cahn equation with fidelity terms with Δu replaced by a graph operator $-L_s$. When $\epsilon \to 0$, the solution to the Allen-Cahn equation can be approximated by motion by mean curvature [21], which the original MBO scheme is very successful in simulating.

In [19], the authors propose a MBO scheme to solve this equation. The method consists of alternating between the following two steps:

- Step 1: Heat equation with forcing term:

$$\frac{u^{n+\frac{1}{2}} - u^n}{dt} = -L_s u^{n+\frac{1}{2}} - C\lambda(x)(u^n - \hat{u}), \tag{6}$$

- Step 2: Thresholding:

$$u_i^{n+1} = e_r, \tag{7}$$

  where e_r is the vertex in the simplex closest to the projection of $u_i^{n+\frac{1}{2}}$ onto the simplex using [6].

The computation can be further simplified by using the eigendecomposition of L_s, which is:

$$L_s = X\Lambda X^T. \tag{8}$$

We approximate X by a truncated matrix retaining only a small number of the leading eigenvectors. If we write $u^n = Xa^n$, $\mu(u^n - \hat{u}) = Xd^n$ and equate coefficients, we can formulate step 1 in the MBO scheme as solving for the coefficients a_k^{n+1}:

$$a_k^{n+1} = \frac{a_k^n - dt d_k^n}{1 + dt\lambda_k}, \tag{9}$$

where λ_k is the k^{th} eigenvalue of L_s.

In computation, we compute the norm of difference between the label matrix u of two consecutive iterations and stop the iteration when the norm is below a threshold. We find the largest value in each row u_i and assign the corresponding index as the class label of the i^{th} data point.

2.3 Unsupervised Algorithm

Sometimes, there is no knowledge of the class of any part of the data set. Thus, we reformulate an unsupervised algorithm using similar methodology based on the Mumford-Shah model [23] which is a famous model used for multi-class segmentation problem. One simplified version of the Mumford-Shah model tailored for images is the piecewise constant model:

$$E^{MS}(\Phi, \{c_r\}_{r=1}^{\hat{n}}) = |\Phi| + \lambda \sum_{r=1}^{\hat{n}} \int_{\Omega_r} (f - c_r)^2, \tag{10}$$

where the contour Φ segments an image region Ω into $\hat{n}$ disjoint sub-regions Ω_r and $|\Phi|$ is the length of the contour. Here, f is the observed image data and $\{c_r\}_{r=1}^{\hat{n}}$ is a set of constant values.

The graph version of the multi-class piecewise constant Mumford-Shah energy is [15]:

$$MS(u, \{c_r\}_{r=1}^{\hat{n}}) = \frac{1}{2}|u|_{TV} + \lambda \sum_{r=1}^{\hat{n}} < ||f - c_r||^2, u_r >, \tag{11}$$

where u is the class assignment matrix, i.e. if $u_r(n_i) = 1$ for some r, then node i belongs to the rth class and $\sum_{r=1}^{\hat{n}} u_r(n_i) = 1$. The length of the contour is estimate by the total variation (TV) of the assignment matrix u. $||f - c_r||^2$ denotes an $N \times 1$ vector $(||f(n_1) - c_r||^2, ..., ||f(n_N) - c_r||^2)^T$ and the n_i $(i = 1, ...N)$ are the N pixels.

For the purpose of segmentation, we need to solve the minimization problem:

$$\min_{u \in B, \{c_r\}_{r=1}^{\hat{n}}} MS(u, \{c_r\}_{r=1}^{\hat{n}}). \tag{12}$$

This problem is essentially equivalent to the K-means method when λ goes to $+\infty$.

The optimal solution of equation (12) is a stationary point and the partial derivative of the MS functional with respect to c is zero at the optimal solution. Thus we have:

$$c_r = \frac{< f, u_r >}{\sum_{i=1}^{N} u_r(n_i)}. \tag{13}$$

It is known that the GL functional converges to the TV seminorm [17], thus we can instead minimize the following funcional:

$$E(u, c_r) = \epsilon u \cdot L_s u + \frac{1}{\epsilon} \int W(u) dx + \lambda \sum_{r=1}^{\hat{n}} < ||f - c_r||^2, u_r > . \tag{14}$$

Similar to the procedure in section 2.2, we can use the gradient decent method and get the following equation:

$$\frac{\partial u}{\partial t} = -\epsilon L_s u - \frac{1}{\epsilon}W^{'}(u) - \lambda(||f - c_1^n||^2, ..., ||f - c_{\hat{n}}^n||^2). \tag{15}$$

So we can use the MBO scheme in section 2.2 for this minimization problem. More detailed derivation can be found in [8]. The problem can be solved by alternating between the following three steps:

- Step 1: Compute
$$u^{n+\frac{1}{2}} = e^{-\tau L_s}u^n - \tau\lambda(||f - c_1^n||^2, ..., ||f - c_{\hat{n}}^n||^2), \tag{16}$$
- Step 2: Threshold
$$u^{n+1}(n_i) = e_r, r = argmax_k u_k^{n+\frac{1}{2}}(n_i) \tag{17}$$
for all $i \in \{1, 2, ..., N\}$, where e_r is the rth standard basis in $\mathbb{R}^{\hat{n}}$
- Step 3: Update c
$$c_r^{n+1} = \frac{< f, u_r^{n+1} >}{< 1, u_r^{n+1} >}. \tag{18}$$

Let $\{\phi_m\}_{m=1}^M$ be the first M orthogonal leading eigenvectors of L, $\{\xi_m\}_{m=1}^M$ the corresponding eigenvalues, and write f as $f^k = \sum_{m=1}^M \phi_m a^m$. Then step 1 of the algorithm can be approximately computed as:

$$u^{n+\frac{1}{2}} = \sum_{m=1}^{M} e^{-\tau\xi_m}\phi_m a^m - \tau\lambda(||f - c_1^k||^2, ..., ||f - c_{\hat{n}}^k||^2). \tag{19}$$

If we normalize the data set in advance, i.e. normalize the feature vector of each pixel to have norm 1, it's equivalent to using the cosine norm instead of euclidean norm. For hyperspectral image, since the data points are in high dimensional space, cosine distance gives better results.

In computation, we initialize the assignment matrix u by random labels and stop the iteration using the same criteria with the one in section 2.2. We assign the index of the largest value of the i^{th} row to be the class label of the i^{th} data point.

2.4 Nyström Extension Algorithm

In both the supervised and unsupervised algorithm, we use the spectral method to accelerate the computation. By only calculating several leading eigenvectors and eigenvalues of the graph Laplacian matrix and project all vectors onto this sub-eigenspace, the iteration step become simply updating coefficients.

An approximation to the eigendecomposition of the graph Laplacian matrix can be computed very efficiently by the Nyström method [9]. This is achieved by calculating an eigendecomposition on 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 for accuracy.

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.

Based on the definition in formula (2), if ξ is the eigenvalue of $\hat{W} = D^{-1/2}WD^{-1/2}$, then $1 - \xi$ is the eigenvalue of L_s. To calculate the eigenvalues and eigenvectors of $\hat{W}$, we need to normalize the matrix above.

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

$$\begin{aligned} d_X &= W_{XX}1_L + W_{XY}1_{N-L}, \\ d_Y &= W_{YX}1_L + (W_{YX}W_{XX}^{-1}W_{XY})1_{N-L}. \end{aligned} \tag{20}$$

Let $A./B$ denote componentwise 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 by

$$\begin{aligned} \hat{W}_{XX} &= W_{XX}./(s_X s_X^T) \\ \hat{W}_{XY} &= W_{XY}./(s_X s_Y^T) \end{aligned} \tag{21}$$

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

It is shown in [2] that if we have the eigendecomposition of two small matrices

$$\hat{W}_{XX} = B_X \Gamma B_X^T \tag{22}$$

and

$$\hat{W}_{XX} + \hat{W}_{XX}^{-1/2}\hat{W}_{XY}\hat{W}_{YX}\hat{W}_{XX}^{-1/2} = A^T \Xi A, \tag{23}$$

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

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

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

3 Numerical Results

3.1 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 by the Hyperspectral Digital Imagery Collection Experiment (HYDICE) in October 1995, whose location is an urban area at Copperas Cove, TX, U.S.. The set consists of an image with dimensions 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 which contains 4 classes with endmembers corresponding to asphalt, grass, tree and roof, respectively [29].

We apply both the semi-supervised and unsupervised algorithms on 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 to Figure 6. Asphalt, grass, trees and roof are labeled in blue, red, green and yellow, respectively. The accuracy of the semi-supervised algorithm is 93.48% and the accuracy of the unsupervised algorithm is 92.35%, while the spectral clustering with K-means has the accuracy of only 75.06%.

3.2 Plume Video Data

We consider the data set of hyperspectral video sequences recording the release of chemical plumes at the Dugway Proving Ground [4]. The data set we use here is the aa12 Victory data set from Algorithms for Threat Detection Data Repository and it has 329 frames in total. Each frame of the video sequence is an 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 with 11,700 nm with a spacing of 30 nm. The videos in this repository were captured by 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. Each frame was captured every five seconds. There are other detection methods for the hyperspectral plume such as [14] (MWIR) and [18] (HYDICE).

We did some preprocessing of the raw data from the repository. We first convert the data in to their spectral emissivity values. Then we locate the pixels with value greater than a certain threshold from their neighborhood and replace them with the mean values of the neighborhood.

Due to the temperature fluctuation during the day, there's flicking inconsistency through 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 7. To eliminate the flicker between frames, the Midway equalization method is used in [13]. In this paper, we do not need 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 [13] uses a combination of dimension reduction and histogram equalization to prepare the hyperspectral videos for segmentation. Principal Component Analysis(PCA) is used for dimension reduction of the hyperspectral video, 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 can be classified using some traditional methods including K-means, spectral clustering, and the Ginzburg-Landau functional. In [26], a binary partition tree method is used to retrieve the real location and extent of the plume. The author of [25] proposes two ways to provide meaningful eigenvectors of the graph Laplacian which can be used in spectral clustering methods. We use the Nyström extension in this work.

We select the 17th frame to the 56th frame, which contain most of the plume scenes, for our tests and we classify all the pixels simutaneously, not frame by frame. One thing to keep in mind is when the number of frames is very large, the number of values we are dealing with may exceed the max 32-bit signed binary integer in compution and become negative. In this case, we need to process the video frames in batches. We choose 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 thresholding, similarly to the procedure in [20]. We show the second to fifth eigenvectors of frame 30 in aa12 Victory video set in Figure 8. We threshold the largest values in the 2nd eigenvector to get the "known" labels for the sky and the smallest values to get "known" labels for the ground. Similarly, we use the smallest values in the 3rd eigenvector for the plume and the smallest values in the 5th eigenvector for the mountain region. We select 2% of the "known" labels for each of the 4 classes. The selected fidelity are shown in Figure 9.

For the unsupervised algorithm, we set the number of classes to be five. This is due to the fact that there are few noisy pixels in frame 22, the frame when the explosion started, which would be classified as one class. The total number of the noisy pixels in this class are 68, which is invisible in classification result.

The classification results of the 29th frame to the 37th frame are shown in Figure 10 for the semi-supervised and Figure 11 for the unsupervised. We also compare the results of our two algorithms with spectral clustering. For spectral clustering, we calculate the first 100 leading eigenvectors of the 40 frames using Nyström extension and then use the K-means algorithm on these 100 leading eigenvectors to find the 4 classes. The result of spectral clustering is shown in Figure 12, in which the plume is separated into two classes.

We also tried our algorithm on 329 frames. Most of the frames are just background without any plume. Thus the plume class contains a much smaller amount of pixels compared to the other three classes and become 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 parameter before the fidelity term. In a certain range, when the parameter goes higher, the plume is thicker and when the parameter goes lower, the plume is thinner. For the unsupervised algorithm, since we do not have any force from the known labels, the plume detected is thinner than the result of the 40 frames.

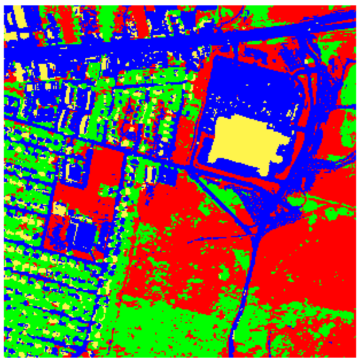
Figure 1: Ground truth

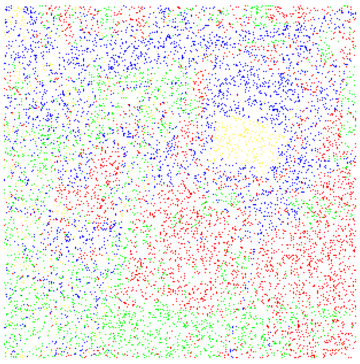
Figure 2: 10% randomly selected ground truth

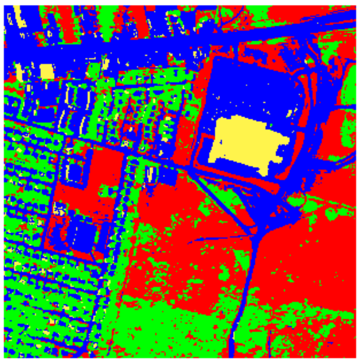
Figure 3: Semi-supervised altorithm results, accuracy 93.48%

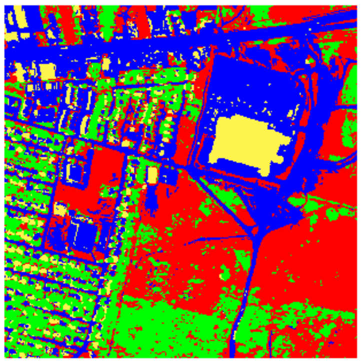
Figure 4: Unsupervised algorithm results, accuracy 92.35%

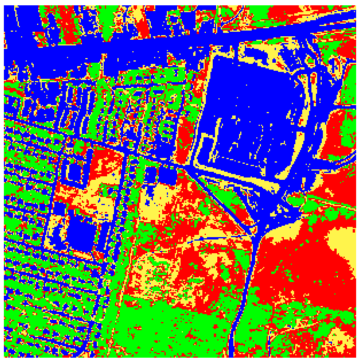
Figure 5: Spectral clustering results, accuracy 75.06%

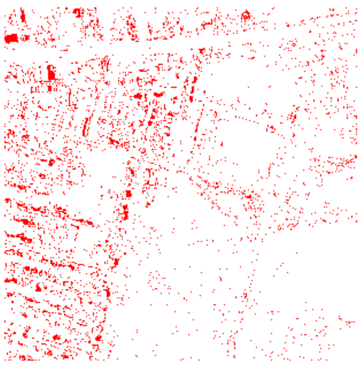
Figure 6: Error of the semi-supervised algorithm

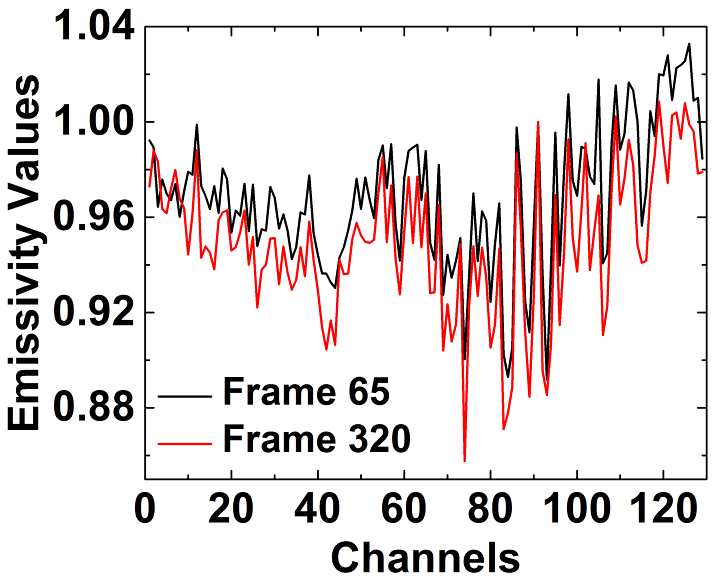

Figure 7: Emissivity value of one fixed pixel of frame 65 and frame 320

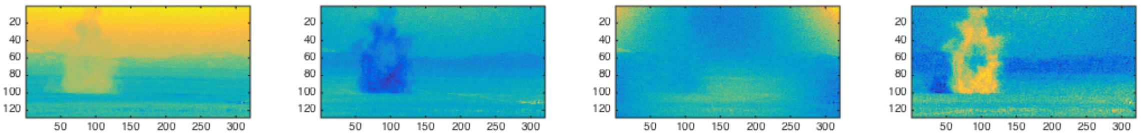

Figure 8: 2nd to 5th eigenvectors for the 30th frame

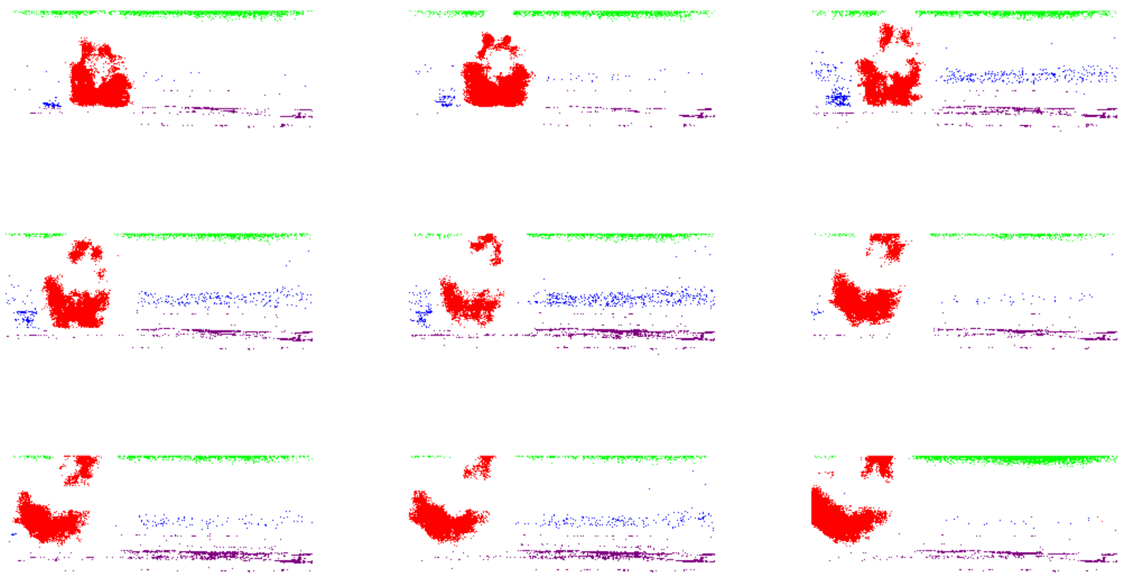

Figure 9: 8% of data selected as known labels by thresholding the eigenvectors, frame 29-37

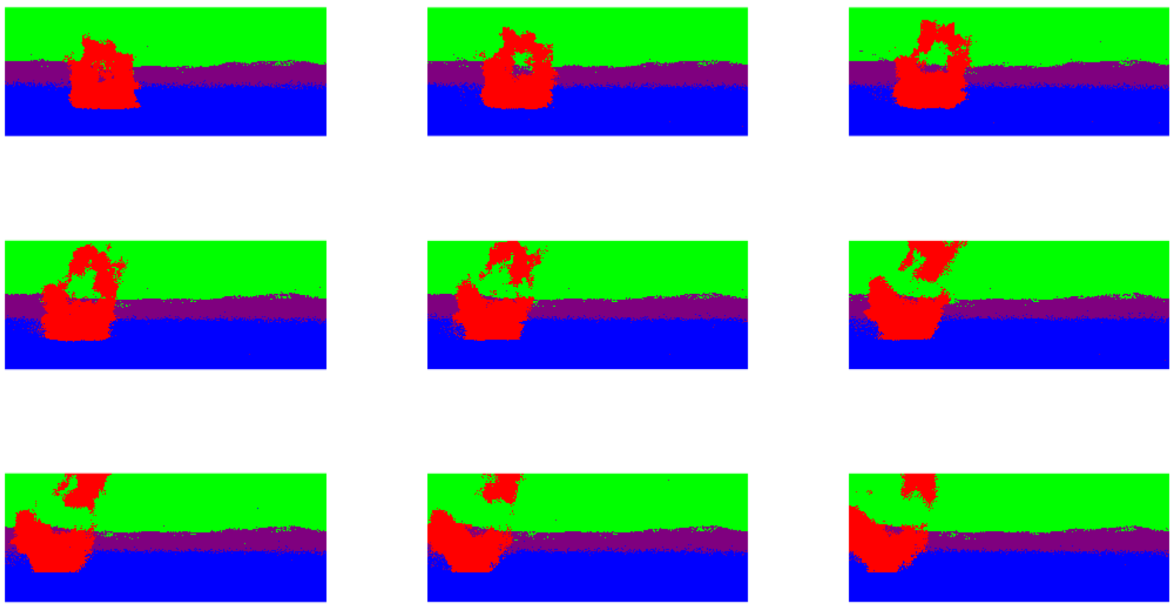

Figure 10: Classification result of frames 29-37 of the aa12 Victory video by the semi-supervised algorithm

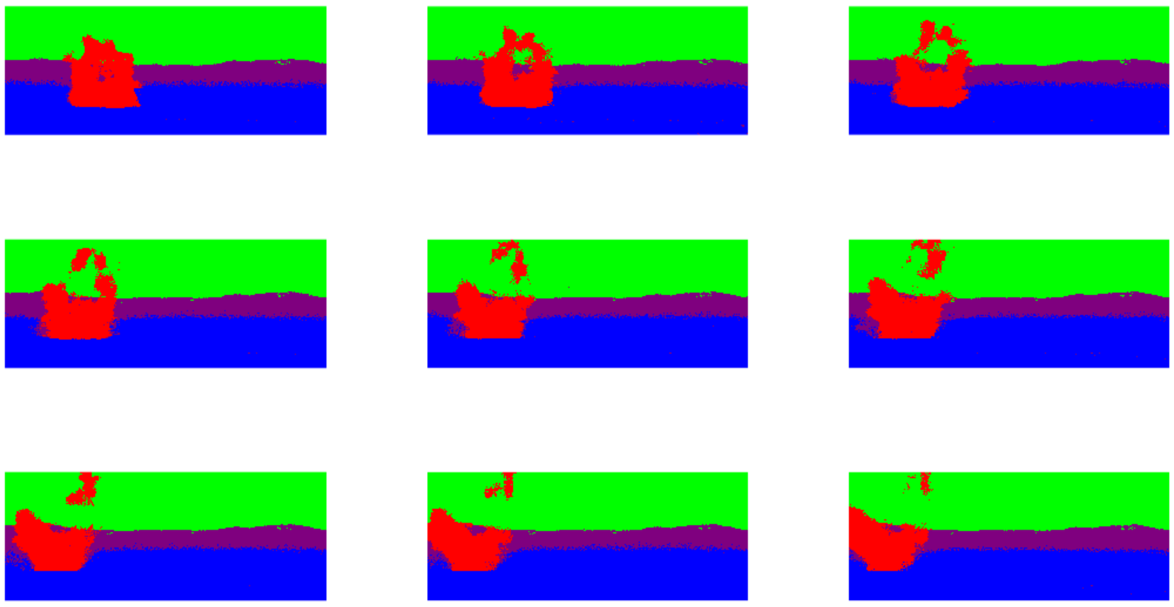

Figure 11: Classification result of frames 29-37 of the aa12 Victory video by the unsupervised algorithm

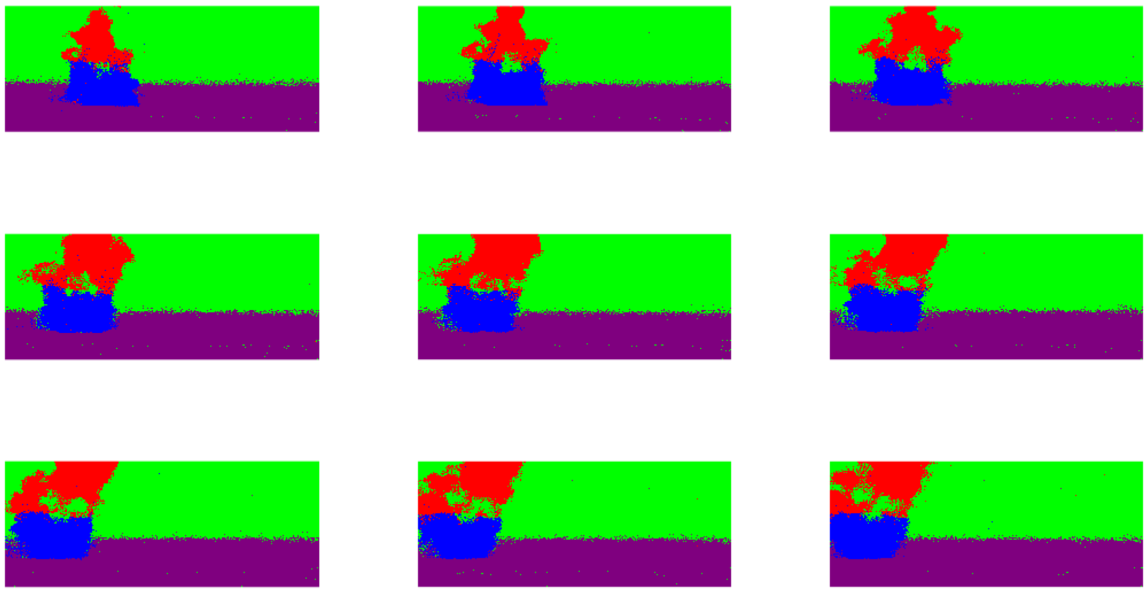

Figure 12: Classification result of frames 29-37 of the aa12 Victory video by the spectral clustering with K-means

4 Parallelization

We now present the parallelization of these two classification algorithms. We discuss challenges and give our solutions. OpenMP is used as the main parallel language and the I/O process is also optimized.

4.1 Analysis of Serial Code

The two algorithms give accurate and fast classification results when running in serial mode on a small data set. Since we use matrix operations to a large extent in the algorithms, we use the LAPACK and BLAS libraries in the codes.(version?,compilers)

We use the 40 frames of the plume video as the testing data set. Each frame is a hyperspectral image with dimension $128 \times 320 \times 129$, where 129 is the dimension of the channel of each pixel. The total number of pixels is 1638400. We use the Cori Phase I machine at NERSC for testing algorithms. Cori is a (some information from nersc)

In order to apply the algorithms on a large data set efficiently, we require parallelization. As a first step, we analyze the serial codes for the two algorithms with Intel VTune Amplifier. VTune is a tool for advanced performance optimization and we use it to locate hotspots and evaluate cache miss rates, potential for vectorization, locality improvement and other performance factors such as contention for shared resources. Here, the major part which consumes most of the run time is a big "For" loop in the Nyström Extension procedure and another is a "For loop" in the unsupervised algorithm. Since OpenMP is a suitable parallelization language for parallelizing For loops and the current data set fits easily on one node of Cori, we choose OpenMP for the parallelization.

4.2 Parallelization of the Nyström Extension

The Nyström Extension is an efficient procedure for calculating eigenvectors and eigenvalues for large dense matrix. As discussed in section 2.3, by using Nyström Extension, we can get the eigendecomposition without calculating the whole weight matrix. Instead, we only need to calculate the two sub-matrices W_{XX} and W_{XY}. The dimension of W_{XX} and W_{XY} are $L \times L$ and $L \times N$ respectively, where L is the number of samples and N is the total number of pixels. The computational cost of constructing W_{XX} is very small since L is usually very small. In this paper, we use $L = 100$. The computational cost of building W_{XY} can be large since it increases linearly with the number of pixels N increases. According the the hotspots collection of VTune, we find that the procedure of building W_{XY} consumes 90% of the total time of the Nyström Extension algorithm. Since we use the double For loop for calculating W_{XY}, we use OpenMP to parallelize the huge For loop.

4.3 Parallelization of the unsupervised algorithm

We also use VTune to collect the hotspots for the semi-supervised and unsupervised algorithms. For the semi-supervised algorithm, it is very fast and usually converges in less than 10 iterations. The computational time is far less than the part of the Nyström Extension. So there's no need to parallelize it. For the unsupervised algorithm, we detect a huge For loop by using VTune. Similarly to the parallelization of the Nyström Extension, we use OpenMP to parallelize this For loop.

4.4 I/O optimization

Since we are dealing with very large data set, we need to be careful about the data storage and data I/O. For 35 frames of video data, the data size is 2.96 GB in txt file and 1.48GB in binary form.

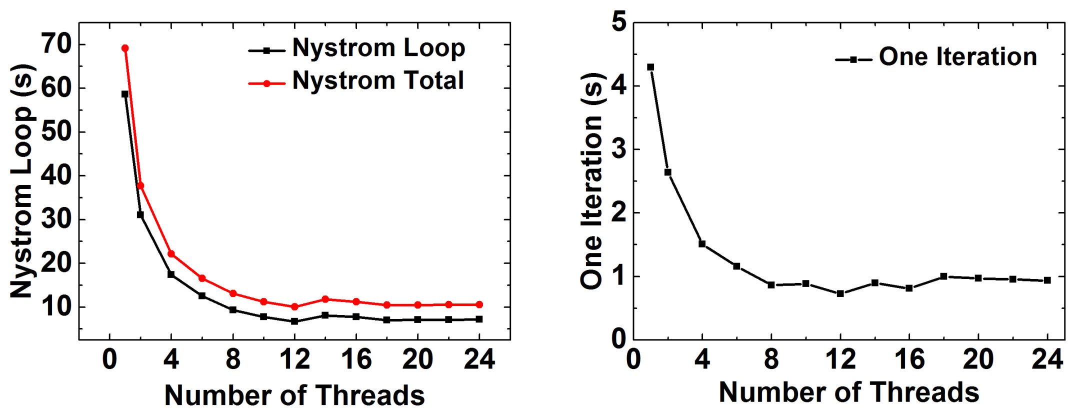

Figure 13: Scaling results on Edison: the left figure shows the run time of the Nyström extension method and one major loop in the Nyström extension verses different number of threads; the right figure shows the run time of one iteration in the unsupervised algorithm verses different number of threads.

The binary form saves nearly half the size of the data. Also, we choose to store data in binary files to ensure fast file I/O. For 35 frames, the input process is 4.63 seconds for binary form and 139.49 seconds for txt form.

4.5 Scaling Results

We parallelize the computationally intense loops with OpenMP and study the scaling properties. For 35 frames of the plume video, the data size is 1.48GB in binary form and the run time of the unsupervised algorithm is 150 seconds on average. We show the speedup factor for different number of threads in table 4.5.

Number of Threads	Nyström loop	Nyström Total	One iter in MBO(unsupervised)
2	1.92	1.66	1.63
4	3.41	2.83	2.85
6	4.69	3.74	3.71
8	6.43	4.78	4.97
10	7.71	5.61	4.87
12	8.94	6.25	5.91
14	7.28	5.28	4.79
16	7.71	5.60	5.29
18	8.33	5.87	4.32
20	8.22	5.91	4.44
22	8.32	5.88	4.50
24	8.36	5.94	4.61

Table 1: Speedup factors.

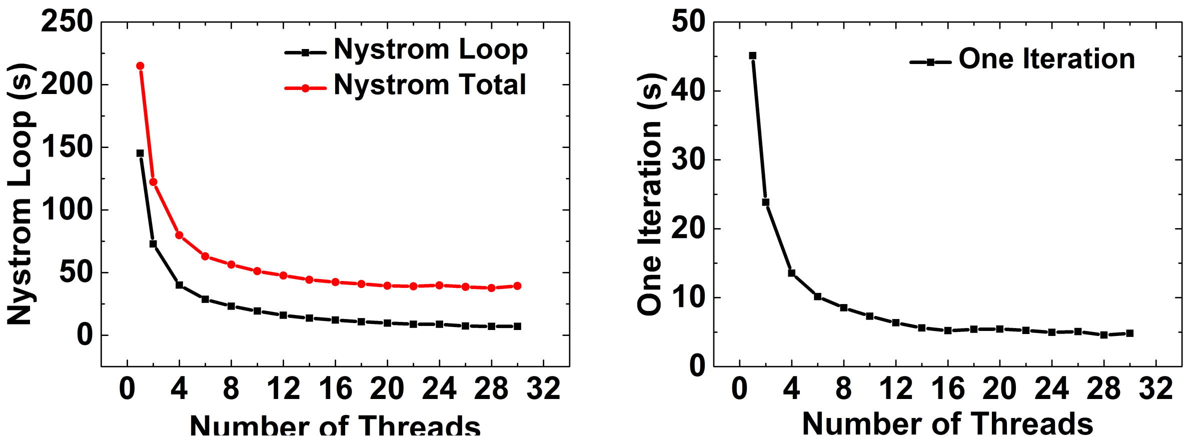

Figure 14: Scaling results on Cori: the left figure shows the run time of the Nyström extension method and one major loop in the Nyström extension verses different number of threads; the right figure shows the run time of one iteration in the unsupervised algorithm verses different number of threads.

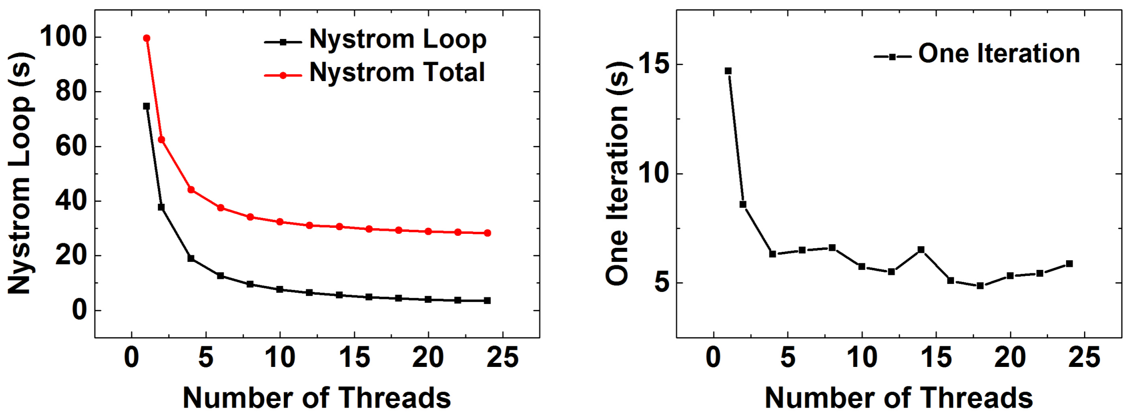

Figure 15: Scaling results on IPOL server: the left figure shows the run time of the Nyström extension method and one major loop in the Nyström extension verses different number of threads; the right figure shows the run time of one iteration in the unsupervised algorithm verses different number of threads.

5 Conclusion

6 Acknowledgements

The work was supported by the NSF grant DMS-1417674, UC Lab Fees Research grant 12-LR-236660, ONR grant N000141210838 and NSF grant DMS-1118971. We would like to thank Dr. Huiyi Hu for her contribution for the unsupervised algorithm. We also would like to thank NERSC for the help of the supercomputers Edison and Cori. Alice's acknowledgement.

References

[1] Serge Belongie, Charless Fowlkes, Fan Chung, and Jitendra Malik, *Spectral partitioning with indefinite kernels using the Nystrom extension*, in Computer VisionECCV 2002, Springer, 2002, pp. 531–542.

[2] Andrea L Bertozzi and Arjuna Flenner, *Diffuse interface models on graphs for classification of high dimensional data*, Multiscale Modeling & Simulation, 10 (2012), pp. 1090–1118.

[3] José M Bioucas-Dias and José MP Nascimento, *Hyperspectral subspace identification*, Geoscience and Remote Sensing, IEEE Transactions on, 46 (2008), pp. 2435–2445.

[4] Joshua B Broadwater, Diane Limsui, and Alison K Carr, *A primer for chemical plume detection using LWIR sensors*, tech. report, National Security Technology Department, 2011.

[5] Tony F Chan and Luminita A Vese, *Active contours without edges*, Image processing, IEEE transactions on, 10 (2001), pp. 266–277.

[6] Yunmei Chen and Xiaojing Ye, *Projection onto a simplex*, arXiv preprint arXiv:1101.6081, (2011).

[7] Fan Chung, *Spectral graph theory*, vol. 92, American Mathematical Soc., 1997.

[8] Selim Esedoglu and Yen-Hsi Richard Tsai, *Threshold dynamics for the piecewise constant Mumford-Shah functional*, Journal of Computational Physics, 211 (2006), pp. 367–384.

[9] Charless Fowlkes, Serge Belongie, Fan Chung, and Jitendra Malik, *Spectral grouping using the Nystrom method*, Pattern Analysis and Machine Intelligence, IEEE Transactions on, 26 (2004), pp. 214–225.

[10] Charless Fowlkes, Serge Belongie, and Jitendra Malik, *Efficient spatiotemporal grouping using the Nystrom method*, in Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on, vol. 1, IEEE, 2001, pp. I–231.

[11] Cristina Garcia-Cardona, Arjuna Flenner, and Allon 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.

[12] Cristina Garcia-Cardona, Ekaterina Merkurjev, Andrea L Bertozzi, Arjuna Flenner, and Allon G Percus, *Multiclass data segmentation using diffuse interface methods on graphs*, Pattern Analysis and Machine Intelligence, IEEE Transactions on, 36 (2014), pp. 1600–1613.

[13] Torin Gerhart, Justin Sunu, Lauren Lieu, Ekaterina Merkurjev, Jen-Mei Chang, Jérôme Gilles, and Andrea 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.

[14] Michele Hinnrichs, James O Jensen, and Gerard McAnally, *Handheld hyperspectral imager for standoff 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.

[15] Huiyi Hu, Justin Sunu, and Andrea 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.

[16] Chih-Cheng Hung, Sameer Kulkarni, and Bor-Chen Kuo, *A new weighted fuzzy c-means clustering algorithm for remotely sensed image classification*, Selected Topics in Signal Processing, IEEE Journal of, 5 (2011), pp. 543–553.

[17] Robert V Kohn and Peter Sternberg, *Local minimisers and singular perturbations*, Proceedings of the Royal Society of Edinburgh: Section A Mathematics, 111 (1989), pp. 69–84.

[18] Dimitris Manolakis, Thristina Siracusa, and Gary Shaw, *Adaptive matched subspace detectors for hyperspectral imaging applications*, in Acoustics, Speech, and Signal Processing, 2001. Proceedings.(ICASSP'01). 2001 IEEE International Conference on, vol. 5, IEEE, 2001, pp. 3153–3156.

[19] Ekaterina Merkurjev, Tijana Kostic, and Andrea L Bertozzi, *An MBO scheme on graphs for classification and image processing*, SIAM Journal on Imaging Sciences, 6 (2013), pp. 1903–1930.

[20] Ekaterina Merkurjev, Justin Sunu, and Andrea L Bertozzi, *Graph MBO method for multiclass segmentation of hyperspectral stand-off detection video*, in Image Processing (ICIP), 2014 IEEE International Conference on, IEEE, 2014, pp. 689–693.

[21] Barry Merriman, James Kenyard Bence, and Stanley Osher, *Diffusion generated motion by mean curvature*, Department of Mathematics, University of California, Los Angeles, 1992.

[22] Barry Merriman, James K Bence, and Stanley J Osher, *Motion of multiple junctions: A level set approach*, Journal of Computational Physics, 112 (1994), pp. 334–363.

[23] David Mumford and Jayant Shah, *Optimal approximations by piecewise smooth functions and associated variational problems*, Communications on pure and applied mathematics, 42 (1989), pp. 577–685.

[24] Saeid Niazmardi, Saeid Homayouni, and Abdolreza Safari, *An improved fcm algorithm based on the svdd for unsupervised hyperspectral data classification*, Selected Topics in Applied Earth Observations and Remote Sensing, IEEE Journal of, 6 (2013), pp. 831–839.

[25] Justin Sunu, Jen-Mei Chang, and Andrea L Bertozzi, *Simultaneous spectral analysis of multiple video sequence data for LWIR gas plumes*, in SPIE Defense+ Security, International Society for Optics and Photonics, 2014, pp. 90880T–90880T.

[26] Guillaume Tochon, Jocelyn Chanussot, Jérôme Gilles, Mauro Dalla Mura, Jen-Mei Chang, and Andrea 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.

[27] Luminita A Vese and Tony 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.

[28] Ulrike Von Luxburg, *A tutorial on spectral clustering*, Statistics and computing, 17 (2007), pp. 395–416.

[29] Feiyun Zhu, Ying Wang, Shiming Xiang, Bin Fan, and Chunhong Pan, *Structured sparse method for hyperspectral unmixing*, ISPRS Journal of Photogrammetry and Remote Sensing, 88 (2014), pp. 101–118.