UNIVERSITY OF CALIFORNIA

Los Angeles

High Performance Computing and Real Time Software for High Dimensional Data Classification

A dissertation submitted in partial satisfaction

of the requirements for the degree

Doctor of Philosophy in Mathematics

by

Zhaoyi Meng

2018

ABSTRACT OF THE DISSERTATION

High Performance Computing and Real Time Software for High Dimensional Data Classification

by

Zhaoyi Meng

Doctor of Philosophy in Mathematics

University of California, Los Angeles, 2018

Professor Andrea Bertozzi, Chair

We present high performance computing and real time software for high dimensional data classification. We investigate the OpenMP parallelization and optimization of two graph-based data classification algorithms. The new algorithms are based on graph and PDE solution techniques and provide significant accuracy and performance advantages over traditional data classification algorithm. We use OpenMP as the parallelization language to parallelize the most time-consuming parts, which is the Nyström extension eigensolver. The Nyström extension calculates eigenvalue/eigenvectors of the graph Laplacian and this is a self-contained module that can be used in conjunction with other graph-Laplacian based methods such as spectral clustering. We then optimize the OpenMP implementations and detail the performance on traditional supercomputer nodes (in our case a Cray XC30), and test the optimization steps on emerging testbed systems based on Intel's Knights Corner and Landing processors. We show both performance improvement and strong scaling behavior. A large number of optimization techniques and analyses are necessary before the algorithm reaches almost ideal scaling. We further develop the parallelized real time software. The software is published on Image Processing On Line (IPOL) and uses IPOL's server. It provides real time computation for image classification.

We apply the parallelized classification algorithms to various problems. The first one is hyperspectral image classification, a challenging modality due to the dimension of pixels which can range from hundreds to over a thousand frequencies depending on the sensor. Our methods use the full dimensionality of the data and 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 the computation of the full graph. The parallelized Nyström extension method randomly samples the graph to construct a low rank approximation of the graph Laplacian. With at most a few hundreds eigenfunctions, we can implement the clustering method to solve variational problems for graph-cut-based semi-supervised and unsupervised classification problems. The second application is ego-motion classification of body-worn videos. Portable cameras record dynamic first-person video footage and these videos contain information on the motion of the individual on whom the camera is mounted, defined as ego. We address the task of discovering ego-motion from the video itself, without other external calibration information. We investigate the use of similarity transformations between successive video frames to extract signals reflecting ego-motions and their frequencies. We use the parallelized graph-based unsupervised and semi-supervised learning algorithms to segment the video frames into different ego-motion categories. We show very accurate results on both choreographed test videos and ego-motion videos provided by the Los Angeles Police Department.

The dissertation of Zhaoyi Meng is approved.

Stanley J Osher

Luminita Aura Vese

Paul Jeffrey Brantingham

Andrea Bertozzi, Committee Chair

University of California, Los Angeles
2018

To my family and friends

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

First, I would like to thank my advisor Andrea Bertozzi. She has provided me with invaluable advice, and guided me in all aspects of my graduate student career. Without her support and collaboration, this dissertation would not exist. I would also like to thank Stanley Osher, Luminita Vese and Jeff Brantingham for being part of my dissertation committee and for their time.

Second, I would like to thank all my collaborators. I would like to thank Alice Koniges for her advice and support. She has hosted me at Lawrence Berkeley National Lab (LBNL) for one year and we have collaborated together and produced two papers. I would also like to express my gratitude to Yun (Helen) He, Samuel Williams, Thorsten Kurth, Brandon Cook and Jack Deslippe from LBNL for their collaboration on one of my papers. I am also grateful for having worked with Ekaterina Merkurjev on one paper published on Image Processing On Line (IPOL). She has provided me with valuable advice. I would also like to thank Da Kuang for his suggestions on optimizing my codes. I am also grateful for having worked with Miguel Colom and Nelson Monzón López to publish the parallelized codes and demo on IPOL. Moreover, I would like to express my gratitude to Jean-Michel Morel for hosting me in CMLA ENS Cachan, during which time we collaborated on one paper. I would also like to thank Javier Sánchez for collaborating with me on the body-worn video paper.

The last five years of my life would not have been the same if I did not have the support of my wonderful family and friends. I would like to thank Michael (Xiyang), Fangbo, Chuyuan, Stephanie and my other friends for making my time in graduate school very memorable. I would like to thank my father Guangke and my mother Fengqin for being the best parents ever and for supporting me all the time. Last, I would like to thank Huajun for his love and support.

This work was supported by NSF grants DMS-1417674, DMS-1045536, DMS-1118971 and DMS-1737770, AFOSR MURI grant FA9550-10-1-0569, UC Lab Fees Research grant 12-LR-236660, ONR grant N00014-16-1-2119, FUI project Plein Phare by BPI-France and NIJ Grant 2014-R2-CX-0101. 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.

VITA

Education

University of California, Los Angeles,

Ph.D., Candidate, Applied Mathematics, **expected Jan 2018**

Shanghai Jiao Tong University, Shanghai,

B.S., Applied Mathematics (minor: Physics) **June 2013**

Expertise

- Background in applied and computational mathematics, optimization, numerical analysis, algorithms, differential equations, scientific computing.
- Programming skills: C++, Python, Matlab, OpenMP, Linux, LATEX, SQL, PHP.

Experience

- Software Engineer Intern at Google, Mountain View, CA, Jun.-Sep. 2017
- Visiting Research Assistant at CMLA Research Center for Applied Math, Paris, Sept.-Dec. 2016
- Ph.D. Software Intern at Facebook, Menlo Park, CA, Jun.-Sept. 2016
- Research Intern at Lawrence Berkeley National Lab, Berkeley, CA, Jun. 2015-May. 2016

Journal/ Conference Publications

- Meng, Z., Sánchez, J., Morel, J. and Bertozzi, A.L. *Ego-motion Classification for Body-worn Videos*, to appear in proceedings of Imaging, vision and learning based on optimization and PDEs, Springer.
- Meng, Z., Merkurjev, E., Koniges, A. and Bertozzi, A.L. *Hyperspectral image classification using graph clustering methods*, Image Processing On Line, 7:218-245, 2017.
- Meng, Z., Koniges, A., He, Y., Williams, S., Kurth, T., Cook, B., Deslippe, J. and Bertozzi, A.L. *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.

Conference Presentations/ Posters

- AWM SIAM 17 workshop, Pittsburg, PA, Jul. 11, 2017
- Joint Statistical Meetings, Chicago, IL, Jul. 30-Aug. 4, 2016
- Southern California Applied Mathematics Symposium, Claremont, CA, Jun. 4, 2016

- SIAM Conference on Imaging Science, Albuquerque, NM, May. 23-26, 2016
- Summer student session at Lawrence Berkeley National Lab, Berkeley, CA, Aug. 6, 2015
- NSF PI meeting, Washington DC, Jul. 13-15, 2015

Teaching/ Mentoring Experience

- Co-mentor for Applied Mathematics REU at UCLA (summer 2014)
- Teaching Assistant for Mathematical Imaging (fall 2015)

CHAPTER 1

Introduction

Multi-class classification is one of the fundamental problems in machine learning and computer vision. The massive amount of data contained in image and video data also requires computational efficiency. This motivates us to develop a parallel implementation of two noval classification algorithms. The original work in this thesis comes from two publications and one manuscript that is accepted for publication. *Chapter 4* presents the work of one publication [107] which contains the full development of the hyperspectral image classification using two graph methods along with the parallelized codes and online demo. *Chapter 5* discusses the publication [106] of detailed development of the parallelized algorithms using different high performance computing techniques. In *Chapter 6*, we discuss a direct application of the parallelized classification algorithm – the ego-motion classification of body-worn videos [108].

The two classification algorithms we focus on use a graphical framework in which each data point is regarded as a node on a graph. *Chapter 2* reviews the definitions and properties of the graph framework. Graphs are often used to exploit underlying similarities in the data [13, 37, 149, 155, 156, 162]. For example, spectral graph theory [43, 118] uses this approach to perform various tasks in imaging and data clustering. Graph-based formulations have been also used extensively for image processing applications [21, 44, 45, 53, 71, 72, 74, 93, 133]. Specifically, algorithms for image denoising in [30], image inpainting and reconstruction in [70, 124, 154], image deblurring in [96] and manifold processing in [53] all utilize such formulations. The graphical approach has appeared in many recent works for different kinds of applications: semi-supervised learning [8, 13, 38, 67, 109, 111–113], image processing [46, 53, 154], machine learning [157], and graph cuts [22, 24–26, 77, 110, 133, 140].

In *Chapters 3*, we outline two very efficient methods to classify data sets, so that the similarity between data in one class is much larger than the similarity between data of different classes. The first algorithm we outline in *Chapter 3* is semi-supervised [112, 113], which requires some known labels as input, and the second one is unsupervised [80]. Both methods make use of the

MBO scheme, which is a well-established PDE method for evolving an interface by mean curvature [114, 117]. It is adapted to the graph setting in [66, 67]. A comprehensive reference for casting continuous PDEs in a graph form is [73]. The semi-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 [36, 146] and 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.

The two algorithms leverage the Nyström extension to calculate eigenvalue/eigenvectors of the graph Laplacian and this is a self-contained module that can be used in conjunction with other graph-Laplacian based methods such as spectral clustering. The calculation is very efficient and the computational complexity is $O(mN)$, where m is a small number. This is achieved by calculating the eigen-decomposition of a smaller system of size $m \ll N$ and then expanding the result back up to N without any significant decrease in the accuracy of the solution. We present the optimized implementation and directive-based OpenMP parallelization of the Nyström eigensolver in *Chapter 5*.

One application of the multi-class classification algorithms is the class detection of pixels in hyperspectral images, where each pixel contains many channels. We present the application to hyperspectral image classification in *Chapter 4*. 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 principle component analysis (PCA) [60], independent component analysis [150], signal subspace identification [17], discrete wavelet transform [85], band reduction based on rough sets [132], projection pursuit algorithm [84], and clonal selection feature-selection algorithm [153]. The Fuzzy C-Means (FCM) algorithm is a well-known tool to find proper clusters, which can be used for hyperspectral image classification [81], and can be further enhanced by the Support Vector Domain Description [121]. Support vector machines (SVM) [19,145] are another popular approach to the supervised classification problem, including one involving hyperspectral data [34, 75, 105]. Similar approaches include transductive SVM [29] and SVM with composite kernels [35]. Some of these methods use kernels, which have been shown to improve performance [34, 92], especially when the data is not linearly separable. Examples of kernel methods can be found in [65,147,152]. Other successful techniques include multinomial logistic regression [18, 90], which was applied to hyperspectral image segmentation in [94, 95], and sparse representation, also applied to such images in [41, 42]. The reader is referred to [100, 141] for more approaches.

For hyperspectral images, we consider each pixel as a node on a graph, and take the values in the channels to form the feature vectors. 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 [67], 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 [31, 69, 70, 112] using the nonlocal means feature vector.

For hyperspectral images, the pixel can have very high dimensions of channels, and the hyperspectral videos contain massive amount of data. The need to completely and accurately capture critical information from the exponential growth of scientific data is prominent and is a big challenge as well. Many successful machine learning methods have not been accelerated by high performance computing. This is a big opportunity and motivates us to develop parallel implementations and optimizations of the two classification algorithms [106] described in *Chapter 3*. We describe parallel implementations and optimizations of the new algorithms in *Chapter 5*. We focus on shared memory many-core parallelization schemes that will be applicable to next generation of exascale architectures such as the upcoming Intel Knights Landing processor. We use performance tools to collect the hotspots and memory access of the serial codes. After analyzing the computational hotspots, we find that an optimized implementation of the Nyström eigensolver is the computational challenge for scalability of graph MBO methods.

In [40] the authors discussed the parallelization of spectral clustering, which makes use of the spectrum (eigenvalues/vectors) of the similarity matrix of the data to perform dimensionality reduction before clustering in fewer dimensions. They developed a parallelized algorithm for an eigensolver for a sparse matrix. However, there is another class of non-sparse graph Laplacians that are prohibitive to compute, and the Nyström method gives a low-rank approximation for the non-sparse Laplacians. This class of algorithms have not been parallelized. We demonstrate a parallelization of the Nyström eigensolver on the exascale-ready machine Cori at NERSC to efficiently compute low rank representations of fully connected similarity graph, the backbone calculation of spectral clustering or semi-supervised/unsupervised machine learning methods. Where possible, we also use library routines. We then optimize the OpenMP implementations and detail the performance on traditional supercomputer nodes (in our case a Cray XC30), and test the optimization steps on emerging testbed systems based on Intel's Knights Corner and Landing processors. We show OpenMP parallelization and SIMD regions in combination with optimized library routines achieve almost ideal scaling and manyfold speedup over serial implementations. A large number of optimization techniques and analyses are necessary before the algorithm reaches almost ideal scaling.

The parallelization of the two classification algorithms make the algorithms efficient enough for video analysis. In *Chapter 6*, we present an approach for segmenting or catagorizing frames of body-worn videos (BWV) by different ego-motion categories. Affordable high-quality cameras for recording the first-person point-of-view experience, such as GoPro, are an increasingly common item in many aspects of people's lives. Prior work on vision-based first-person human action analysis has focused a lot on indoor activities, such as object recognition [126], hand gesture recognition [104] [137], sign language recognition [135], context aware gesture recognition [98], hand tracking [136] and detecting daily life activities [125]. Work with body-worn sensors has also been shown to be effective for categorizing human actions and activities [82] [134]. An unsupervised ego-action learning method was proposed in [88] for sports videos. The basis of video indexing is to model the transformation between successive frames in the video. For the purpose of video indexing, several studies have examined parametric models of frame transformation such as [20], [87]. Parametric models can be also used for video stabilization [130], and panorama construction [86].

We know that human motion observed from a first-person point-of-view can be captured by the global displacement between successive frames. This means that we should be able to aggregate global motion and marginalize out local outlier motion. We also know that motion involving the human gait has an inherent frequency component. Therefore we expect that frequency analysis can be used as an important feature for ego-motion categorization. We develop a parametric model for calculating the simple global representation of motion. This approach produces a low dimensional representation of the motion of the ego. We then classify the ego-motion using the graph-based semi-supervised and unsupervised learning algorithms described in *Chapter 3*.

We consider the ego-motion classification problem with both benchmark and real-world data. Working with both types of data is critical because of the stark differences in the degree of difficulty in the analysis of video data collected under controlled and uncontrolled or “wild“ conditions.

Benchmark datasets with known ground truth are developed under experimental conditions controlled by the researcher. Such datasets attempt to simulate the types of behaviors that are of most interest to the researcher. Simulations may favor positive outcomes because they seek not only to limit sources of error linked to video image quality, but also enhance target behaviors of interest. For example, experimental protocols that seek to enhance camera stability, ensure adequate lighting conditions, avoid obstructions may all assist in the algorithmic task. Ensuring that experimental participants enact well-defined or discrete transitions between different types of behavior, or exaggerate the differences between behavioral modes may favor accurate segmentation. We draw on choreographed video collected under controlled circumstances to develop our approach.

Videos not collected under controlled conditions may nevertheless be hand-labeled by the researcher to produce a ground truth. Such videos may be subject to many more quality challenges than simulated scenes. Actual behavior and conditions as they exist on the ground are unforgiving. People in real-world settings may not act in discrete, linear sequences, nor are they necessarily inclined to exaggerate their different actions for easy detection. Ego-motions may also proceed so quickly that they defy discrete recognition. We may also lack sufficient semantic categories to capture the diversity of real-world behavior. Real-world video systems suffer from poor camera stability, low frame rate, low resolution, poor color saturation and data collection errors (both human and mechanical). All of these effects can drastically impact the ability of the researcher to label video for ground truth, which introduces errors into algorithmic methods. We draw on police body-worn video (BWV) to evaluate how our methods perform under challenging real-world conditions. Police BWV is typically shaky, contains noise from low light conditions, poor color saturation and occlusions, and represents diverse and often mixed motion routines.

The rest of this thesis is organized as follows. *Chapter 2* introduces the background and preliminaries of the graph setting, the definition and properties of the graph Laplacians. In *Chapter 3*, we present the review of the semi-supervised and unsupervised graph MBO methods. We also present the theoretical background of the Nyström extension methods. In *Chapter 4*, we present the full development of both the graph MBO algorithms on a broad range of hyperspectral imagery including the online parallelized codes and demo. We apply the algorithms on much larger data sets using much less training data than previous work. We further discuss the procedure and details of the parallelization of the two graph algorithms in *Chapter 5*. We use OpenMP as the parallelization language and parallelize the most time-consuming part of the codes. Math library use, vectorization, chunk methods and the roofline model are discussed. In *Chapter 6*, we deal with a real world problem which is a direct application of the graph classification methods. We address the task of discovering ego-motion from the body-worn videos. The feature vectors are built using the similarity transformations between successive video frames and the graph MBO methods are used to segment the video frames into different ego-motion categories.

CHAPTER 2

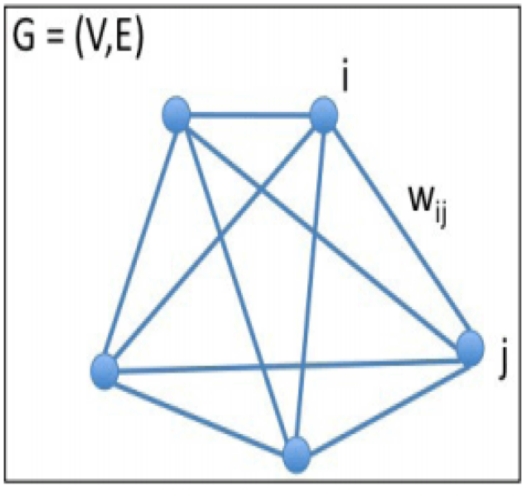
Background

2.1 Graphical Framework

For all methods, we consider a graphical framework with an undirected graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$, where $\mathbf{V}$ is the set of vertices and $\mathbf{E}$ is the set of edges. Let $\mathbf{w}$ be the weight function (defined on the set of edges) which measures the similarity between each two vertices. Also, let

$$d(x) = \sum_{y \in V} w(x, y)$$

be the degree of vertex x. The diagonal matrix $\mathbf{D}$ contains the degree along its diagonal entries and the matrix $\mathbf{W}$ contains values of the weight function. Both matrices are of dimension N by N, where N is the number of vertices. A representation of the graphical framework is shown below.



One advantage of using a graphical framework is that it allows one to be non-local and take into account the relationship between any two nodes in the data set. Therefore, repetitive structure and texture can be captured. The graphical framework is also more general, and can be easily constructed for any data set.

Weight Function

When choosing a weight function, the goal is to give a large weight to an edge if the two vertices it is connecting are similar and a small weight if they are dissimilar. One popular choice for the weight function is the Gaussian

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

where $d(x,y)$ is some distance measure between the two vertices x and y, and σ is a parameter to be chosen. For example, if the data set consists of points in $\mathbb{R}^2$, $d(x,y)$ can be the Euclidean distance between point x and point y, since points farther away are less likely to belong to the same cluster than points closer together. For images, $d(x,y)$ can be defined as the weighted 2-norm of the difference of the feature vectors of pixels x and y, where the feature vector of a node consists of intensity values of pixels in its neighborhood, as described in [69].

Another choice for the similarity function used in this work is the Zelnik-Manor and Perona weight function [123] for sparse matrices:

$$w(x,y) = e^{-\frac{d(x,y)^2}{\sqrt{\tau(x)\tau(y)}}}, \tag{2}$$

where the local parameter $\tau(x) = d(x,z)^2$, and z is the M^{th} closest vertex to vertex x.

Note that it is not necessary to use a *fully connected graph setting*, which might be a computational burden. Specifically, the fully connected graph can be approximated by a much smaller graph by only including an edge between vertex x and y if x is a k-nearest neighbor of y or vice versa. This is called a *k-nearest neighbor graph.* One can also create a *mutual k-nearest neighbor graph* by only including an edge between x and y if both of them are k-nearest neighbors of each other.

Graph Laplacian

In the graphical framework, it is possible to introduce some common mathematical operators in a graphical setting. For this section, we will only be concerned with the graph version of the differential Laplace operator. Although many versions exist, we mention the following three matrices that are related to the differential Δ operator:

* $\mathbf{L} = \mathbf{D} - \mathbf{W}$, unnormalized Laplacian.

* $\mathbf{L}_s = \mathbf{D}^{-\frac{1}{2}}\mathbf{L}\mathbf{D}^{-\frac{1}{2}}$, symmetric Laplacian.

* $\mathbf{L_{rw}} = \mathbf{D^{-1}L}$, random walk Laplacian.

The last two matrices represent normalized versions of the original Laplacian, as it is sometimes desirable to scale, especially in high dimensions. Note that we have the following equations:

$$\mathbf{L}u(x) = \sum_y w(x,y)(u(x) - u(y)),$$

$$\mathbf{L}_s u(x) = \frac{1}{d(x)} \sum_y w(x,y)(u(x) - u(y)),$$

$$\mathbf{L_{rw}} u(x) = \frac{1}{\sqrt{d(x)}} \sum_y w(x,y)\big(\frac{u(x)}{\sqrt{d(x)}} - \frac{u(y)}{\sqrt{d(y)}}\big).$$

The graph Laplacian $\mathbf{L}$ has the following easily shown properties [43, 148]:
1) $\mathbf{L}$ is symmetric and positive semi-definite.
2) $\mathbf{L}$ has N non-negative, real-valued eigenvalues $0 = \lambda_1 \leq \lambda_2 \leq \lambda_N$.
3) The smallest eigenvalue of $\mathbf{L}$ is 0; eigenvector is just a constant vector.

The graph Laplacians $\mathbf{L}_s$ and $\mathbf{L_{rw}}$ have the following easily shown properties:
1) $\mathbf{L}_s$ and $\mathbf{L_{rw}}$ are positive semi-definite.
2) $\mathbf{L}_s$ and $\mathbf{L_{rw}}$ have N non-negative, real-valued eigenvalues $0 = \lambda_1 \leq \lambda_2 \leq \lambda_N$.
3) λ is an eigenvalue of $\mathbf{L_{rw}}$ with eigenvector u if and only if λ is an eigenvalue of $\mathbf{L}_s$ with eigenvector $w = D^{\frac{1}{2}}u$.
4) The smallest eigenvalue of $\mathbf{L}_s$ and $\mathbf{L_{rw}}$ is 0.

It is also worthwhile to mention that the multiplicity of eigenvalue 0 equals the number of connected components in the graph.

Other Graph Operators

Another important operator that arises from the need to define variational methods on graphs is the mean curvature on graphs. This non-local operator was introduced by Osher and Shen in [122], who defined it via graph based p-Laplacian operators. p-Laplace operators are a family of quasilinear elliptic partial differential operators defined for $1 \leq p < \infty$:

$$\mathbf{L}^p(u) = \nabla \cdot (\mid \nabla u \mid^{p-2} \nabla u).$$

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

The discrete graph version of p-Laplace operators is defined in [53] as:

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

Note that the graph 2-Laplacian is just the graph Laplacian, which is consistent with the continuous case.

Let us now define the mean curvature on graphs- the discrete analog of the mean curvature of the level curve of a function defined on a continuous domain of $\mathbb{R}^N$:

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

Note that in the case of unweighted mesh graph, κ_w becomes a numerical discretization of mean curvature. This curvature, κ_w, is also used in [48] as a regularizer in a graph adaptation of the Chan-Vese method. In their work [144], Van Gennip et al. propose a different definition of mean curvature on graphs and prove convergence of the MBO scheme on graphs.

Previous Work Using Graphs

Graph-based formulations have been used extensively for image processing applications [21, 44, 45, 53, 71, 72, 74, 93]. Interesting connections between these different algorithms, as well as between continuous and discrete optimizations, have been established in the literature. Grady has proposed a random walk algorithm [72] that performs interactive image segmentation using the solution to a combinatorial Dirichlet problem. Elmoataz et al. have developed generalizations of the graph Laplacian [53] for image denoising and manifold smoothing.

Couprie et al. in [44] define a conveniently parameterized graph-based energy function that is able to unify graph cuts, random walker, shortest paths and watershed optimizations. There, the authors test different seeded image segmentation algorithms, and discuss possibilities to optimize more general models with applications beyond image segmentation. In [45], alternative graph-based formulations of the continuous max-flow problem are compared, and it is shown that not all the properties satisfied in the continuous setting carry over to the discrete graph representation. For general data segmentation, Bresson et al. in [23], present rigorous convergence results for two algorithms that solve the relaxed Cheeger cut minimization, and show a formula that gives the correspondence between the global minimizers of the relaxed problem and the global minimizers of the combinatorial problem.

2.2 Graphical Framework, Extended

For the MBO method, we only need to define the Laplace operator in a more general graphical framework, since this is the only operator encountered in the procedure. We also outline the graphical framework in more detail here, giving more general definitions for other operators. We define operators on graphs in a similar fashion as done in [76,143], where the justification for these choices is shown. We note that some of the formulism is not used elsewhere in this thesis, however, we include it here for completeness.

Assume m is the number of vertices in the graph and let $\mathcal{V} \cong \mathbb{R}^m$ and $\mathcal{E} \cong \mathbb{R}^{\frac{m(m-1)}{2}}$ be Hilbert spaces (associated with the set of vertices and edges, respectively) defined via the following inner products:

$$\langle u, \gamma \rangle_V = \sum_x u(x)\gamma(x)d(x)^r,$$

$$\langle \psi, \phi \rangle_{\mathcal{E}} = \frac{1}{2}\sum_{x,y} \psi(x,y)\phi(x,y)w(x,y)^{2q-1}$$

for some $r \in [0,1]$ and $q \in [\frac{1}{2}, 1]$. Let us also define the following norms:

$$\|u\|_{\mathcal{V}} = \sqrt{\langle u, u \rangle_{\mathcal{V}}} = \sqrt{\sum_x u(x)^2 d(x)^r},$$

$$\|\phi\|_{\mathcal{E}} = \sqrt{\langle \phi, \phi \rangle_{\mathcal{E}}} = \sqrt{\frac{1}{2}\sum_{x,y} \phi(x,y)^2 w(x,y)^{2q-1}},$$

$$\|\phi\|_{\mathcal{E},\infty} = \max_{x,y} |\phi(x,y)|.$$

The gradient operator $\nabla : \mathcal{V} \to \mathcal{E}$ is then defined as:

$$(\nabla u)_w(x,y) = w(x,y)^{1-q}(u(y) - u(x)). \tag{3}$$

The Dirichlet energy does not depend on r or q:

$$\frac{1}{2}\|\nabla u\|_{\mathcal{E}}^2 = \frac{1}{4}\sum_{x,y} w(x,y)(u(x) - u(y))^2.$$

The divergence $div : \mathcal{E} \to \mathcal{V}$ is defined as the adjoint of the gradient:

$$(\mathrm{div}_w\, \phi)(x) = \frac{1}{2d(x)^r}\sum_y w(x,y)^q(\phi(x,y) - \phi(y,x)), \tag{4}$$

where we define the adjoint using the following definition: $\langle \nabla u, \phi \rangle_{\mathcal{E}} = -\langle u, \operatorname{div}_w \phi \rangle_{\mathcal{V}}$.

We now have a family of graph Laplacians $\triangle_r = \operatorname{div}_w \dot{\nabla} : \mathcal{V} \to \mathcal{V}$:

$$(\triangle_w u)(x) = \sum_y \frac{w(x,y)}{d(x)^r}(u(y) - u(x)). \tag{5}$$

Viewing u as a vector in $\mathbb{R}^m$, we can write

$$-\triangle_w u = (\mathbf{D}^{1-r} - \mathbf{D}^{-r}\mathbf{W})u.$$

The case with $r = 0$ is the unnormalized Laplacian

$$\mathbf{L} = \mathbf{D} - \mathbf{W}.$$

However, the matrix $\mathbf{L}$ is usually scaled to guarantee convergence to the continuum differential operator in the limit of large sample size [15]. Although several versions exist, we consider two popular versions of the symmetric Laplacian

$$\mathbf{L_s} = \mathbf{D}^{-\frac{1}{2}}\mathbf{L}\mathbf{D}^{-\frac{1}{2}} = \mathbf{I} - \mathbf{D}^{-\frac{1}{2}}\mathbf{W}\mathbf{D}^{-\frac{1}{2}} \tag{6}$$

and the random walk Laplacian ($r = 1$)

$$\mathbf{L_{rw}} = \mathbf{D}^{-1}\mathbf{L} = \mathbf{I} - \mathbf{D^{-1}W}.$$

The advantage of the former formulation is its symmetric property which allows for more efficient implementations.

A family of anisotropic total variations $TV_w : \mathcal{V} \to \mathbb{R}$ can now be defined:

$$TV_w(u) = \max\left\{\langle \operatorname{div}_w \phi, u\rangle_{\mathcal{V}} : \phi \in \mathcal{E}, \|\phi\|_{\mathcal{E},\infty} \leq 1\right\} = \frac{1}{2}\sum_{x.y} w(x,y)^q |u(x) - u(y)|. \tag{7}$$

Lastly, in this section, we consider the following graph-based Ginzburg Landau functional:

$$GL_\epsilon(u) = \|\nabla u\|^2_{\mathcal{E}} + \frac{1}{\epsilon}\sum_x W(u(x)).$$

Remark. *It is noted in [143] that although the first term in the continuous Ginzburg-Landau functional*

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

is scaled by ϵ, the first term of GL_ϵ contains no ϵ. This occurs because the Dirichlet energy in the continuous Ginzburg-Landau functional is unbounded for functions of bounded variation and *taking on two values of the minima of the double-well potential (almost everywhere). However, the difference terms of GL_ϵ are finite even in the case of binary functions, and no rescaling of the first term is necessary.*

It remains to choose the parameters q and r. In [143], the authors choose $q = 1$, where it is shown that for any r, TV_w is the Γ-limit (Gamma convergence) of a sequence of graph-based Ginzburg-Landau (GL)-type functionals:

Theorem 1. *$GL_\epsilon \overset{\Gamma}{\to} GL_0$ as $\epsilon \to 0$, where*

$$GL_\epsilon(u) = \|\nabla u\|^2_{\mathcal{E}} + \frac{1}{\epsilon}\sum_x W(u(x)) = \frac{1}{2}\sum_{x,y} w(x,y)(u(x)-u(y))^2 + \frac{1}{\epsilon}\sum_x W(u(x)),$$

$$GL_0(u) = \begin{cases} TV_w(u)\textit{with q=1} & \textit{for } u \textit{ s.t. } u(x) \in \{0,1\} \\ \infty & \textit{otherwise} \end{cases}$$

Proof See Theorem 3.1 of [143]. ■

It is also shown that *the addition of a fidelity term is compatible with Γ-convergence*.

We choose $r = 1$ because it results in a normalized random walk Laplacian and the eigenvectors as well as the corresponding eigenvalues of the matrix can be efficiently calculated. Although the random walk Laplacian matrix itself is not symmetric, spectral graph theory described in [43] shows that the eigenvectors of the random walk Laplacian can be directly computed from knowing the diagonal matrix $\mathbf{D}$ and the eigenvectors of the symmetric graph Laplacian (which is a symmetric matrix) $\mathbf{L_s}$. In particular, λ is an eigenvalue of $\mathbf{L_{rw}}$ with eigenvector u if and only if λ is an eigenvalue of $\mathbf{L_s}$ with eigenvector $w = \mathbf{D}^{\frac{1}{2}}u$. This is proved by multiplying the eigenvalue equation $\mathbf{L_{rw}}u = \lambda u$ by $\mathbf{D}^{\frac{1}{2}}$ from the left and then substituting $w = \mathbf{D}^{\frac{1}{2}}u$, obtaining $\mathbf{L_s}w = \lambda w$.

We take advantage of this property by calculating the eigenvalues and eigenvectors of the symmetric graph Laplacian (since symmetric matrices allow for more efficient implementations) and then using this information to calculate the same for the random walk Laplacian.

To summarize, we use the above operator definitions with $q = 1$ and $r = 1$.

Therefore, we use

$$TV_w(u) = \max\left\{\langle \operatorname{div}_w \phi, u\rangle_{\mathcal{V}} : \phi \in \mathcal{E}, \|\phi\|_{\mathcal{E},\infty} \leq 1\right\} = \frac{1}{2}\sum_{x.y} w(x,y)|u(x)-u(y)|. \tag{8}$$

2.3 Clustering

The increase in both volume and the variety of data requires advances in methodology to automatically understand, process, and summarize the data [83]. Often, a clear distinction is made between learning problems that are (i) supervised (classification) or (ii) unsupervised (clustering), the first involving only labeled data (training data with known class labels) while the latter involving only unlabeled data [52]. There is a growing interest in a hybrid setting, called semi-supervised learning [39]; in semi-supervised classification, the labels of only a small portion of the training data set are available. The unlabeled data, instead of being discarded, are also used in the learning process.

The goal of data clustering, also known as cluster analysis, is to discover the natural groupings of a set of patterns, points and objects. An operational definition of clustering can be stated as follows: Given a representation of n objects, find K groups based on a measure of similarity such that the similarity between objects in the same group are high while the similarities between objects in different groups are low. We provide a brief overview of two clustering methods: K-means and spectral clustering here.

K-Means

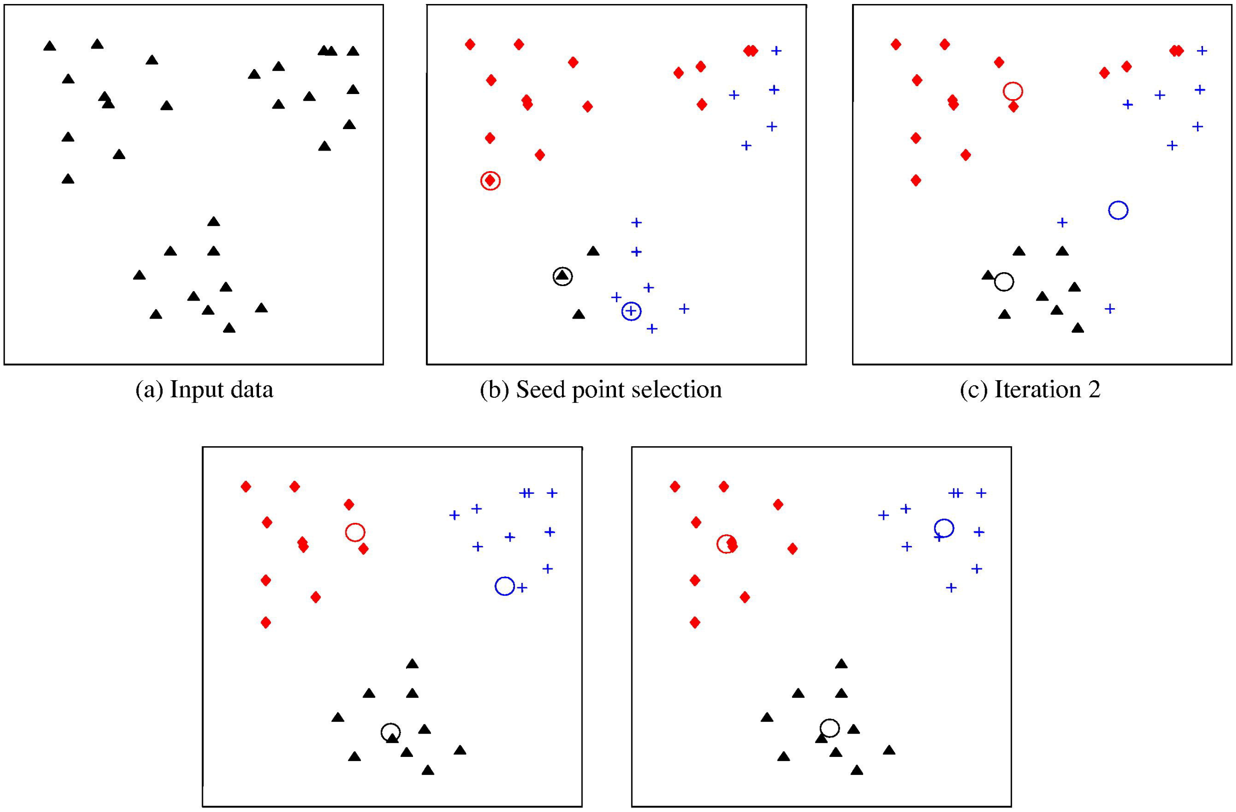
One of the most popular and simple clustering algorithms is K-means [101]. In spite of the fact that K-means was proposed over 50 years ago and thousands of clustering algorithms have been published since then, K-means is still widely used. We provide a brief overview of K-means here.

Let $X = x_i,\ i = 1, ..., n$ be the set of n-dimensional points to be clustered into a set of K clusters, $C = c_k, k = 1, ..., K$. K-means algorithm finds a partition such that the squared error between the empirical mean of a cluster and the points in the cluster is minimized. Let μ_k be the mean of cluster c_k. The squared error between μ_k and the points in cluster c_k is defined as:

$$J(c_k) = \sum_{x_i \in c_k} ||x_i - \mu_k||^2. \tag{9}$$

The goal of K-means is to minimize the sum of squared error over all K clusters,

$$J(C) = \sum_{k=1}^{K} \sum_{x_i \in c_k} ||x_i - \mu_k||^2.$$



(a) Input data

(b) Seed point selection

(c) Iteration 2

(d) Iteration 3

(e) Final clustering

Figure 1: Illustration of K-means algorithm. [83] (a) Two-dimensional input data with three clusters. (b) Three seed points selected as cluster centers and initial assignment of the data points to clusters. Panels (c) and (d) show intermediate iterations updating cluster labels and their centers. (e) Final clustering obtained by K-means algorithm at convergence.

Minimizing this objective is known to be an NP-hard problem. Thus K-means, which is a greedy algorithm, can only converge to a local minimum. K-means starts with an initial partition with K clusters and assign data points to clusters so as to reduce the squared error. The main steps of K-means are as follow:

* 1. Select an initial partition with K clusters; repeat steps 2 and 3 until cluster membership stablizes.

* 2. Generate a new partition by assigning each data point to its closes cluster center.

* 3. Compute new cluster centers.

Spectral Clustering

Spectral clustering [148] 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 as follows:

Spectral Clustering

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

A generalized version of spectral clustering using the p-Laplacian is proposed in [32].

CHAPTER 3

Graph MBO Method

Chapter 3 is a review of a fast algorithm (MBO method) for classification and image processing [57, 80, 112]. The graph MBO method is inspired by diffuse interface models that have been used in a variety of problems, such as those in fluid dynamics and materials science. As an alternative to L^1 compressed sensing methods, Bertozzi and Flenner introduce a graph-based model based on the Ginzburg-Landau functional in their work [15] to solve the semi-supervised classification problem. To define the functional on a graph, the spatial gradient is replaced by a more general graph gradient operator. Analogous to the continuous case, the first variation of the model yields a gradient descent equation with the graph Laplacian, which is then solved by a numerical scheme with convex splitting. To reduce the dimension of the graph Laplacian and make the computation more efficient, the authors propose the Nyström extension method [62] to approximate eigenvalues and the corresponding eigenvectors of the graph Laplacian. Moreover, many applications suggest that the MBO scheme of Merriman, Bence and Osher [115] for approximating the motion by mean curvature performs very well in minimizing functionals built around the Ginzburg-Landau functional. For example, the authors of [57] propose an adaptation of the scheme to solve the piecewise constant Mumford-Shah functional. The author of [112] adapted the MBO scheme for solving semi-supervised graph based equations to create a simple algorithm that achieves faster convergence through a small number of computationally inexpensive iterations. The unsupervised graph MBO method is developed based on the Mumford-Shah model and generalized to the graph setting [80]. The graph MBO methods can be applied to various problems, such as image processing, classification and object detection in, for example, hyperspectral data.

3.1 Semi-supervised Graph MBO Algorithm

In this chapter, 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 [113].

We approach the semi-supervised classification problem using energy minimization techniques. Similar approaches have been used in [15, 16], where the problem is formulated as a minimization of the Ginzburg-Laudau (GL) functional (in graph form) with a fidelity term. In [112], the authors propose an MBO scheme to solve the binary classification problem. The algorithm was use successfully in classification and image inpainting.

Derivation

By using the Ginzburg-Landau functional for the regularization term and the L^2 fidelity term, we obtain the following minimization problem:

$$\min_u \big\{E(u) = \frac{\epsilon}{2}\int |\nabla u|^2 dx + \frac{1}{\epsilon}\int W(u)dx + \int \lambda(x)|u-u_0|^2 dx\big\}. \tag{11}$$

The energy can be minimized in the L_2 sense using gradient descent. This leads to the following dynamic equation (*modified Allen-Cahn equation*):

$$\frac{\partial u}{\partial t} = -\frac{\delta GL_\epsilon}{\delta u} - \mu\frac{\delta F}{\delta u} = \epsilon\Delta u - \frac{1}{\epsilon}W'(u) - \mu\frac{\delta F}{\delta u}, \tag{12}$$

where Δ represents the Laplacian operator. A local minimizer of the energy is obtained by evolving this expression to steady state. Note that E is not convex, and may have multiple local minima.

In their work [15], Bertozzi and Flenner propose a segmentation algorithm for solving (11) in a graph setting. The functional is minimized using the method of gradient descent and convex splitting. The main purpose of the MBO method is to develop a more efficient and simple method for minimizing (11) in the small ϵ limit. An answer comes from the relation between the Allen-Cahn equation and the motion by mean curvature.

Let us start by reviewing this connection in the continuous setting. In [114], Merriman, Bence and Osher propose an algorithm to approximate motion by mean curvature, or motion in which normal velocity equals mean curvature, using threshold dynamics. The authors note that if one applies the heat equation to an interface, then the diffusion blunts the sharp points of the boundary, but has very little effect on the flatter regions. Therefore, one can imagine that diffusion creates some sort of motion by mean curvature, providing that we specify the boundaries of the moving set.

Given a phase field $u(x,t)$, consider the basic (unmodified) Allen-Cahn equation, namely equation (12) without the fidelity term:

$$\frac{\partial u}{\partial t} = \epsilon \Delta u - \frac{1}{\epsilon} W'(u). \tag{13}$$

For small values of ϵ, the following time-splitting scheme can be used numerically to evolve the Allen-Cahn equation:

1. The first step is propagation using:
$$\frac{\partial u}{\partial t} = \epsilon \Delta u.$$
2. The second step is propagation using:
$$\frac{\partial u}{\partial t} = -\frac{1}{\epsilon} W'(u).$$

Note, however, that in the $\epsilon \to 0$ limit, the second step is simply thresholding [114]. Thus, as $\epsilon \to 0$, the time splitting scheme above consists of alternating between diffusion and thresholding steps.

It has been shown [127] that in the limit $\epsilon \to 0$, the rescaled solutions $u_\epsilon(z, t/\epsilon)$ of (13) yield motion by mean curvature of the interface between the two phases of the solutions. This motivates the two sequential steps of the MBO scheme [112]:

1. *Diffusion.* Let $u^{n+\frac{1}{2}} = S(\delta t) u^n$ where $S(\delta t)$ is the propagator (by time δt) of the standard heat equation:
$$\frac{\partial u}{\partial t} = \Delta u.$$
2. *Thresholding.* Let
$$u^{n+1} = \begin{cases} 1 & \text{if } u^{n+\frac{1}{2}} \geq 0, \\ -1 & \text{if } u^{n+\frac{1}{2}} < 0. \end{cases}$$

Barles [11] and Evans [58] have proven rigorously that this scheme approximates motion by mean curvature.

Multiple extensions, adaptations and applications of the MBO scheme are present in literature. We find the modification of the MBO scheme for solving the inhomogeneous Allen-Cahn equation proposed in [57] particularly interesting. To create a fast image segmentation algorithm, Esedoḡlu and Tsai propose a thresholding scheme for minimizing a diffuse interface version of the piecewise constant Mumford-Shah functional

$$MS_\epsilon(u, c_1, c_2) = \int_D \epsilon|\nabla u|^2 + \frac{1}{\epsilon}W(u) + \lambda\{u^2(c_1 - f)^2 + (1-u)^2(c_2 - f)^2\}dx, \tag{14}$$

where f is the image. The first variation of the model (14) yields the following gradient descent equation:

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

and the adaptation of the MBO scheme is used to solve it. Esedoḡlu and Tsai propose the following scheme (similar to the MBO scheme where the propagation step based on the heat equation is combined with thresholding) [57]:

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

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

with appropriate boundary conditions.

* **Step 2** Set

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

Some other extensions of the MBO scheme appeared in [54, 55, 116]. An efficient algorithm for motion by mean curvature using adaptive grids was proposed in [128].

The motion by mean curvature of the MBO scheme can be generalized to the case of functions on a graph in much the same way as the procedure followed for the modified Allen-Cahn equation (12). We now use the same ideas and apply a two-step time splitting scheme to (12) so that the second step is the same as the one in the original MBO scheme. The idea is then to replace all the operators with a more general graph term, since we are considering the graphical framework. The only operator to deal with here, is the Δ operator, and we can replace it by several different versions of the graph Laplacian. In the graphical framework, we have the following three versions that are related to the differential Δ operator:

* $\mathbf{L} = \mathbf{D} - \mathbf{W}$, unnormalized Laplacian.

* $\mathbf{L}_s = \mathbf{D}^{-\frac{1}{2}}\mathbf{L}\mathbf{D}^{-\frac{1}{2}}$, symmetric Laplacian.

* $\mathbf{L_{rw}} = \mathbf{D^{-1}L}$, random walk Laplacian.

Since $\mathbf{L}_s$ is a symmetric matrix, we use the symmetric Laplacian, and thus replace Δu by $-\mathbf{L}_s \boldsymbol{u}$.

The discretized version of the algorithm is:

Binary MBO Algorithm:

Initialize u. Until convergence, alternate between the following two steps:

1. Heat equation with forcing term:

$$\frac{\boldsymbol{u}^{n+\frac{1}{2}} - \boldsymbol{u}^n}{dt} = -\mathbf{L}_s \boldsymbol{u}^n - \boldsymbol{\mu}(\boldsymbol{u}^n - \hat{\boldsymbol{u}}). \tag{15}$$

2. Thresholding:

$$u_i^{n+1} = \begin{cases} 1, & \text{if } u_i^{n+\frac{1}{2}} > 0, \\ -1, & \text{if } u_i^{n+\frac{1}{2}} < 0. \end{cases}$$

Here, after the second step, u_i^n can take only two values of 1 or -1; thus, this method is appropriate for binary segmentation. Note that the fidelity term scaling can be different from the one in (12).

The first part of the two step scheme is solved using the spectral decomposition of the symmetric graph Laplacian. Let $u^n = \sum_k a_k^n \phi_k(x)$ and $C_1 \lambda (u^n - u_0) = \sum_k d_k^n \phi_k(x)$, where $\phi(x)$ are the eigenfunctions of the symmetric Laplacian. Using the obtained representations and equation (15), we obtain

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

where λ_k are the eigenvalues of the symmetric graph Laplacian.

In practice, it can be productive to repeat the diffusion step a number of times before thresholding. In order to keep the convention that one iteration of the diffusion-thresholding procedure corresponds to one time step, we divide dt by the number of diffusion steps per iteration, which we denote N_S.

To compute the eigenvalues and eigenvectors of the graph Laplacian, we use two methods. One of them is the Raleigh-Chebyshev procedure of [6] and the second one is the Nyström extension [15, 62, 63].

Application to Image Inpainting

The problem of fitting information in the missing pixels of an image is an important inverse problem in image processing with various applications. Obviously, the goal is to produce a modified image that will look natural to an observer. The problem of inpainting may also be seen as the problem of removing occlusive objects from an image. Sparse reconstruction refers to the problem of recovering randomly distributed missing pixels.

There are numerous approaches to solve these problems in the current literature. Local TV methods became state-of-the-art techniques for image impainting. However, since they do not perform well on images with high texture, methods that decompose images into cartoon and texture and simultaneously inpaint both are developed [14, 131]. The problem is also solved with nonlocal inpainting methods. We are particularly interested in the nonlocal inpainting algorithm from [70] as we develop a computationally efficient nonlocal method. Some very successful nonlocal methods for inpainting and sparse reconstruction are given in [7] and [59]. Recently, the class of methods that use dictionaries of small patches that commonly appear in natural images became increasingly popular. Those methods, besides inpainting, are also successful in denoising as shown in [102]. In addition, a method for image inpainting using Navier-Stokes fluid dynamics is proposed in [15]. The authors use Navier-Stokes dynamics to propagate isophotes into the inpainting region, thus simulating the way painting restoration is done. Wavelets and framelets are also successfully applied to solve inpainting problems [33, 49].

The segmentation algorithm can be modified slightly for the purpose of image inpainting.

Binary Image Inpainting Although the key steps of the segmentation algorithm remain the same when it is modified for image inpainting, there are differences to be noted. For example, if a damaged image is used to construct the adjacency matrix W, the results might not be accurate, so we first apply a fast and simple H^1 inpainting algorithm on the image and then use the result to create W. The H^1 inpainting algorithm we apply is just the local minimization problem:

$$\min_u \int |\nabla u|^2 dx + \int \lambda(x)(u - u_0)^2 dx.$$

The latter term in the sum is the fidelity term, and, of course, we are not limited to this formulation of it. Although the latter algorithm is fast, it does not perform well on images with high textures and repetitive structures nor does it preserve edges [64], something that we achieve.

The matrix W is built by using a window of a certain size around each pixel. We set $W(x, y) =$

0 for all pixels j that are not in the window of pixel i. Inside the window, $W(x,y) = w(x,y)$, where the weight function is calculated in the same way as for binary image labeling. No updating of the matrix W is necessary. The Rayleigh-Chebyshev procedure is used to calculate the eigenvectors and eigenvalues of the graph Laplacian for binary inpainting. The fidelity region is the non-damaged region. We initialize u to be the middle value for the non-fidelity points.

Grayscale Image Inpainting To generalize to graycale inpaining, we split the signal bit-wise into channels, as in [49]:

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

where u_m denotes the m^{th} digit in the binary representation of the signal, and $u_m \in \{0,1\}$ for $\forall x$.

A fully connected graph is created in the same way as in the binary inpainting case. The Nyström extension method is used to calculate the eigenvalues and corresponding eigenvectors since the size of the graph is very large. The fidelity region and initialization is the same as in the binary inpainting case.

Updating the weight matrix is often necessary for grayscale inpainting, since the adjacency matrix formed from the damaged image is usually not good enough to restore texture and complex patterns, as it contains "bad" regions whose values lie far from the true value. In our tests, every few iterations, the matrix is updated using the result from the last iteration as the "new image".

The grayscale inpainting results along with their PSNR are displayed in Figures 2. The graph MBO method achieves better result and faster calculation than the other inpaining methods.

Extension to Multiclass Case

A multi-class extension of that algorithm is described in [67, 111]. 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

$$\Sigma^{\hat{n}} := \left\{ (x_1, \ldots, x_{\hat{n}}) \in [0,1]^{\hat{n}} \,\middle|\, \sum_{k=1}^{\hat{n}} x_k = 1 \right\}. \tag{16}$$

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.

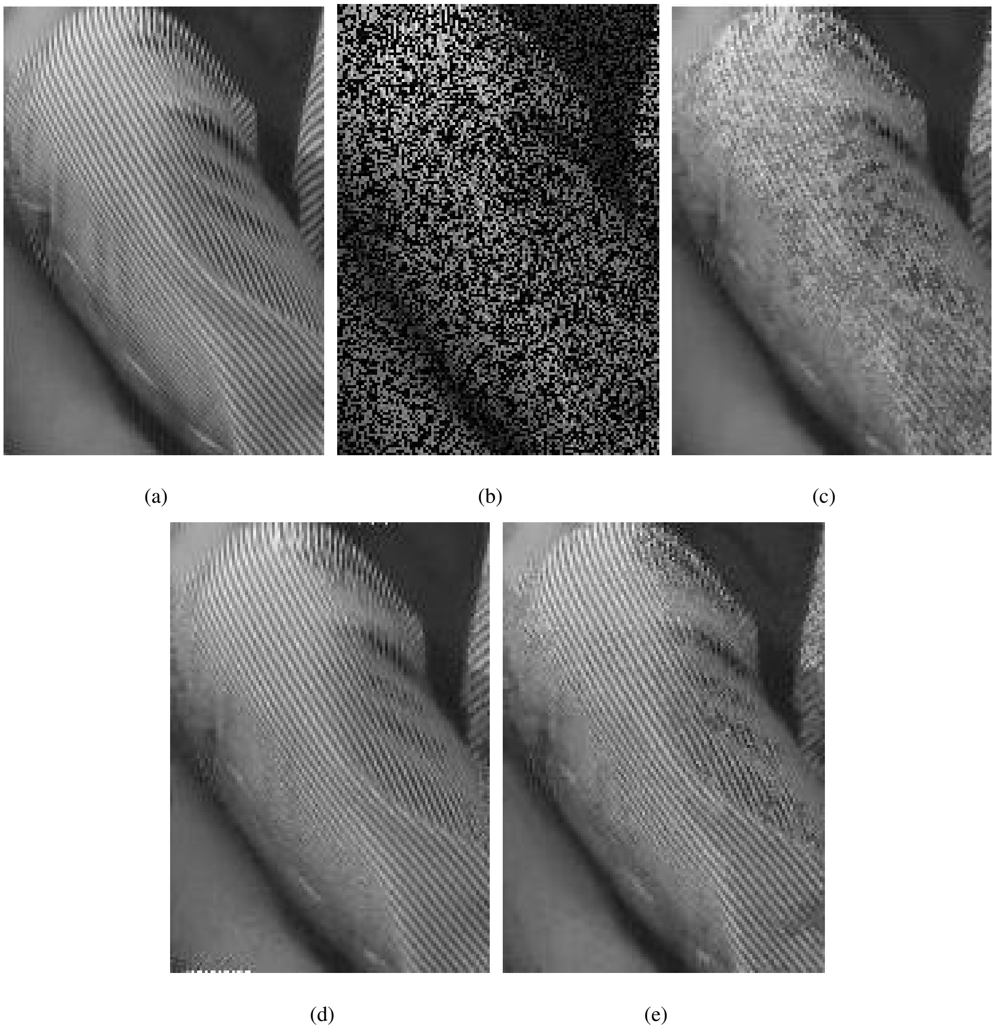


Figure 2: 50% Reconstruction Example. (a) Original image- Barbara. (b) Damaged image- Barbara. (c) Local TV inpainting- PSNR 23.6049. (d) Nonlocal TV inpainting- PSNR 27.8196. (e) The graph MBO method - PSNR 27.1651

The optimization problem we consider consists of minimizing the following energy:

$$E(u) = \epsilon\langle u, L_s u\rangle + \frac{1}{\epsilon}\sum_i W(u_i) + \sum_i \frac{\mu}{2}\lambda(x_i)||u_i - \hat{u}_i||^2_{L_2}, \tag{17}$$

encountered also in [67, 111, 112]. The first two terms of (17) 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:

$$W(u_i) = \prod_{k=1}^{\hat{n}} \frac{1}{4}||u_i - e_k||^2_{L_1}. \tag{18}$$

The last term of (17) 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 (17), 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:

$$\frac{\partial u}{\partial t} = -\epsilon L_s u - \frac{1}{\epsilon}W^{'}(u) - \mu\lambda(x)(u - \hat{u}). \tag{19}$$

This is the Allen-Cahn equation [5,61] with fidelity term with the differential operator Δu replaced by a more general graph operator $-L_s$ [99]; when $\epsilon \to 0$, the solution to the Allen-Cahn equation approximates motion by mean curvature [114]. Note that in the last term of (19), the product is meant to be calculated on each node.

In [67], the authors propose an MBO scheme to solve (19). We slightly modify this scheme to solve (19) in the formulation of our semi-supervised method.

We now present the semi-supervised algorithm, detailed in Figure 3, and which is based on that of [67]. 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:

$$\frac{u^{n+\frac{1}{2}} - u^n}{dt} = -L_s u^n - \mu\lambda(x)(u^n - \hat{u}), \tag{20}$$

* Step 2: Threshold

$$u_i^{n+1} = e_r, r = \arg\max u_i^{n+\frac{1}{2}} \tag{21}$$

for all $i \in \{1, 2, ..., N\}$, 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 [144].

Step 1 can be computed very efficiently and simply by using the eigendecomposition of L_s, which is:

$$L_s = X \Lambda X^T, \tag{22}$$

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

$$u^n = X a^n, \qquad \mu \lambda(x)(u^n - \hat{u}) = X d^n \tag{23}$$

and equate coefficients, we can formulate step 1 in the MBO scheme as solving for the coefficients a_k^{n+1}:

$$a_k^{n+1} = (1 - dt\lambda_k) \cdot a_k^n - dt \cdot d_k^n, \tag{24}$$

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, (24) 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 [79], 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 Graph MBO 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

Figure 3: Semi-Supervised Algorithm.

```
Semi-Supervised Algorithm

Require: a data set of $N$ points (embedded in a graphical framework $G = (V, E)$) to be classified into $\hat{n}$
  classes, parameters $dt$ and $\mu$ and threshold value $\delta$.
  $\star$ Initialize $u^0$ and compute $d^0$.
  $\star$ Calculate $m \ll N$ 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
    $\star$ $a^n = X^T \cdot u^n$
    $\star$ $a_k^{n+1} = (1 - dt\lambda_k) \cdot a_k^n - dt \cdot d_k^n$ for $k = 1$ to $k = m$
    $\star$ Compute $u^{n+\frac{1}{2}}$ via $u^{n+\frac{1}{2}} = Xa^{n+1}$
    $\star$ Compute $d^{n+1}$ via (23)
    $\star$ $u_i^{n+1} = e_r, r = \arg\max u_i^{n+\frac{1}{2}}$ for $i = 1$ to $i = N$
    $n \leftarrow n + 1$
  end while
  $\star$ If $u_f$ is the final iterate of $u$, let $C_i = \arg\max(u_f)_i$
  return matrix C denoting the final class of all nodes
```

model [119], 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 [57, 146]:

$$E^{MS}(\Phi, \{c_r\}_{r=1}^{\hat{n}}) = |\Phi| + \mu \sum_{r=1}^{\hat{n}} \int_{\Omega_r} (f - c_r)^2, \tag{25}$$

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 [80] for hyperspectral images:

$$MS(u, \{c_r\}_{r=1}^{\hat{n}}) = \frac{1}{2}|u|_{TV} + \mu \sum_{r=1}^{\hat{n}} \langle ||f - c_r||^2, u_{\star,r} \rangle, \tag{26}$$

where u is the class assignment matrix (described in the previous section) in which each row is an element of the Gibbs simplex (16). The length of the contour is estimated by the total variation (TV) of the assignment matrix u. In (26), the term $||f - c_r||^2$ denotes an $N \times 1$ vector $(||f(x_1) - c_r||^2, ..., ||f(x_N) - c_r||^2)^T$ and the x_i $(i = 1, ...N)$ 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 (26), $\langle , \rangle$ 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 (26). This problem is essentially equivalent to the K-means method when μ approaches +∞.

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

$$c_r = \frac{\langle f, u_r \rangle}{\sum_{i=1}^{N} u_{\star,r}(x_i)}, \tag{27}$$

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 [89, 143]; thus, we modify (26) using a diffuse interface approximation:

$$E(u, c_r) = \epsilon \langle u, L_s u \rangle + \frac{1}{\epsilon} \sum_i W(u_i) + \mu \sum_{r=1}^{\hat{n}} \langle ||f - c_r||^2, u_{\star,r} \rangle. \tag{28}$$

Similarly to the procedure in Section 3,1, using gradient descent yields:

$$\frac{\partial u}{\partial t} = -\epsilon L_s u - \frac{1}{\epsilon} W^{'}(u) - \mu(||f - c_1^n||^2, ..., ||f - c_{\hat{n}}^n||^f 2). \tag{29}$$

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 [57]. A class of algorithms for the high order geometric motion of planar curves following a similar thresholding procedure can be found in [56].

We now present our unsupervised algorithm, detailed in Figure 4. 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

$$\frac{u^{n+\frac{1}{2}} - u^n}{dt} = -L_s u^n - \mu(||f - c_1^n||^2, ..., ||f - c_{\hat{n}}^n||^2). \tag{30}$$

* Step 2: Threshold

$$u_i^{n+1} = e_r, r = \arg\max u_i^{n+\frac{1}{2}} \tag{31}$$

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

* Step 3: Update c

$$c_r^{n+1} = \frac{\langle f, u_{\star,r}^{n+1} \rangle}{\langle 1, u_{\star,r}^{n+1} \rangle}. \tag{32}$$

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 $m \ll N$ 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:

$$u^{n+\frac{1}{2}} = X(1 - dt \cdot \Lambda)a^n - dt \cdot \mu(||f - c_1^k||^2, ..., ||f - c_{\hat{n}}^k||^2). \tag{33}$$

Due to the fact that, in practice, only the leading eigenvalues and eigenvectors need to be computed to obtain a good accuracy, (33) 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%.

3.3 Background on the Nyström Extension Technique

In the procedure of both the semi-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 [62, 62, 63], a matrix completion method often used in image processing applications, such as kernel principle component analysis [51] and spectral clustering [120]. This procedure performs much faster than many alternate techniques because it uses approximations based on calculations on small submatrices of the original large matrix. When the size of the matrix becomes very large, this method is especially valuable.

The Nyström extension method calculates an eigendecomposition of a smaller system of size $m \ll N$ and then expands the results back up to N dimensions. The computational complexity

Figure 4: Unsupervised Algorithm.

```
Unsupervised Algorithm

Require: 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 $\mu$.
  ⋆ Initialize $u^0$ and compute $a^0$ via (23).
  ⋆ Calculate $m \ll N$ smallest eigenvectors and eigenvalues of the symmetric graph Laplacian $L_s$. Let $X$=
  the eigenvector matrix, $\Lambda$= the diagonal matrix containing the eigenvalues.
  ⋆ Set $n = 0$.
  while $purity(u^n, u^{n-1}) < 99.99\%$ do
    ⋆ Compute $c_r^{n+1}$ via (32) for $r = 1$ to $r = \hat{n}$
    ⋆ Compute $u^{n+\frac{1}{2}}$ via (33)
    ⋆ $u_i^{n+1} = e_r, r = \arg\max(u^{n+\frac{1}{2}})_i$ for $i = 1$ to $i = N$
    ⋆ Compute $a^{n+1}$ via (23)
    ⋆ $n \leftarrow n + 1$
  end while
  ⋆ If $u_f$ is the final iterate of $u$, let $C_i = \arg\max(u_f)_i$
  return matrix C denoting the final class of all nodes
```

is almost $O(mN)$. We can set $m \ll N$ 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.

Based on the definition in formula (6), if λ is an eigenvalue of $\hat{W} = D^{-\frac{1}{2}} W D^{-\frac{1}{2}}$, then $1 - \lambda$ is an eigenvalue of L_s, and the two matrices have the same eigenvectors. We formulate a method to calculate the eigenvectors and eigenvalues of $\hat{W}$ and thus of L_s.

Let w be the similarity function, λ be an eigenvalue of W, and ϕ its associated eigenvector. The Nyström method approximates the eigenvalue equation

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

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

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

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

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

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

This system of equations cannot be solved directly since the eigenvectors are not known. To overcome this problem, the m eigenvectors of W are approximated using calculations involving submatrices of W.

Let W_{XY} be defined as

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

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

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

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 mN computations versus N^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 in the literature. See Chapter 5 for details.

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

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

$$\begin{aligned} d_X &= W_{XX}1_L + W_{XY}1_{N-L}, \\ d_Y &= W_{YX}1_L + (W_{YX}W_{XX}^{-1}W_{XY})1_{N-L}. \end{aligned} \tag{34}$$

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 in the following manner to obtain $\hat{W}_{XX}$ and $\hat{W}_{XY}$:

$$\begin{aligned} \hat{W}_{XX} &= W_{XX}./(s_X s_X^T), \\ \hat{W}_{XY} &= W_{XY}./(s_X s_Y^T), \end{aligned} \tag{35}$$

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

It is shown in [15] that if we have the eigendecomposition of two small matrices

$$\hat{W}_{XX} = B_X \Gamma B_X^T \tag{36}$$

and

$$\hat{W}_{XX} + \hat{W}_{XX}^{-1/2} \hat{W}_{XY} \hat{W}_{YX} \hat{W}_{XX}^{-1/2} = A^T \Xi A, \tag{37}$$

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

$$\Phi = \left[\begin{array}{c} B_X \Gamma^{1/2} B_X^T A \Xi^{-1/2} \\ \hat{W}_{YX} B_X \Gamma^{-1/2} B_X^T A \Xi^{-1/2} \end{array} \right].$$

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

Therefore, the efficiency of the Nyström extension method lies with the fact that when computing the eigenvalues and eigenvectors of an $N \times N$ matrix, where N is large, it approximates them using calculations involving only much smaller matrices, the largest of which has dimension $N \times m$, where m is small.

When the graph is sparse and is of moderate size, around 5000×5000 or less, they Rayleigh-Chebyshev procedure outlined in [6] is a proper eigensolver. It is a modification of an inverse subspace iteration method that uses adaptively determined Chebyshev polynomials. The procedure is also a robust method that converges rapidly and that can handle cases when there are eigenvalues of multiplicity greater than one. When the graph is very large, such as in the case of image segmentation, the Nyström extension method is used.

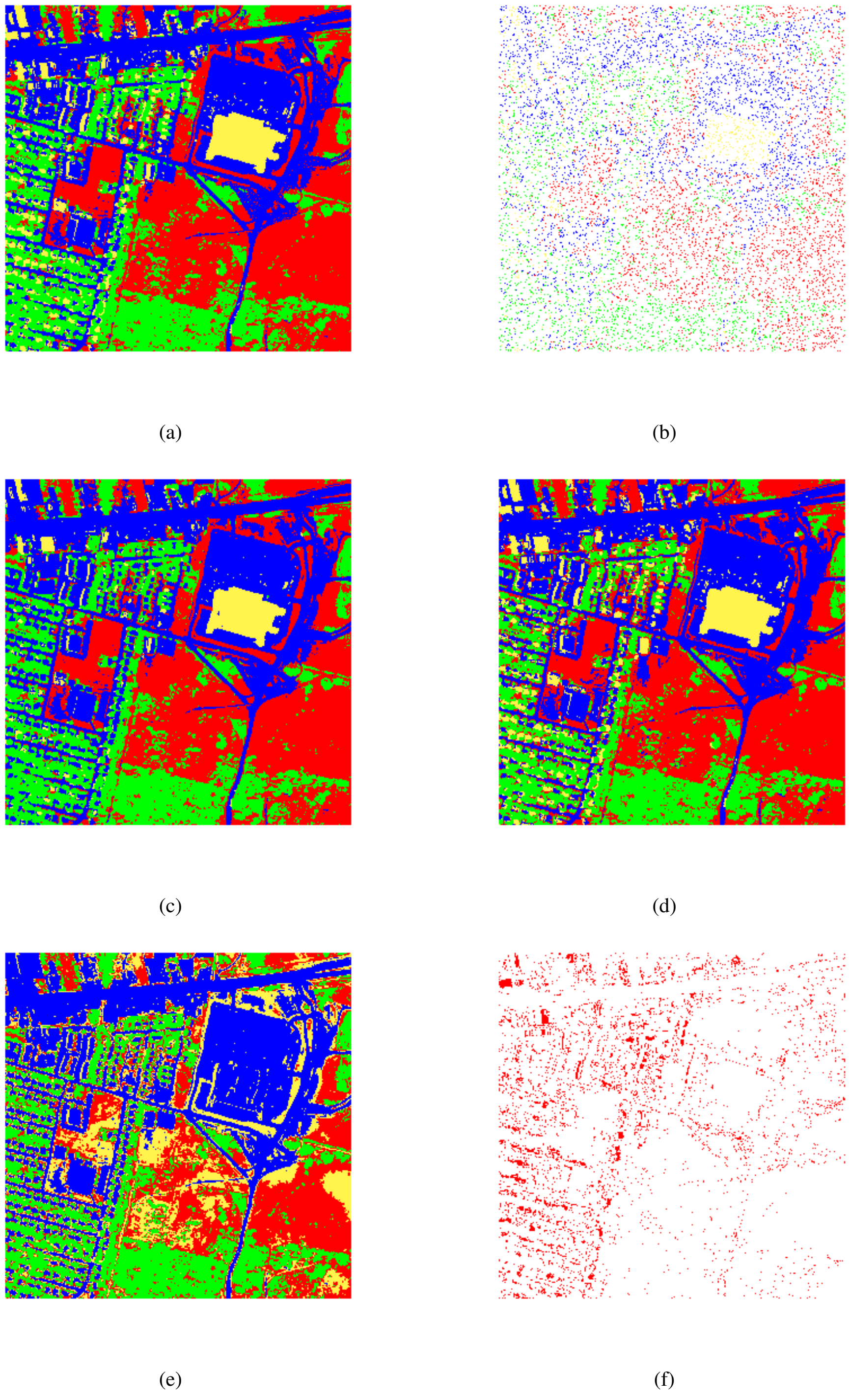
CHAPTER 4

Hyperspectral Image Classification

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. In [113], the graph MBO method is introduced for hyperspectral image classification. The unsupervised graph MBO classification method for hyperspectral image is proceeded in [80]. The first main contribution of this thesis is the full development of both algorithms for broad range of high-dimensional images including online parallelized code. In this chapter, we present the application of the two graph-based classification methods to various hyperspectral imagery datasets and we have developed and published online codes and demos for these problems [107]. Using the full dimensionality of the data, we consider a similarity graph based on pairwise comparisons of pixels. We also develop a parallel version of the Nyström extension method to randomly sample the graph to construct a low rank approximation of the graph Laplacian. The details of the parallelization is discussed in *Chapter 5*. 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. 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.

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



(a) (b)

(c) (d)

(e) (f)

Figure 5: The classification result of the semi-supervised and unsupervised graph MBO methods on the Urban data for four classes. (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.

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 [161], which contains 4 classes with end members corresponding to asphalt, grass, tree and roof, respectively. The goal is to accurate 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 Fig. 5. 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.

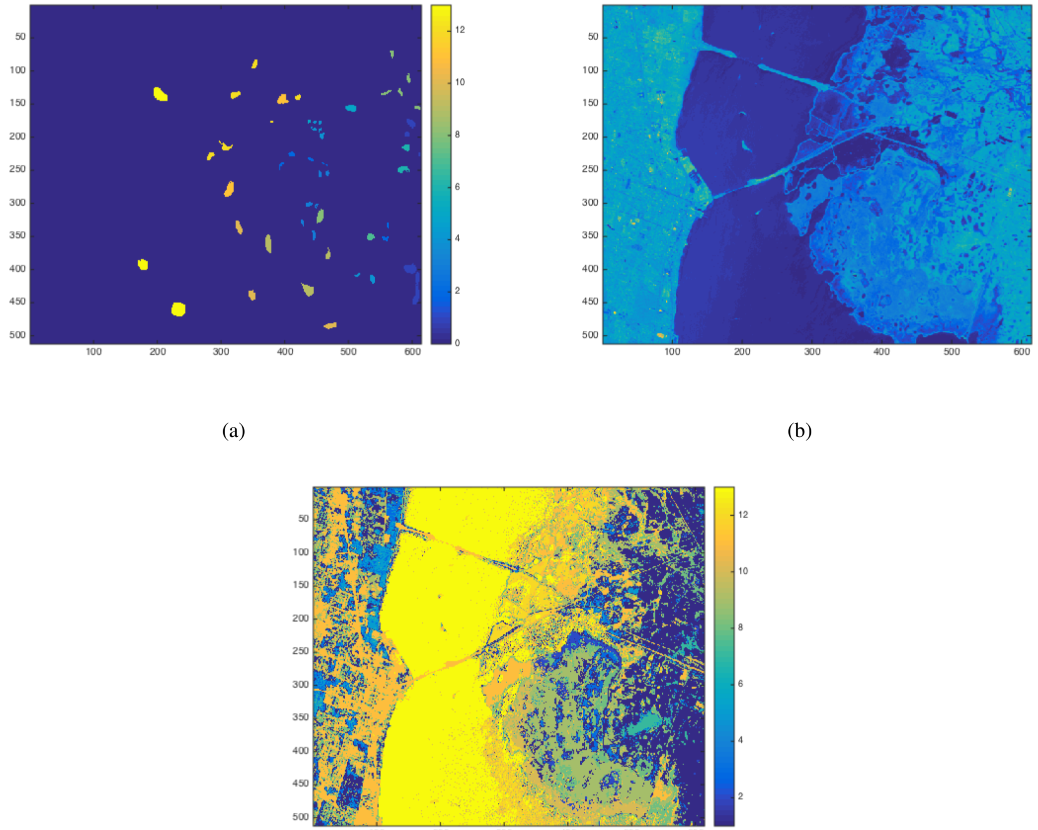
4.2 Kennedy Space Center Data

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 each 10 nm apart 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 Fig. 6(a). The image of band 50 is shown in Fig. 6(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 Fig. 6(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

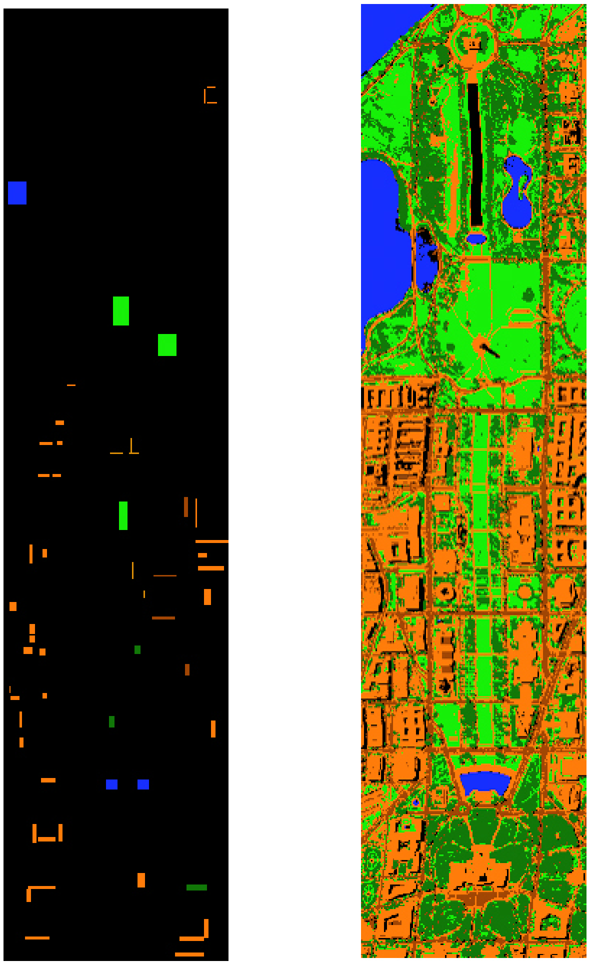


(a)

(b)

(c)

Figure 6: The classification result the semi-supervised graph MBO method on the Kennedy Space Center data. (a) Ground truth. (b) The image of band 50. (c) The classification result of 13 classes.



(a)

(b)

Figure 7: The classificaction result of the semi-supervised graph MBO method on the DC mall data. (a) Ground truth. (b) The classification result of 7 classes.

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 Fig. 7(a). We use 10 pixels per class for fidelity and apply our semi-supervised algorithm on this data set. The result is shown in Fig. 7(b), and the overall accuracy is 98.11%. We note that when 20 pixels 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 Fig. 8(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 Fig. 8(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 [27]. Segmentation of the gas is difficult due to the diffusive nature of the cloud. The use of hyperspectral imagery provides non-visual data for this problem, allowing for the utilization of a richer array of sensing information. Detecting chemical plumes in the atmosphere is a problem that can be applied to many areas, such as defense, security and environmental protection. If the airborne toxins are identified accurately, one can combat the use of chemical gases as weapons, prevent fatalities due to accidental leakage of toxic gases and avoid contamination of the atmosphere. Identification of harmful gases with high fidelity is needed to provide warnings in threatening situations. In these grave scenarios, it is crucial to accurately track the diffusion of dangerous plumes into atmosphere. Laboratory measured signatures of toxic chemicals are available to assist in chemical plume identification. However, testing and training data is not readily available due to the inherent danger of these real world situations. Instead, open air testing with surrogate chemicals is conducted to study the diffusion of chemical plumes. The developed plume detection methods must meet strict requirements to ensure the fidelity of a detector. The plume data set has been studied in [113] and [80] using the graph MBO methods. These two conference papers only worked with seven frames of the video data due to the computational limit of laptop. We apply the graph MBO methods on the full video sequence which contains 329 frames using the high performance computing techniques and parallelized algorithms. The parallelization details are discussed in *Chapter 5*. For the semi-supervised algorithm, we only use 8% fidelity (training) data, which is much less than [113]. We also normalize the feature vectors

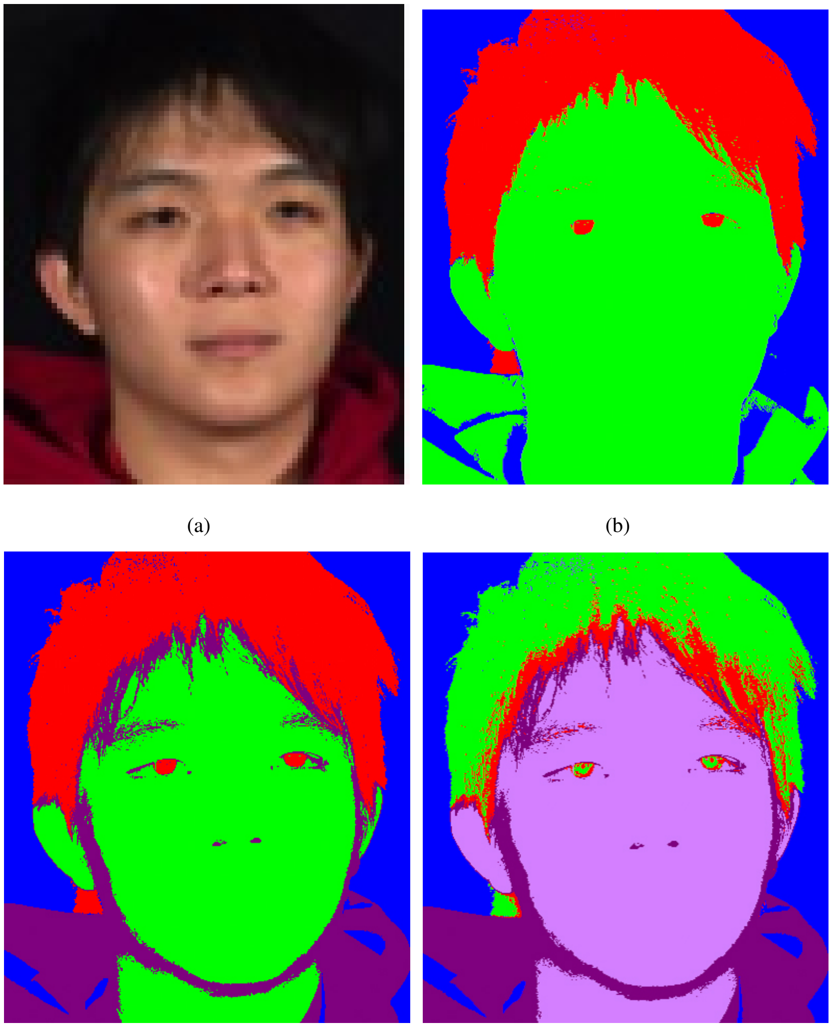


Figure 8: The classification result of the unsupervised graph MBO method for three, four, and five classes of the face data. (a) The male face image. (b) The classification result of three classes. (c) The classification result of four classes. (d) The classification result of five classes.

before using the unsupervised algorithm and get better classification result than [80].

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 with 11,700 nm with a spacing of 30 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 Fig. 9. To eliminate the flicker between frames, the Midway equalization method is used in [68]. 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 [68] 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 [142], a binary partition tree method is used to retrieve the real location and the extent of the plume. Moreover, the author of [138] proposes two ways to compute meaningful eigenvectors of the graph Laplacian. Other detection methods for hyperspectral plumes include [78] (MWIR) and [103] (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 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 [113]. We show the 2^{nd} to 5^{th} eigenvectors of frame 30 in aa12 Victory video set in Fig. 10. 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 Fig. 11.

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 Fig. 12 for the semi-supervised algorithm and Fig. 13 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 Fig. 14, 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

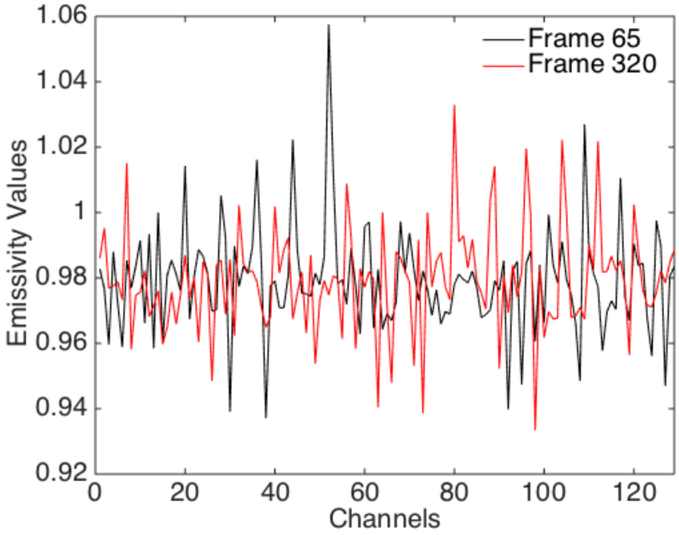


Figure 9: Emissivity value of one fixed pixel of frame 65 and frame 320.

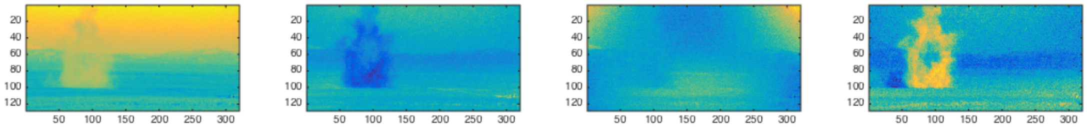


Figure 10: 2nd to 5th eigenvectors of the 30th frame.

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.

4.6 Nonlocal Means Method for RGB image

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

The nonlocal means method is widely used in image processing. Zhou and Schölkopf in their papers [160] [159] [158] formulated a theory of nonlocal operators that is related to the discrete graph Laplacian described in section 2.1. Buades, Coll, and Morel applied this nonlocal theory to denoising algorithms in their work [31]. Osher and Gilboa proposed using nonlocal operators to define functionals involving the TV semi-norm for various image processing applications in their

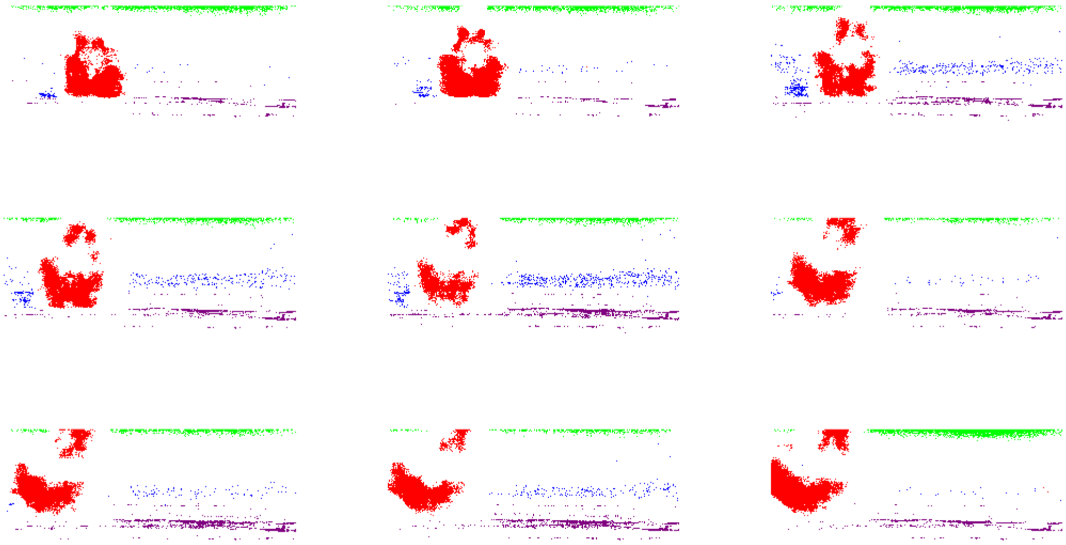


Figure 11: 8% of aa12 Victory data selected to be the fidelity region by thresholding the eigenvectors, frames 29-37.

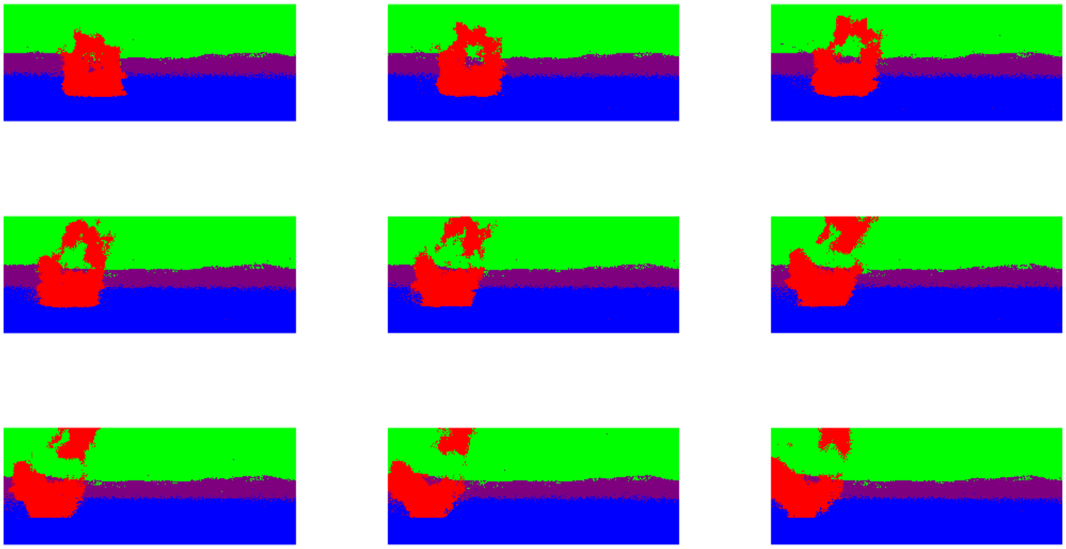


Figure 12: Classification result of the semi-supervised algorithm (frames 29-37 of the aa12 Victory video).

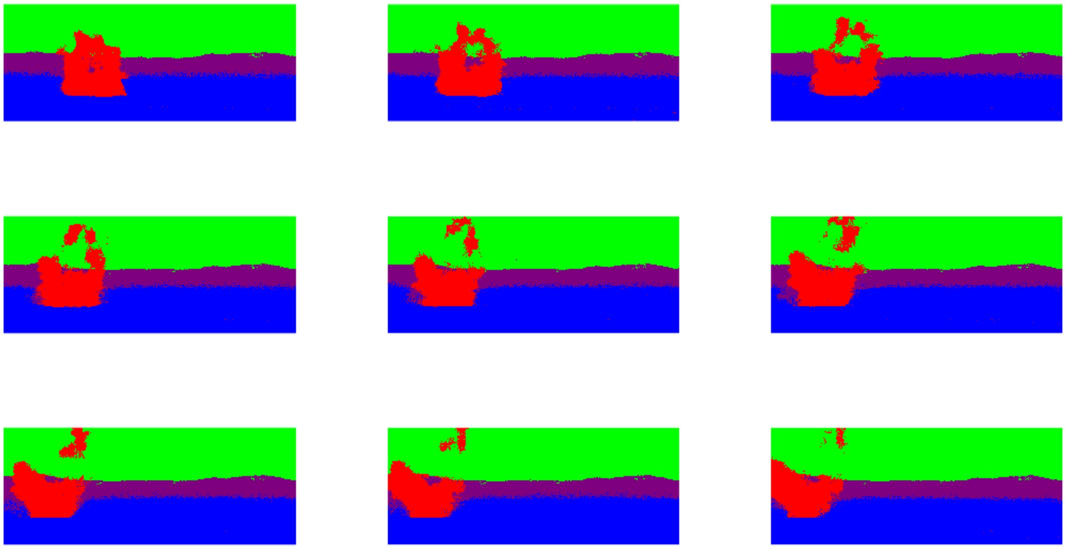


Figure 13: Classification result of the unsupervised algorithm (frames 29-37 of the aa12 Victory video).

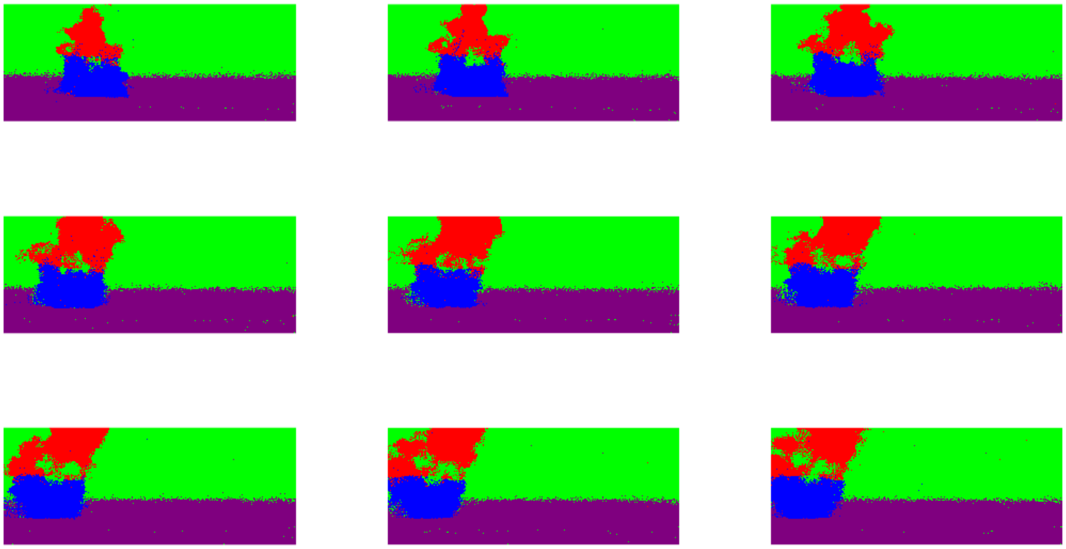


Figure 14: Classification result of spectral clustering with K-means (frames 29-37 of the aa12 Victory video).

work [69, 70].

In our work, we use the nonlocal means method to compute the feature vector of each pixel as follows:

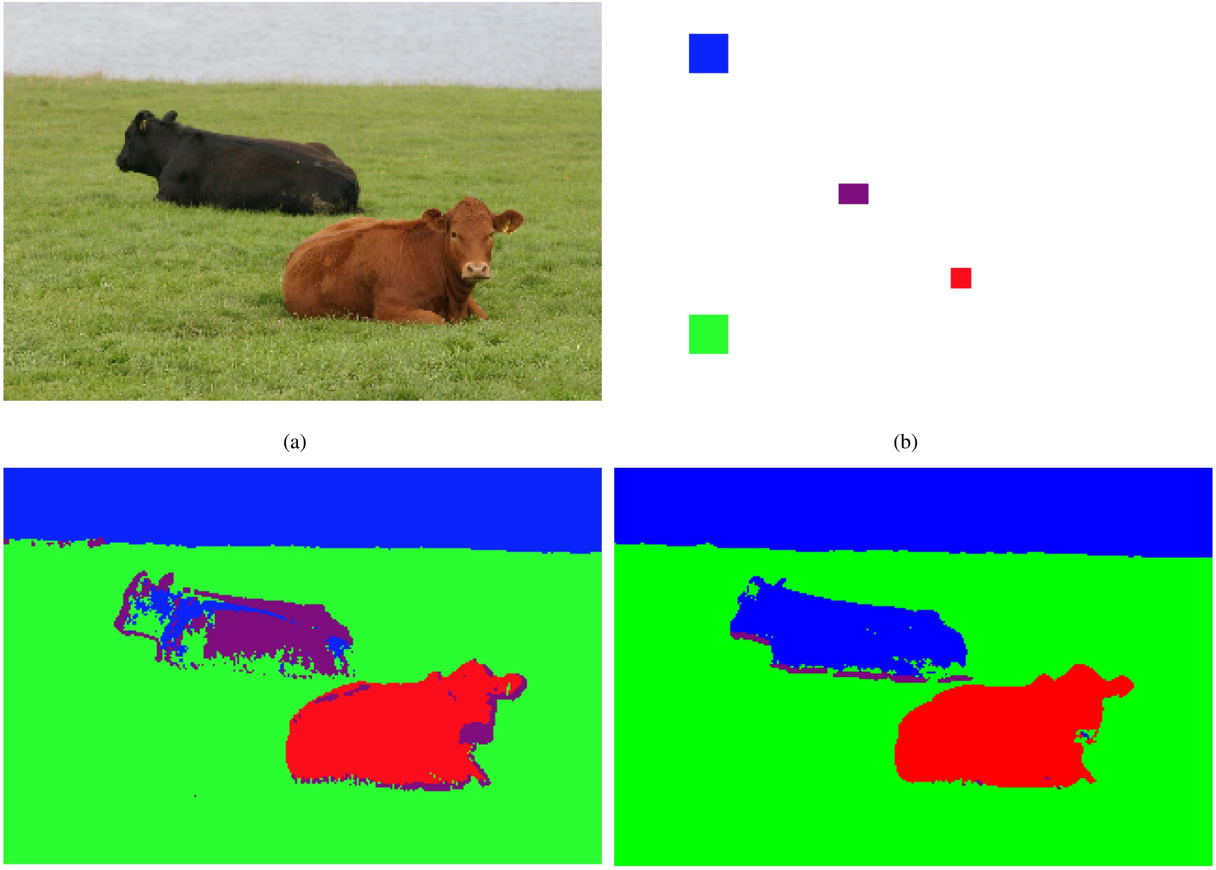
Nonlocal Means Method

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

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 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 Fig. 15. Fig. 15a 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, we select some pixels with known labels to be the fidelity. The fidelity is shown in Fig. 15b and the classification result of the semi-supervised algorithm is shown in Fig. 15c. 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 Fig. 15d. 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



(a)

(b)

(c)

(d)

Figure 15: Classification result of the nonlocal method on the RGB Image of Two Cows. (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.

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 the 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 potion of known labels for all data sets. We also compared the un-supervised 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 for the plume data set which is shown in Figure 14.

	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.

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 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.

	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

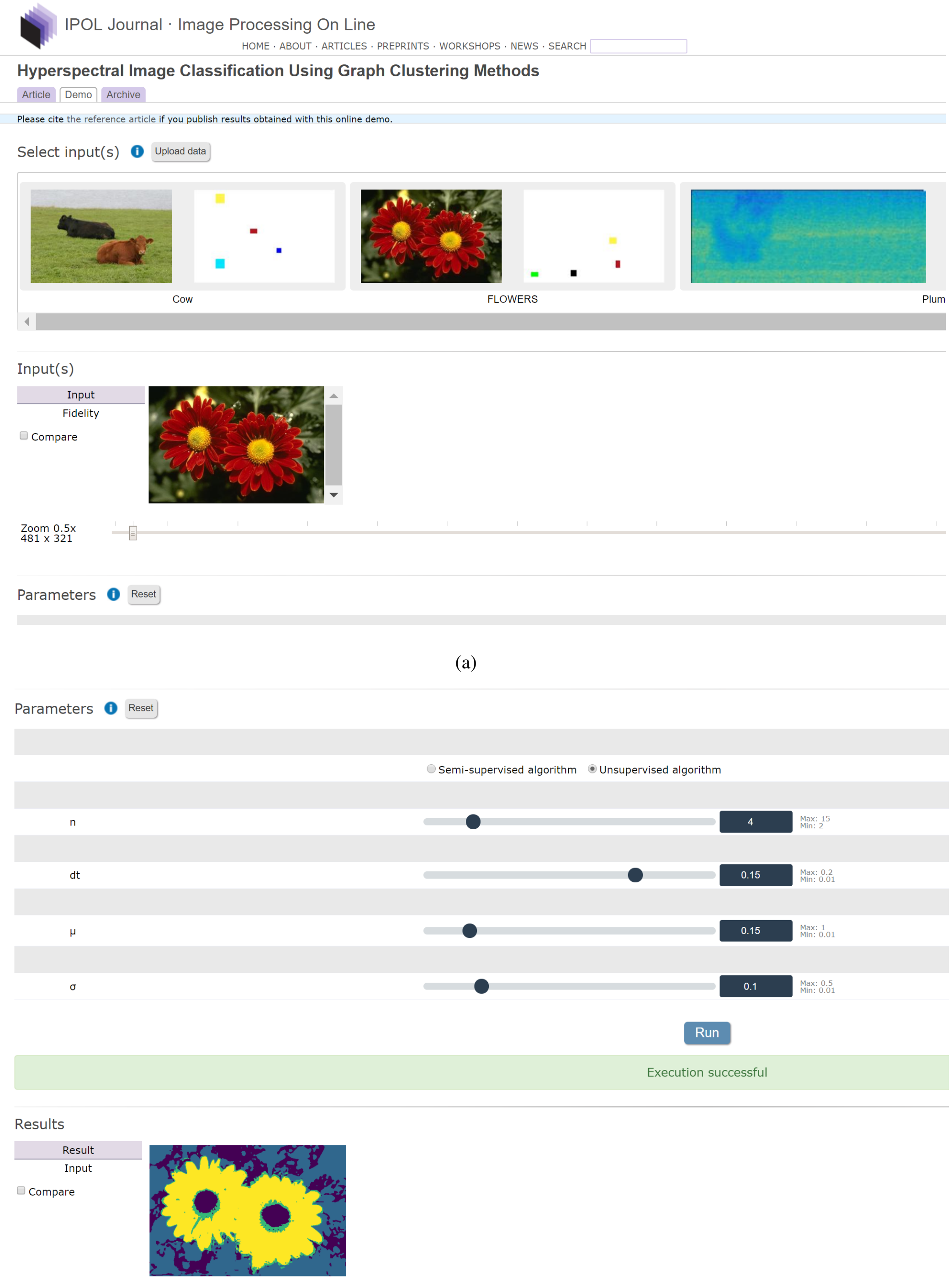
Table 2: Run time summary of hyperspectral images and videos (in seconds) on IPOL server Purple.

4.8 Online Demo

We have developed the parallelized codes for the hyperspectral image and RGB image classification discussed in the previous sections in this chapter. We have published the codes on the Image Processing On Line (IPOL) and developed an online demo for users. IPOL is a research journal of image processing and image analysis which emphasizes the role of mathematics as a source for algorithm design and the reproducibility of the research. Each article contains a text on an algorithm and its source code, with an online demonstration facility and an archive of experiments. Text and source code are peer-reviewed and the demonstration is controlled. IPOL is an Open Science and Reproducible Research journal.

The demo is shown in Figure 16. The users can choose the images we provide for a quick test of the classification results. They can also upload their own images and run our codes for classification. We provide both the semi-supervised algorithm and the unsupervised algorithm for the users to choose. To use the semi-supervised algorithm, the users needs to upload both the image and the fidelity (training data with labels). If the user doesn't upload any fidelity data, the demo automatically use the unsupervised algorithm for the classification.

The users can also tune the parameters of the algorithms. There are four parameters – n, dt, μ and σ. n is the number of classes. The users can set this parameter based on the known number or their observations of the possible number of classes in the image. dt is the amount of time for which a diffusion (heat equation + forcing term) takes place for one iteration of the algorithm. It is also known as the learning rate in machine learning language. Users can tune it in the range of (0.01, 0.2). We set the default value to be 0.15. μ is the parameter before the fidelity term, for semi-supervised algorithm, larger μ means you rely more on the known labels. For the unsuper-



IPOL Journal · Image Processing On Line

HOME · ABOUT · ARTICLES · PREPRINTS · WORKSHOPS · NEWS · SEARCH

Hyperspectral Image Classification Using Graph Clustering Methods

Article | Demo | Archive

Please cite the reference article if you publish results obtained with this online demo.

Select input(s) Upload data

Cow | FLOWERS | Plum

Input(s)

Input

Fidelity

□ Compare

Zoom 0.5x
481 x 321

Parameters Reset

(a)

Parameters Reset

○ Semi-supervised algorithm ● Unsupervised algorithm

n		4	Max: 15 Min: 2
dt		0.15	Max: 0.2 Min: 0.01
μ		0.15	Max: 1 Min: 0.01
σ		0.1	Max: 0.5 Min: 0.01

Run

Execution successful

Results

Result

Input

□ Compare

(b)

Figure 16: Online demo of the hyperspectral image classification on IPOL.

vised algorithm, as μ goes larger, the algorithm is more close to the K-means algorithm. σ is the parameter in the definition of weights. Larger sigma gives smaller weights between two nodes. Different images may require different parameters in order to get the best classification results.

After the users hit the “Run” button, the IPOL server runs the parallelized codes on its multi-core server. The run time is usually several seconds for one image. After running, the classification result is shown on the webpage. Users may choose to download the image or the class labels in text format. Since our method is non-convex, the results from different runs may vary due to the random initialization. The users may need to run several times for one set of parameters to check the best classification results.

CHAPTER 5

Parallelization of the Graph MBO methods

We investigate the OpenMP parallelization and optimization of the two graph classification algorithms. The results of this Chapter are published in [106]. The new algorithms provide significant accuracy and performance advantages over traditional data classification algorithms in serial mode. The methods leverage the Nyström extension to calculate eigenvalue/eigenvectors of the graph Laplacian. This is a self-contained module that can be used in conjunction with other graph-Laplacian based methods such as spectral clustering. We use performance tools to collect the hotspots and memory access of the serial codes and use OpenMP as the parallelization language to parallelize the most time-consuming parts. Where possible, we also use library routines. We then optimize the OpenMP implementations and detail the performance on traditional supercomputer nodes (in our case a Cray XC30), and test the optimization steps on emerging testbed systems based on Intel's Knights Corner and Landing processors. We show both performance improvement and strong scaling behavior. A large number of optimization techniques and analyses are necessary before the algorithm reaches almost ideal scaling.

5.1 Math Library Usage and Optimizations

All the data are in matrix form and there are intensive linear algebra calculations. Also, we apply Singular Value Decomposition (SVD) to two small matrices. So, we make use of the LAPACK (Linear Algebra PACKage) and BLAS (Basic Linear Algebra Subprograms) libraries in the codes. The LAPACK provides routines for the SVD and the BLAS provides routines for vector-vector (Level 1), matrix-vector (Level 2) and matrix-matrix (Level 3) operations. BLAS and LAPACK are also highly vectorized and multithreaded using OpenMP.

We use the Intel Performance Tool VTune Amplifier to analyze the performance and find bottlenecks [2]. The hotspots collection shows some computationally expensive parts are related to calculating the inner product of two vectors. In the unsupervised graph MBO algorithm, this operation occurs when calculating the matrix P and takes 84% of the run time. Also, it occurs when calculating the matrix W_{XY} in the Nyström extension algorithm and takes 90% of the run time. We optimize this procedure by forming all the vectors into matrices and doing the inner product of two matrices. In this way, we make use of BLAS 3 (matrix-matrix) instead of BLAS 1 (vector-vector). The part of calculating the matrix P in the unsupervised algorithm is 22.5× faster using BLAS 3. This optimization is based on the fact that BLAS 1,2 are memory bound and BLAS 3 is computational bound [47, 91].

5.2 Parallelization of the Nyström Extension

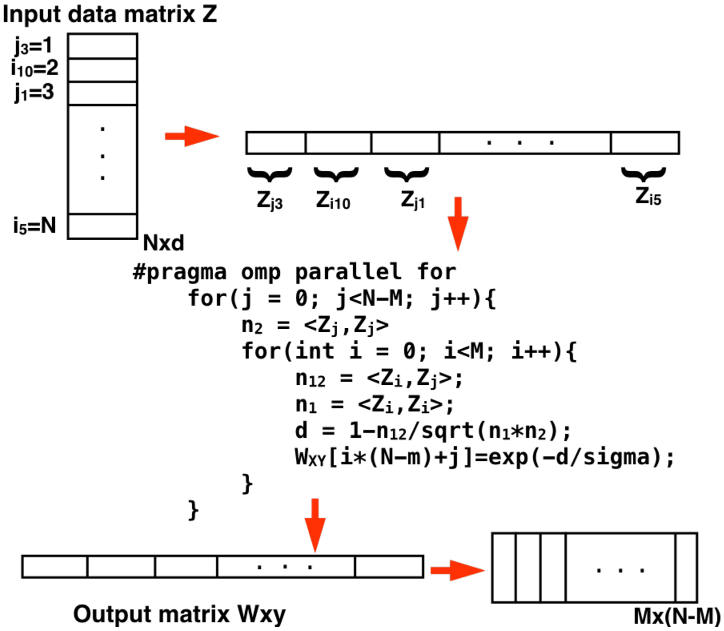
Parallelization of these two classification algorithms involves a parallel for. It is critical to further optimize the OpenMP implementation to get nearly ideal scaling. We detail this process using more complex features of OpenMP such as SIMD and vectorization. Then we use the uniform sampling and chunk of data to get the best performance.

We consider the data set, described in more detail in [28], composed of hyperspectral video sequences recording the release of chemical plumes at the Dugway Proving Ground. We use the 329 frames of the video. 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 13,475,840. Since we are dealing with very large data set we choose binary form for smaller storage space and faster I/O. Our test data is 13.91 GB in binary form and the I/O is 36.8× faster than the txt format when testing on Cori Phase I.

We conduct our experiments on single nodes of systems at the National Energy Research Scientific Computing Center (NERSC). Cori Phase I is the newest supercomputer system at NERSC. The system is a Cray XC based on the Intel Haswell multi-core processor. Each node has 128 GB of memory and two 2.3 GHz 16-core Haswell processors. 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. Peak performance per node is about 1.2 TFlop/s and peak bandwidth is about 120 GB/s. The resultant machine balance of 10 flops per byte strongly motivates the use of BLAS 3 like computations. Cori Phase II will be a Cray XC system based on the second generation of Intel Xeon Phi Product Family, called Knights Landing (KNL) Many Integrated Core (MIC) Architecture. The test system available to us now features 64 cores with 1.3 Ghz clock frequency (Bin-3 configuration) with support for four hyper-threads each. Each core additionally has two 512bit-wide vector processing units. Additionally, the chip is equipped with 16 GB on-package memory shared between all cores. it is referred to as HBM or MCDRAM with a maximum bandwidth of 430 GB/s measured using the STREAM triad benchmark. The 512 KB L2 cache is shared between two cores (i.e. within a tile) and the 16 KB L1 cache is private to the core. Furthermore, the single socket KNL nodes are equipped with 96 GB DDR4 6-channel memory with a maximum attainable bandwidth of 90 GB/s.

5.3 OpenMP Parallelization

Analysis with VTune shows that the most time consuming phase of both two classification algorithms is construction of W_{XY} in the Nyström extension procedure. This phase is a good candidate for OpenMP parallelization because each element of W_{XY} can be computed independently. The procedure of calculating W_{XY} is shown in Fig. 17. We form the data in a N by d matrix Z. Each row of Z corresponds to a data point and it's a vector of dimension d. In computation, we store Z in an array in row major. We randomly select M rows to form the sampled data set $X = \{Z_i\}_{i=1}^{M}$. The other rows form the data set $Y = \{Z_j\}_{j=1}^{N-M}$. Then we use the nested for-loop to calculate the values of W_{XY} by the formula (1). We then put the corresponding value in an array which represent the M by $N - M$ matrix W_{XY}.



```
#pragma omp parallel for
    for(j = 0; j<N-M; j++){
        n2 = <Zj,Zj>
        for(int i = 0; i<M; i++){
            n12 = <Zi,Zj>;
            n1 = <Zi,Zi>;
            d = 1-n12/sqrt(n1*n2);
            WXY[i*(N-m)+j]=exp(-d/sigma);
        }
    }
```

Figure 17: The procedure for calculating W_{XY}:

5.3.1 Reordering Loops

We have tested re-ordering loops as a means to optimize the algorithm. With analysis, we notice the j-loop is far larger than the i-loop. There are still two ways to do the parallelization. One way is to parallelize the j-loop as inner loop and the other way is to parallelize the j-loop as outer loop. We tried both ways and compared the results.

Step A: Parallelizing the inner j-loop

```
for $i = 0; i < M; i++$
    $n_1 =< Z_i, Z_i >$
    #pragma omp parallel for
    for $j = 1 : N - M$
        $n_{12} =< Z_i, Z_j >$
        $n_2 =< Z_j, Z_j >$
        $d = 1 - n_{12}/\sqrt{n_1 \cdot n_2}$
        $W_{XY}(i, j) = exp(-d/\sigma)$
    end
end
```

Step B: Parallelizing the outer j-loop

```
#pragma omp parallel for
for $j = 1 : N - M$
    $n_2 =< Z_j, Z_j >$
    for $i = 1 : M$
        $n_{12} =< Z_i, Z_j >$
        $n_1 =< Z_i, Z_i >$
        $d = 1 - n_{12}/\sqrt{n_1 \cdot n_2}$
        $W_{XY}(i, j) = exp(-d/\sigma)$
    end
end
```

The results show that parallelizing the outer j-loop is much faster. The run time decreases by a factor of 7. This is because on Cori, each core has its own L1 and L2 cache. When parallelizing the outer j-loop, all the X_is can be read and reside on the L2 of each core and can be used repeatedly. If instead we parallelize the inner j-loop, there are more reads of the X_i and thus the calculation takes more time. Parallelizing the outer j loop also means each thread has more work to do, since the inner i-loop is also part of the j-loop. In this way less overhead and more load balance can be achieved. While if we parallelize the inner j-loop, not only each thread has less work and large load imbalance, but also there are multiple times of thread creation and overhead.

5.3.2 Vectorization and Chunk

We further optimize the OpenMP parallelization using vectorization. First, we notice, the norms of Z_is are computed repeatedly in the i-loop. So, we normalize all the Z_is in the previous step, calculating W_{XX}, and store all the normalized Z_is in a new matrix X_{mat}. Then we can calculate the inner product of each Z_j and all the Z_is (X_{mat}) all at once. This make use of BLAS 2 instead of the previous BLAS 1. Also, we can vectorize the loop when calculating W_{XY}. This optimization reduce the run time of calculating W_{XY} by a factor of 3.

Step C: Calculating W_{XY}, normalize and form all Z_is to X_{mat}

```
#pragma omp for
for $j = 1 : N - M$
    $n_2 =< Z_j, Z_j >$
    $n_{vec} = 1- < X_{mat}, Z_j > /\sqrt{n2}$
    #pragma omp simd aligned
    for $i = 1 : M$
        $W_{XY}(i, j) = exp(-n_{vec}/\sigma)$
    end
end
```

The Nyström extension algorithm is based on a random partition of the whole dataset Z into two disjoint data sets X and Y, where $X = \{Z_i\}_{i=1}^{M}$ and $Y = \{Z_j\}_{j=1}^{N-M}$ and $M \ll N$. Assuming we can uniformly partition the dataset, so that Z_is are evenly distributed, we can form chunks of Z_js to matrix and further optimize this calculation. The procedure is shown in Fig. 18. First, when calculating W_{XX}, we evenly sample Z_is and normalized them. We form the normalized Z_is to a matrix X_{mat}. Then all the data in between two consecutive Z_is are the chunk of Z_js. Since the chunk size is still very large, we further decompose each Y-chunk into sub-chunks. There are several considerations for choosing the sub-chunk size. If it is too small, we waste potential of combining expensive operations. If it is too large, the sub-chunk may run out of lower level

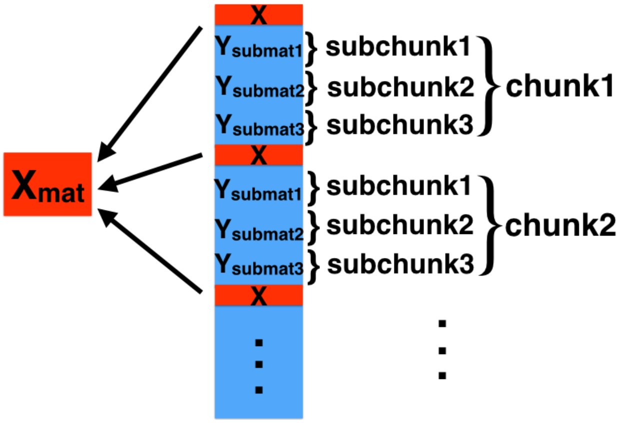


Figure 18: Uniform sampling and dividing Y into chunks and sub-chunks

cache and needs to be put into the higher cache levels, up to the point where they spill over into DRAM which may cause a substantial performance hit. The optimal value depends on the cache hierarchy, their respective sizes, their latency and so on. For a different architecture, one may consider choosing another value. We pick the the $subchunksize = 64$ when running the codes on Cori Phase I and it can be further optimized. Then for each sub-chunk, we calculate the Euclidean norm of each row and store them in a vector $n2_{vec}$. This calculation can be vectorized since calculating the norm of each row is independent. We further divide the norms by 1. We then calculate the matrix multiplication $X_{mat} \cdot Y_{submat}$ using BLAS 3 function DGEMM. The result is a $m \times subchunksize$ matrix $n12_{mat}$. It is the result of all the inner product of rows in X_{mat} and rows in Y_{submat}. Then we can vectorize the final calculation of values in W_{XY}.

Step D: Calculating W_{XY} using uniform sampling and chunked Y matrices

```
#pragma omp for collapse(2)
for $ychunk = 0; ychunk < m; ychunk++$
    for $j = chunkstart; j < chunkstop; j+ = subchunksize$
        #pragma omp simd aligned
        for $k = 0; k < subchunksize; k++$
            $n2_{vec}[k] =< Z_{j+k}, Z_{j+k} >$
            $n2_{vec}[k] = 1/\sqrt{n2_{vec}[k]}$
        end
        $n12_{mat} =< X_{mat}, Y_{submat_j} >$
        #pragma omp simd aligned
        for $i = 0; i < m; i++$
            for $k = 0; k < subchunksize; k++$
            $d = 1 - n12_{mat}[i, k] \cdot n2_{vec}[k]$
            $W_{XY}(i, j + k) = exp(-d/\sigma)$
        end
    end
end
```

In this uniform sampling, the chunk size is defined as $chunksize = floor(N/M)$. When M is not divisible by N, the last chunk is larger than the other chunks. Also, $subchunksize$ may not be divisible by $chunksize$. So the size of the last sub-chunk in each chunk needs to be adjusted. The procedure of uniform sampling gives good results as compared to the random sampling and further improves the performance by a factor of 1.7.

5.3.3 Thread Affinity

We also consider the effect of thread affinity. We choose the thread affinity setting as "OMP_PROC_BIND=scatt and "OMP_PLACES=cores (or thread)", because it uses one hardware thread per core. While if we use the thread affinity setting to be "OMP_PROC_BIND=close" and "OMP_PLACES=thread", it puts more threads on each physical core and leaving other cores idle, which affects scaling performance.

5.3.4 Experimental Results

Cori Phase I: we examined optimization steps on a single node of Cori Phase I. The run time decrease and scaling results of different steps of optimizing the OpenMP parallelization are shown in Fig. 19. We show the significant speed up of the Nyström loop part. In Step A, in addition to parallelizing the Nyström loop, we also use BLAS 3 optimization on the graph MBO algorithm. Since we use BLAS and LAPACK in the serial part of Nyström algorithm and the graph MBO algorithm, their run time also decrease when using multi-cores. We show the OpenMP thread scaling results on Cori Phase I in Fig. 20. Almost ideal scaling results are achieved. Each Cori Phase 1 node has two sockets (NUMA domain) and each socket has 16 cores. Although the absolute performance increases when using more than 16 threads on a single node, NUMA effect is observed that the scaling slows down due to remote memory access to a far NUMA domain.

Knight's Landing: we employed the same optimizations already used for the Haswell optimization with three exceptions: we have compiled the code with AVX-512 support to make use of the wider vector units as well as doubled the sub-chunk size as depicted in Fig. 18 accordingly.[1] Furthermore, we have enabled fast floating point model and imprecise divides with `-fp-model fast=2` and `-no-prec-div` respectively. The (strong) thread scaling of the various sections of the code is depicted in Fig. 21 for one hyper-thread per core. We found that this configuration delivered the best performance. Utilizing two or more hyper-threads significantly decreased the performance, especially that of the Nyström loop. We observe that our code obtains

[1] We have explored various sub-chunk sizes but found that twice the optimal Haswell value, i.e. 128 vectors, yield the best performance.

good strong scaling up to all 64 cores. We leave the hyper-threads analysis for further investigation and we are looking into improving scaling of step D.

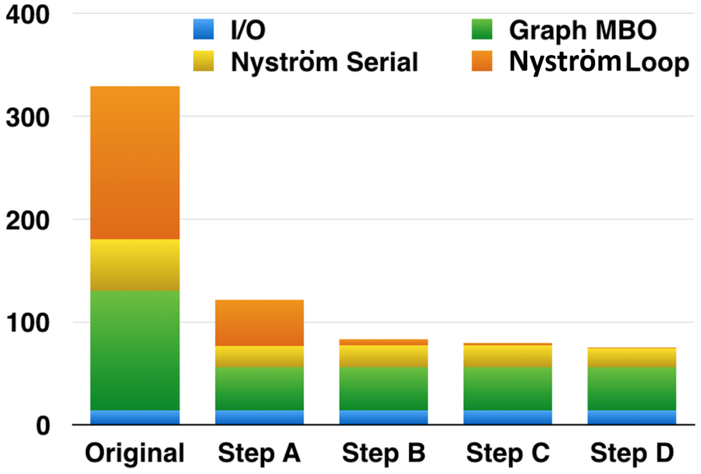


Figure 19: The run time of different optimization steps on Cori Phase I. Step A: parallelizing the inner j-loop and BLAS 3 optimization on Graph MBO. Step B: parallelizing the outer j-loop. Step C: normalizing and forming all Z_is to X_{mat}. Step D: using uniform sampling and chunked Y matrices.

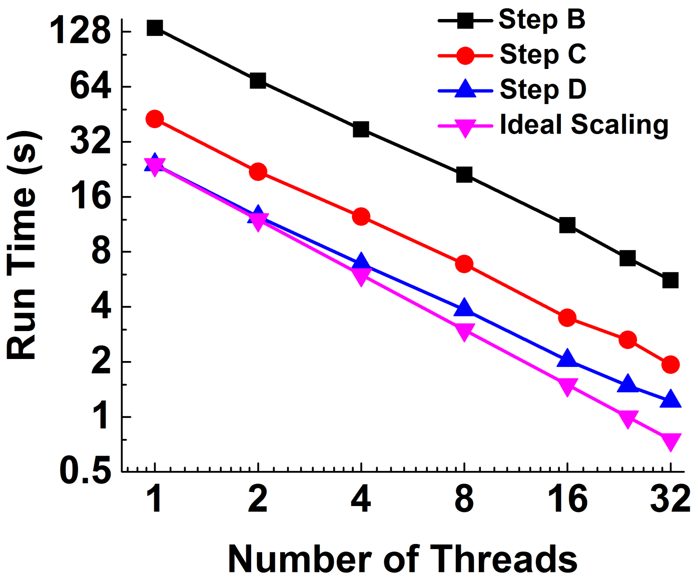


Figure 20: The scaling results of the OpenMP parallelization of the Nyström loop on Cori Phase I. The black line with squares, the red line with circles and the blue line with triangles show the scaling results of step B, C and D respectively. The pink line with upside down triangles shows the ideal scaling.

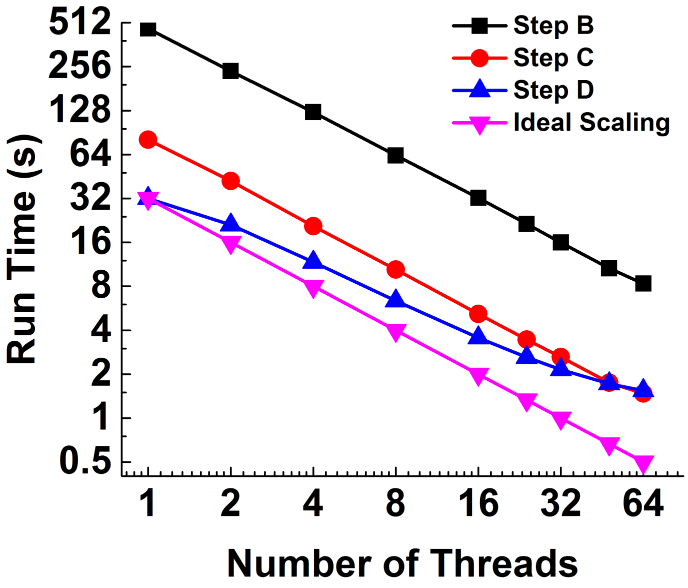


Figure 21: The scaling results of the OpenMP parallelization of the Nyström loop on our KNL test system. The black line with squares, the red line with circles and the blue line with triangles show the scaling results of step B, C and D respectively. The pink line with upside down triangles shows the ideal scaling. All values were obtained employing one hyper-thread per core.

5.4 Arithmetic Intensity and Roofline Model

Arithmetic intensity is the ratio of floating-point operations (FLOP's) performed by a given code (or code section) to the amount of data movement (Bytes) that are required to support those operations. Arithmetic intensity in conjunction with the Roofline Model [151] can be used to bound kernel performance and qualify performance in a manner more nuanced than percent-of-peak. Fig. 22 shows the result of using the Roofline Toolkit [3] to characterize the performance of a Cori Phase I node (full 32 cores). The resultant lines ("ceilings") are bounds on performance. Clearly, in order to attain high performance, one must design algorithms that deliver high arithmetic intensity.

In order to characterize the Nyström loop, we used Intel's Software Development Emulator Toolkit (SDE) to record FLOP's and Intel's VTune Amplifier to collect data movement when running on 32 cores of a Cori Phase I node [1, 50]. We can then compare the results to a theoretical estimate based on the inherent requisite computation and data movement.

As shown in Fig. 17, the memory access has two major components — one must read data

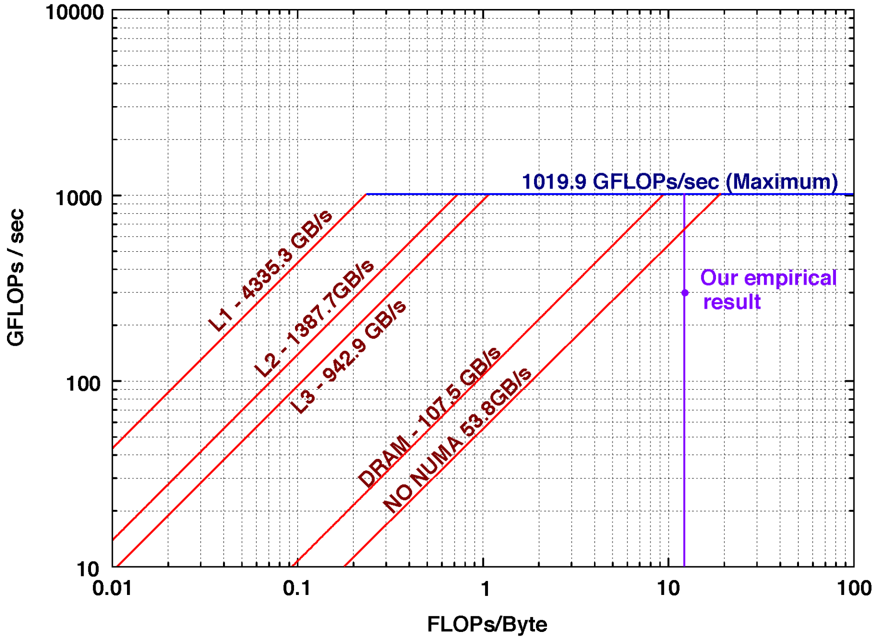


Figure 22: Empirical Roofline Toolkit results for a Cori Phase I node. Observe, DRAM bandwidth constrains performance for a wide range of arithmetic intensities.

from the matrix Z from DRAM and then write the results in to a matrix W_{XY}. The size of data matrix Z is $N \times d$, where $N = 13,475,840$ and $d = 129$ for our test data. As we store the data in double precision, the total size of the matrix (and hence volume of data read) is at least 13.907×10^9 bytes. In the inner loop, the processor must continually access M rows of the matrix Z. As the resultant volume of data (103,200 bytes) easily fits in cache, we need only real each Z_i once (data movement is well proxies by compulsory cache misses). The size of the matrix W_{XY} is $(N - M) \times M$, where $M = 100$. As each double-precision element is written once, we can bound write data movement as $(N - M) \times M \times 8$ = 10.78×10^9 bytes. A similar calculation can be performed to calculate the requisite number of floating-point operations. In the optimized code, although there are dot products for $< Z_j, Z_j >$ coupled with a reciprocal square root and one exponential per element of W_{XY}, the DGEMM used for calculating $X_{mat} \times Y_{submat}$ should dominate the flop count. The matrix X_{mat} is 100×129, the matrix Y_{submat} is on average 64×129, and there are roughly $13,475,840/64 = 210560$ Y_{submat} matrices. Thus, the number of floating-point operations in the loop is about $210560 \times 2 \times 64 \times 129 \times 100 = 347.68 \times 10^9$ (ignoring any BLAS 2 operations, the dot products, and exponential).

	Theoretical	Empirical
Bytes Read	13.907×10^9	17.123×10^9
Bytes Written	10.781×10^9	12.256×10^9
FP operations	$>347.68\times10^9$	385.59×10^9
Arithmetic Intensity (flop:byte)	>14.1	13.12

Table 3: Theoretical estimates (ignoring dual-socket nature of the machine) and Empirical measurements (using VTune and SDE) of data memory and floating-point operations for the Nyström loop.

Table 3 presents our theoretical estimates and empirical measurements (using VTune and SDE) of data memory and floating-point operations for the Nyström loop. As expected, our rough theoretical model slightly underestimated each quantity. Multiple sockets (each with their own caches) may be required to read unique bytes, but in reality will access overlapping data due to the realities of large cache lines and hardware stream prefetchers. In terms of floating-point operations it is clear DGEMM (the basis for our theoretical model) constitutes over 90% of the total flop count with the remainder likely arising from exponentials and dot products. Overall, with a run time of about 1.28 seconds, the optimized code attains about 300GFlop/s of performance and 23GB/s of DRAM bandwidth at an arithmetic intensity of just over 13 flops per byte. At such a high arithmetic intensity, Figure 22 suggests the full node DRAM bandwidth will not be the ultimate limiting factor. However, as we have not included any NUMA optimizations in the implementation, we expect the single socket's DRAM bandwidth (slightly less than 54GB/s) to be a substantial performance impediment. Additional data movement in the cache hierarchy coupled with performance challenges associated with transcendental operations like reciprocal-square-root and exponential likely impeded our ability to fully saturate even a single socket's bandwidth.

CHAPTER 6

Ego-motion Classification

In this chapter, we develop the graph classification method for a real-world application from body worn cameras [108]. Portable cameras record dynamic first-person video footage and these videos contain information on the motion of the individual on whom the camera is mounted, defined as ego. We address the task of discovering ego-motion from the video itself, without other external calibration information. We deal with this problem in two steps. The first step is presented in section 6.1. We investigate the use of similarity transformations between successive video frames to extract signals reflecting ego-motions and their frequencies. These signals are processed to make the feature vectors. Then, we use the graph-based unsupervised and semi-supervised learning algorithms to segment the video frames into different ego-motion categories. In section 6.2, we show the promising results on both the choreographed and real-world data. To use the semi-supervised method, we hand labeled around 10% of the frames as training data. Since each video may contain huge amount of frames, we make use of the parallelized codes developed in *Chapter 5* for efficient calculation. Our results show very accurate results on both choreographed test videos and ego-motion videos provided by the Los Angeles Police Department.

6.1 Motion Features

We characterize motion in a video sequence using a set of features. The features represent the relative movements of ego, the individual on whom the video camera is mounted. The features depend on the estimation of parametric models between successive frames and on the analysis of periodic signals of the motion through characteristic frequencies. We illustrate our method of constructing the motion features in Figure 23. In subsection 2.1, we discuss how to use the inverse compositional algorithm to estimate the similarity transformation between successive frames. This transformation is represented by four parameters t_x, t_y, a and b. In subsection 2.2, we construct four of the features to be used for the video segmentation – horizontal displacement (x), vertical

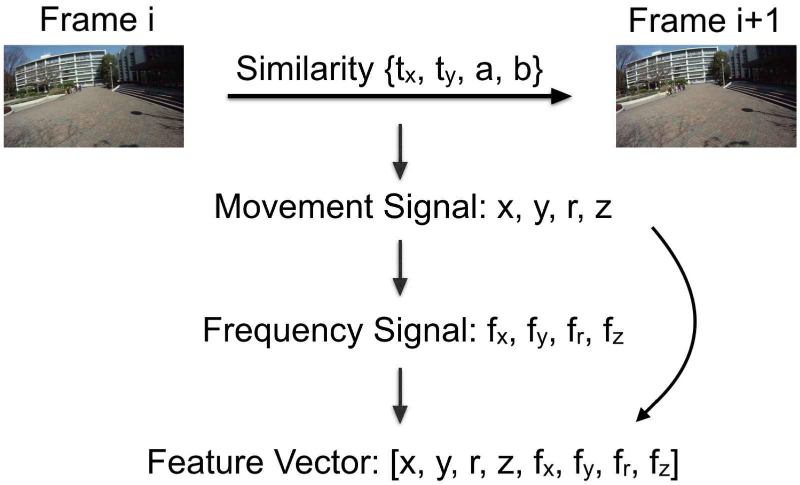

Frame i
Frame i+1
Similarity {t_x, t_y, a, b}
Movement Signal: x, y, r, z
Frequency Signal: f_x, f_y, f_r, f_z
Feature Vector: [x, y, r, z, f_x, f_y, f_r, f_z]

Figure 23: The process of constructing the motion features for each two successive frames.

displacement (y), angle of rotation (r) and zoom (z) using the similarity transformation. In addition, the characteristic frequencies of these four signals are computed using the method discussed in subsection 2.3. In subsection 2.4, we combine the four movement features and four frequency features to obtain the eight-dimensional feature vector for each transformation between two successive frames. It is this feature vector that will be used for the graph-based machine learning method.

6.1.1 Transformations between two Successive Frames

To compute the motion of the video sequence, we estimate the similarity transformations between consecutive frames using the inverse compositional algorithm [9, 10]. It is possible to use more general parametric motions, such as affinities or homographies. However, the calculation of these is more prone to errors when some camera shake is present. In any case, we find that the four parameters of the similarity are sufficient to characterize motion.

The inverse compositional algorithm is an improvement of the Lucas-Kanade method [10, 97] for image registration. Its implementation in [129] includes the use of robust error functions, which allows estimating the correct transformation even in the presence of occlusions or multiple motions. Let $\mathbf{I}_1(\mathbf{x})$ and $\mathbf{I}_2(\mathbf{x})$ be two images, with $\mathbf{x} = (x, y)$. Let $\mathbf{p}$ be the global displacement vector between the two images and $\Delta\mathbf{p}$ be the incremental displacement vector at each iteration. Let $\mathbf{x}'(\mathbf{x}; \mathbf{p}, \Delta\mathbf{p})$ be the correspondence map from the left to the right image, or equivalently two frames in a video sequence, parameterized by $\mathbf{p}$ and the incremental refinement $\Delta\mathbf{p}$. The energy model is given by

$$E(\Delta \mathbf{p}) = \sum_{\mathbf{x}} \rho \left(\left| \mathbf{I}_2(\mathbf{x}'(\mathbf{x};\mathbf{p})) - \mathbf{I}_1(\mathbf{x}'(\mathbf{x};\Delta\mathbf{p})) \right|_2^2 ; \lambda \right), \tag{38}$$

where $\rho\left(\cdot\right)$ is a function that gives less weight to large values of the argument, where the difference in image intensities is big (e.g., $\rho(s^2, \lambda) = 0.5 s^2/(s^2 + \lambda^2)$).

Minimizing the energy with respect to $\Delta\mathbf{p}$ yields:

$$\Delta\mathbf{p} = H_{\delta}^{-1} \sum_{\mathbf{x}} \rho' \cdot \left(\nabla \mathbf{I}_1(\mathbf{x}) \mathbf{J}(\mathbf{x})\right)^T \left(\mathbf{I}_2(\mathbf{x}'(\mathbf{x};\mathbf{p})) - \mathbf{I}_1(\mathbf{x})\right), \tag{39}$$

with

$$\begin{aligned} H_{\delta} =& \sum_{\mathbf{x}} \rho' \cdot \left(\nabla \mathbf{I}_1(\mathbf{x}) \mathbf{J}(\mathbf{x})\right)^T \nabla \mathbf{I}_1(\mathbf{x}) \mathbf{J}(\mathbf{x}) \\ =& \begin{pmatrix} \sum\limits_{\mathbf{x}} \rho' \cdot \left(\mathbf{I}_{1,x}(\mathbf{x}) \mathbf{J}(\mathbf{x})\right)^T \mathbf{I}_{1,x}(\mathbf{x}) \mathbf{J}(\mathbf{x}) & \sum\limits_{\mathbf{x}} \rho' \cdot \left(\mathbf{I}_{1,x}(\mathbf{x}) \mathbf{J}(\mathbf{x})\right)^T \mathbf{I}_{1,y}(\mathbf{x}) \mathbf{J}(\mathbf{x}) \\ \sum\limits_{\mathbf{x}} \rho' \cdot \left(\mathbf{I}_{1,x}(\mathbf{x}) \mathbf{J}(\mathbf{x})\right)^T \mathbf{I}_{1,y}(\mathbf{x}) \mathbf{J}(\mathbf{x}) & \sum\limits_{\mathbf{x}} \rho' \cdot \left(\mathbf{I}_{1,y}(\mathbf{x}) \mathbf{J}(\mathbf{x})\right)^T \mathbf{I}_{1,y}(\mathbf{x}) \mathbf{J}(\mathbf{x}) \end{pmatrix}, \end{aligned} \tag{40}$$

and $\rho' := \rho'\left(\left| \mathbf{I}_2(\mathbf{x}'(\mathbf{x};\mathbf{p})) - \mathbf{I}_1(\mathbf{x}) \right|_2^2 ; \lambda \right)$. $\mathbf{J}(\mathbf{x};\mathbf{p}) = \frac{\partial \mathbf{x}'(\mathbf{x};\mathbf{p})}{\partial \mathbf{p}}$ is the Jacobian of the transformation. Table 4 lists the similarity transformation and its Jacobian using the parametrization proposed in [139].

Transform	**Parameters –** $\mathbf{p}$	**Matrix –** $\mathbf{H}(\mathbf{p})$	**Jacobian –** $\mathbf{J}(\mathbf{x};\mathbf{p})$
Similarity	(t_x, t_y, a, b)	$\begin{pmatrix} 1+a & -b & t_x \\ b & 1+a & t_y \\ 0 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 & x & -y \\ 0 & 1 & y & x \end{pmatrix}$

Table 4: Similarity transformation and its Jacobian

The minimum of this energy provides the parameters of the transformation. To reach a highly accurate solution, the algorithm uses an iterative process. It also includes a coarse-to-fine strategy for estimating large displacements. See [129] for further details.

6.1.2 Movement Signal

Simple motions, such as horizontal (x) and vertical (y) movements, zoom (z) and rotation (r) information can be computed given the similarity. The procedure for calculating the displacement of the central pixel is shown in Algorithm 1.

```
Algorithm 1: Calculate the displacement of the central pixel
  Input  : The similarity $\mathbf{H}$, size of the frame $n_x$ and $n_y$
  Output: $x$,$y$
  $\mathbf{p_m} \leftarrow (n_x/2, n_y/2, 1)^T$ {the center of the frame}
  $(p_1, p_2, p_3)^T \leftarrow \mathbf{H} \cdot \mathbf{p_m}$ {project the center point using the similarity }
  $(p_1, p_2, p_3)^T \leftarrow (p_1, p_2, p_3)^T / p_3$ {normalize by the third component}
  $x \leftarrow p_1 - n_x/2$ {the horizontal movement}
  $y \leftarrow p_2 - n_y/2$ {the vertical movement}
  return $x$, $y$
```

Since the similarity includes the composition of a zoom and rotation matrices, it is easy to obtain these coefficients from the parametrization of Table 4. In this case, the rotation and zoom factor are calculated as

$$r = \arctan\left(\frac{b}{1+a}\right), \ z = \sqrt{(1+a)^2 + b^2}, \tag{41}$$

respectively.

The signals from raw video footage may have abnormally large values. We filter out these values in preprocessing. We replace the signal value by μ, where μ is the mean of the signal sequence and σ is the standard derivation, if the signal value is outside the $(\mu - 3\sigma, \mu + 3\sigma)$ region. The filtered signals can still be very noisy. We use convolutions with a Gaussian function to smooth these signals, which is the basic idea in video stabilization [130].

We use the QUAD video data set [2] to examine ego-motion signals. We discuss the details of this data set in section 6.2.

The motion signals we calculate using Algorithm 1 and Equation (41) are shown in Figure 24. The left column gives the raw data x, y, z and r and the right column the corresponding filtered and smoothed data.

The periodic pattern correlates with the periodic actions in the QUAD video. The large oscillation of x corresponds to ego turning left and right repeatedly. The large oscillation of y corresponds to ego repeatedly looking up and looking down. The four peaks in z correspond to ego walking and running, since the frames zoom fast when the person is walking or running. The large oscillations of rotation r also correlate with the movements of turning left, turning right, looking up and

[2] The data set can be found at: http://www.cs.cmu.edu/~kkitani/datasets/

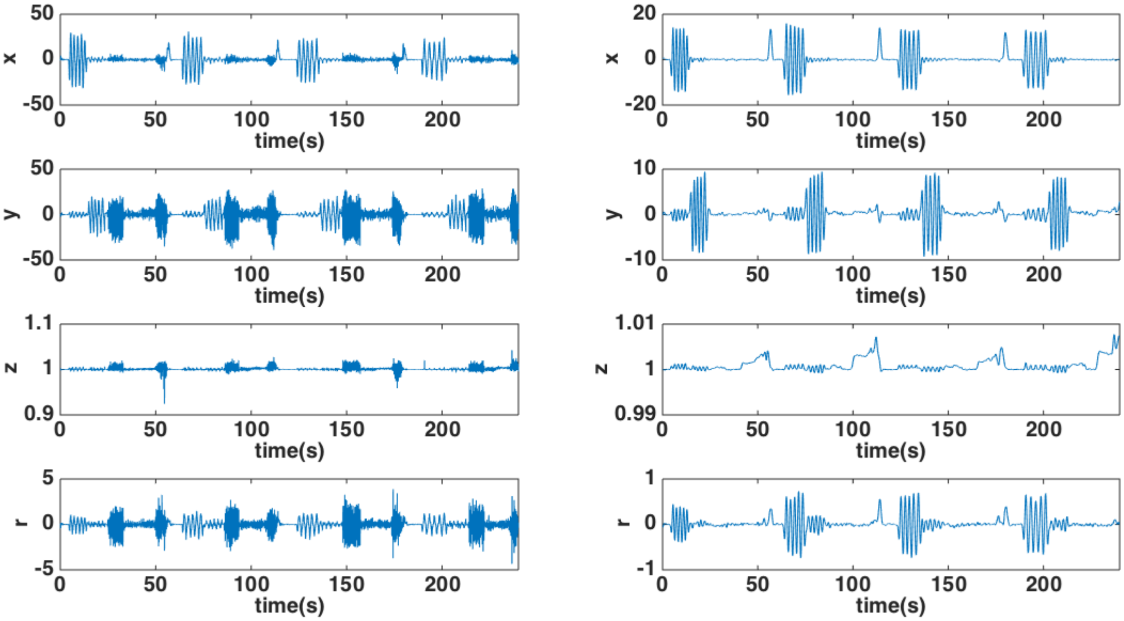


Figure 24: The x, y, r and z signals. On the left, the original signals and, on the right, the corresponding filtered and smoothed data.

looking down.

6.1.3 Frequency Signal

Some ego-motions are periodic, such as jumping, walking and running. Periodic motions have different characteristic frequencies. This observation leads us to investigate the frequencies of x, y, z and r using Fourier analysis. We use the short-time Fourier transform (STFT) to determine the sinusoidal frequency and phase content of local sections of a signal as it changes over time. In practice, the procedure for computing STFTs is to use a sliding window of fixed length and compute the Fourier transform as the window slides over the whole signal. We use the Hann window here:

$$w(n) = 0.5\left(1 - cos(\frac{2\pi n}{N-1})\right). \tag{42}$$

As shown in Figure 25, the Hann window is zero at the boundaries which reduces the artifacts at the boundary. The STFT is defined by:

$$STFTx[n](m,\omega) = X(m,\omega) = \sum_{n=0}^{N} x[n]w[n-m]e^{-j\omega n}, \tag{43}$$

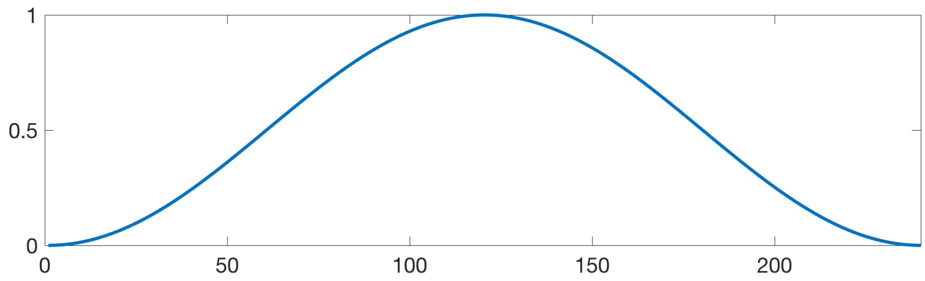


Figure 25: The Hann window

where the length of the window is N and m indicates the window sampling rate.

The magnitude squared of the STFT yields the spectrogram of the function:

$$spectrogram\{x[n]\}(m,\omega) = |X(m,\omega)|^2. \tag{44}$$

We use a five-second window in our experiments. We show the spectrogram of six different motions of the y signal in Figure 26. The frequency is very small when ego repeatedly turns left and right. Looking up and down causes a frequency at 0.6 Hz. The spectrogram of small steps and walking are very similar. The largest frequency is at 7.8 Hz. When ego walks at 0.5 seconds per step, the frequency is 2 Hz. However, because the GoPro camera is head-mounted, the camera also has an oscillation when ego is walking. This camera oscillation causes this observed high frequencies. For jumping and running, the spectrogram gives accurate frequencies at 2Hz and 3.4 Hz, respectively.

We select the characteristic frequency of the window, which is defined as:

$$f_w = \begin{cases} f_{max}, & \text{if } f_{max} > 3\delta \\ 0, & \text{otherwise} \end{cases}, \tag{45}$$

where f_{max} is the frequency corresponding to the largest value in the spectrogram and δ is the standard deviation of the spectrogram. The condition of being larger than 3δ guarantees that the frequency picked is unlikely to be caused by noise.

In practice, we choose N to be 300 frames (5 seconds) and let the window moves 60 frames (1 second) each time. In this case, at each frame, there are 5 f_ws. We choose the median of these f_ws to be the final frequency at the frame.

We apply this procedure with the four movement signals x, y, r, and z and get four frequency signals f_x, f_y, f_r and f_z. In other words, in addition to four movement signals, each frame transition is also associated with four characteristic frequencies. We compute these frequencies for

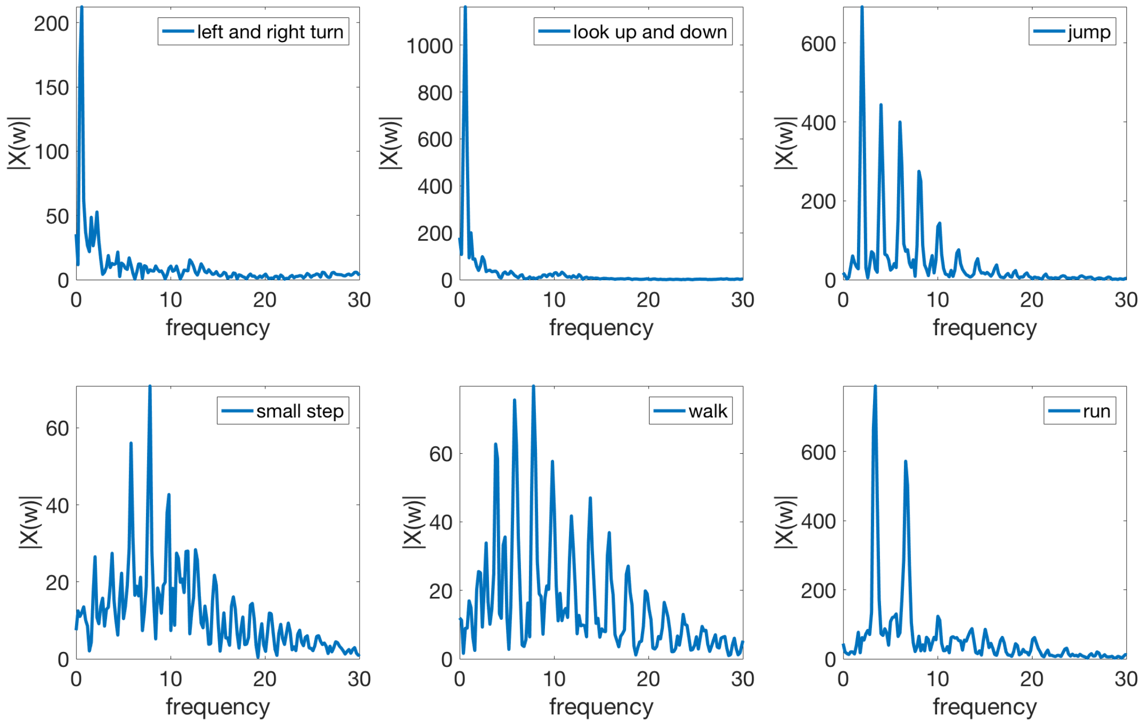


Figure 26: Spectrogram of 6 kinds of motions in the QUAD video

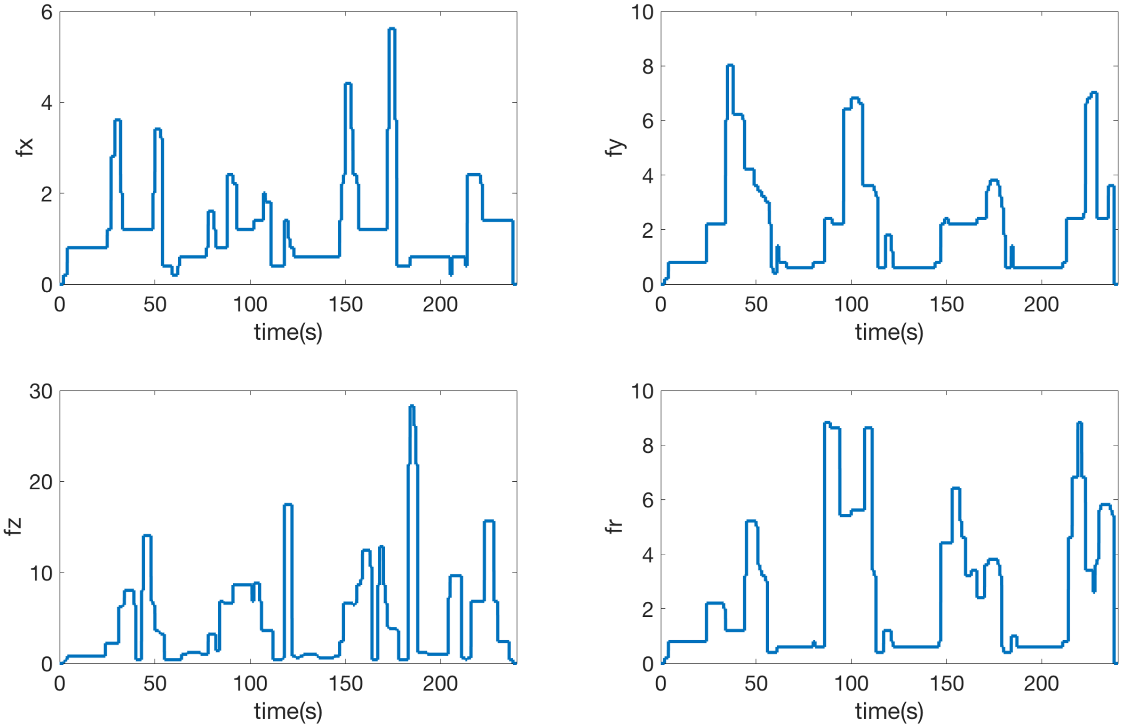


Figure 27: The four characteristic frequencies fx, fy, fz, fr of the QUAD video.

the QUAD video and show their values in Figure 27. We can observe four time intervals in the frequencies corresponding to specific action periods in the video.

6.1.4 Equalization of Variance

We always force the variance of each signal to be 1 by forcing x to be $\bar{x} + \frac{x-\bar{x}}{\sigma(x)}$, where $\bar{x}$ is the mean and $\sigma(x)$ is the standard variation. In this way, each signal gives equal contribution to the combined feature vector. Different graph weights can be considered for different signals based on the importance of the signals.

After equalizing the variance of the 8 signals, we combine them into a final motion feature f_{motion}. It is an $N \times 8$ matrix, where N is the number of frames in the video. Each row represents the eight-dimensional feature vector of one frame. In this way, we code the video to the feature matrix f_{motion},

$$f_{motion} = [x, y, r, z, f_x, f_y, f_r, f_z]. \tag{46}$$

6.2 Experimental Results

To evaluate the performance of our method we need both choreographed video sequences to run controlled experiments and real-world videos to observe performance of our method in naturalistic settings. It is easy to define the ground truth for the choreographed videos since the ego motion is discrete, and well-defined. For example, looking left and right never coincides with running. However, real-world body-worn video usually contains a combination of different motions with noise and it is therefore harder to define a ground truth.

6.2.1 Choreographed Video

The first video we use is QUAD [88]. We show one frame of the QUAD video in Figure 28. This video is four minutes and 10 seconds in length and has 60 frames per second. It contains nine ego-motions (stand still, turn left, turn right, look up, look down, jump, step in place, walk and run) taken from a head-mounted GoPro camera. There are nine actions in order repeated four times. The ground truth is shown in the first row of Figure 29. The horizontal axis represents time and colors represent different ego-motion categories. The order of the movements are standing still, turning left and turning right repeatedly, looking up and looking down repeatedly, jumping,

Figure 28: One frame of the QUAD video.

stepping, walking, running, turning left and then start the same series of motions again for another three times.

We compute the feature vector for each two successive frames as described in section 6.1. Then we use K-means, the unsupervised graph MBO algorithm and the semi-supervised graph MBO algorithm for the ego-motion classification. We use 10% known labels (evenly sampled) in the semi-supervised graph MBO algorithm. The classification results of these three algorithms are shown in the 2nd, 3rd and 4th rows of Figure 29. For the K-means and the unsupervised MBO algorithm, we ran the experiments several times and pick the best results here. Depending on the initialization, these two algorithms can converge to different local minima, which is common for most non-convex variational methods. The K-means algorithm gives relatively good results, except that it does not recognize the category of looking down and misclassifies some parts of running, jumping, small steps and walking. The unsupervised graph MBO algorithm gives results similar to K-means. The semi-supervised graph MBO algorithm with 10% known labels gives very accurate results. The accuracy summary of these three algorithms is shown in Table 5.

6.2.2 Real-world Body-worn Video

We also investigated real-world body-worn videos. We use a data set from the Los Angeles Police Department. The videos are from police, wearing chest-mounted cameras while patrolling areas of Los Angeles on foot. The videos record a wide array of police activities from basic patrol through foot chases and arrest. Our ego-motion classification results may be used in modeling the

Table 5: Accuracy Summary of the QUAD data set

Accuracy	Overall	Average	1.Stand still	2.Turn left	3.Turn right	4.Look up
K-means	**64.84%**	**61.79%**	95.82%	72.26%	77.28%	73.24%
Unsupervised MBO	**66.62%**	**67.59%**	79.99%	76.82%	83.37%	69.41%
Semi-supervised MBO	**89.14%**	**88.74%**	87.90%	89.43%	92.80%	80.36%
Accuracy	5.Look down	6.Jump	7.Step	8.Walk	9.Run	
K-means	0	83.29%	49.29%	36.66%	68.25%	
Unsupervised MBO	77.82%	39.38%	43.54%	83.27%	54.68%	
Semi-supervised MBO	84.59%	92.71%	93.98%	84.52%	92.38%	

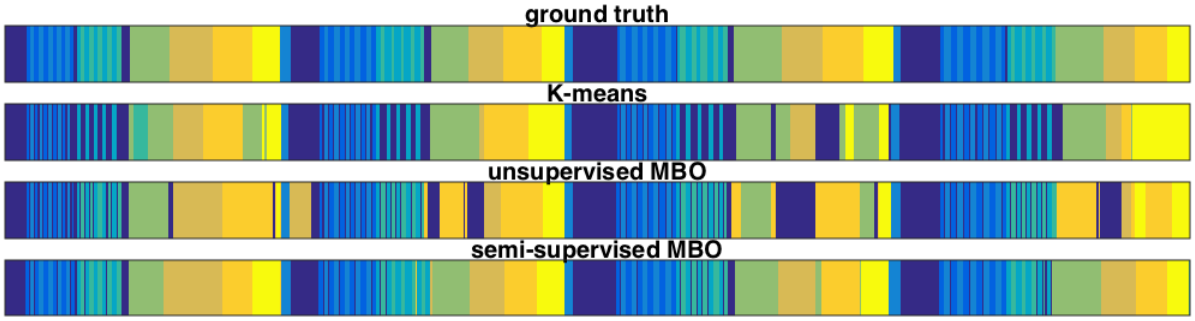


Figure 29: Ego-motion classification results of the QUAD video. The 9 colors represent 9 different ego-motion classes: standing still (dark blue), turning left (moderate blue), turning right (light blue), looking up (dark green) and looking down (light green), jumping (bud green), stepping (aztec gold), walking (orange), running (yellow).

routine activities of police and their interactions with the public.

Police BWV is not collected under controlled circumstances. Ego-motions may evolve rapidly without clear or discrete transitions. Much body worn video is collected at night impacting light and color saturation. The videos also have distortion due to the use of a fish-eye lens. Since there has been very little formal analysis of police BWV, there is a lack of appreciation for the diversity of police behavior likely to be encountered (i.e., very limited semantic dictionaries). The ground-truth is labeled by us without input from the police.

We show here the video segmentation result of one clip of police video. The video is 8 minutes and 16 seconds in length, with 14991 frames in total. In the video, police arrive at an apartment building, talk with some people in front of the building, go upstairs, wait outside a room, enter and search the room, leave the room, walk downstairs, and talk to several people outside the building. We define four ego-motion categories in this video – standing (or very slow motions not easy to define), walking, going upstairs, and going downstairs. The ground truth classification of this video is shown in the first row of Figure 30. The dark blue segments represent the category of standing or slow movements when the officer talks with others in front of the building. It also contains actions when the officer enters the room. The video of this period is very shaky and not easily defined as one motion category. The light blue segment corresponds to the walking category. The green segment corresponds to the police going upstairs and the yellow part is going downstairs.

We explore the same algorithms for the police body-worn video. We are not using the unsupervised graph MBO algorithm because the result is not consistent. The results are shown in Figure 30. K-means captures the difference between going upstairs and downstairs. However, K-means frequently misclassifies walking and going downstairs. Some standing frames are classified as other motion categories. This later result is reasonable since standing in this video combines some other movements. Then we use the semi-supervised graph MBO algorithm with 10% known labels on this piece of video. The segmentation results are shown in the third row of Figure 30. It can be seen that the result is much better than K-means, and the four categories are all captured almost correctly. The accuracy summary is shown in Table 6. The overall accuracy of the semi-supervised graph MBO algorithm with 10% known labels is 90.17%.

6.3 Future Work

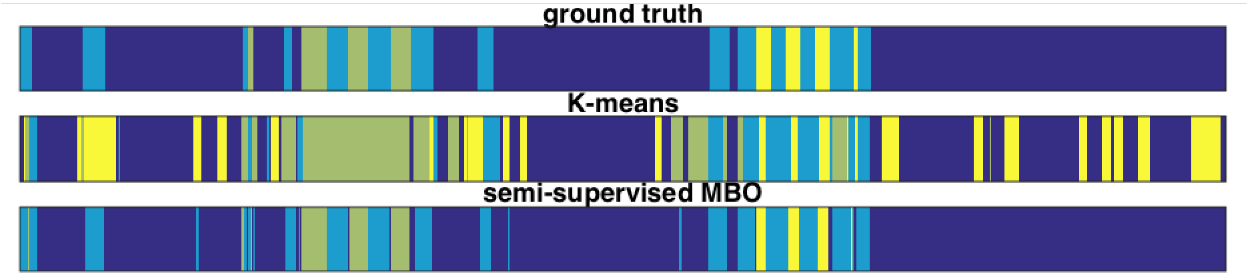


Figure 30: Ego-motion classification results of the police video. The 4 colors represent 4 different ego-motion classes: standing or very slow motions and motions not easy to define (dark blue), walking (light blue), going upstairs (green) and going downstairs (yellow).

Table 6: Accuracy Summary of the police body-worn video data set

Accuracy	Overall	Average	1.Stand	2.Walk	3.Upstairs	4.Downstairs
K-means	**63.62%**	**63.77%**	68.91%	37.78%	91.84%	56.53%
Semi-supervised MBO	**90.17%**	**74.09%**	96.10%	82.12%	83.45%	34.71%

The potential for future advances in this area is significant particularly in relation to police body-worn video. At full deployment of body-worn video in 2018, the Los Angeles Police Department is projected to collect 3.2 million individual videos totaling more than 200K hours of total video feed per year. This represents both a vast resource and a significant analytical challenge. The amount of data suggests that the full array of ego-motions practiced by police might eventually be discovered and subject to classification, moving us towards a realistic picture of the diversity of police activities. There will clearly be no lack of training data with which to tackle this problem.

The work presented in this chapter is already being extended. The student at the summer REU program at UCLA have labeled more video frames for nine motion classes: standing, walking, in moving or stationary car, obscured camera, at car window, enter driver, enter passenger, exit driver and exit passenger. They have also added more features for each frame, such as the color signatures, texture signatures, etc. The classification accuracies for these nine classes are shown in Figure 31. This figure also show that the sizes of the classes affect the classification accuracy.

The same surfeit of video data is proving to be true in other domains outside of policing. Recognition of the diversity of ego-motion in policing activity may also lead to novel extensions of the methods into dyadic- and n-person motion models. In the dyadic-motion case there is much to be learned. It is well known that relative motion of individuals with respect to one another encodes fundamental social information [12]. For example, an individual running away from ego may encode avoidance or fear, while an individual running directly towards ego may encode attraction and threat. More complex social interactions may be captured in n-person motion models.

The challenges to achieving such outcomes with real-world video are also significant. In the police body-worn video case, semi-supervised classification clearly outperforms the unsupervised approach. Yet even a small fraction of fidelity points (10% in the current method) is probably infeasible given the volumes of video arriving each day. Semi-supervised methods will therefore need to rely on as few fidelity points as possible. However another approach is video labeling where activities segmented in one video might be used as labels for semi-supervised segmentation in another video. This was demonstrated in [15, 16] for image labelling. It will also be necessary to consider how generalizable methods are across real-world video examples. Ideally, a handful of videos might be exhaustively labeled for ground-truth and these would then work across the growing set of videos. This is an empirical questions that we can start addressing now with the recognition that new methods may be needed to account for the variability of real-world video.

Finally, we also point out that body-worn video is but one sensor platform in what is increasingly a multi-sensor world. It is worth investigating whether there is an advantage to doing more with single sensors, or whether it is better to integrate the signals from many independent sensors. For example, we can imagine doing both ego-motion and scene topic classification from the same video sequence, or as an alternative use accelerometers to capture ego-motion and matching these data to scene classification from video. Importantly, the issues are not strictly technological. Police body-worn video is treated as evidence and therefore is subject to all of the evidence handling rules required by law. Each sensor implies a different packet of physical of evidence that must be maintained and handled appropriately. Future work will need to examine these sorts of tradeoffs in detail.

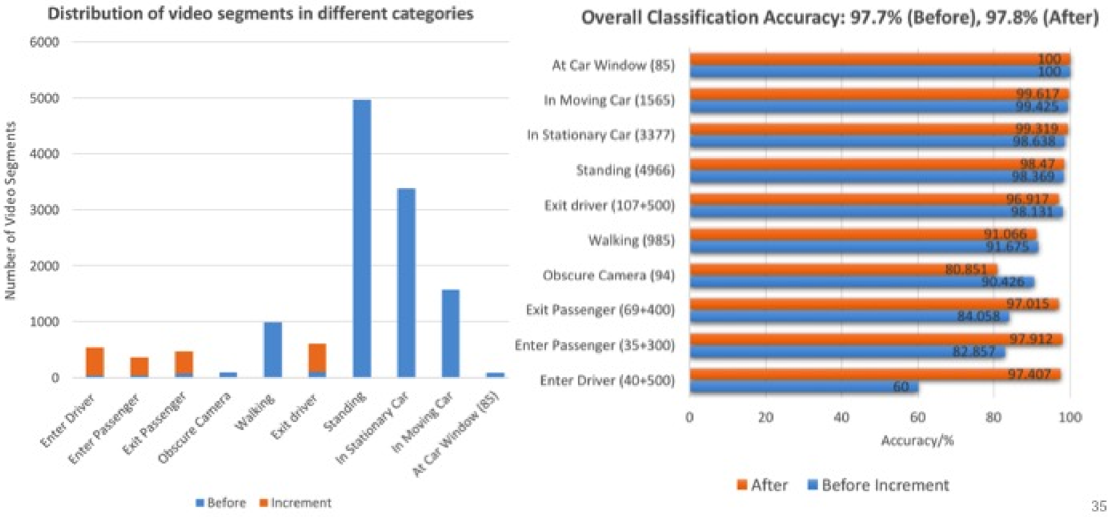


Figure 31: Left: Distribution of video segments across different ego-motion categories before and after increment by replicating the video segments. Right: Increment of video segments significantly increasesthe classification accuracy of the last three categories. [4]

Summary

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 times. In fact, the entire process only takes one minute using 64 threads on a supercomputer for 329 frames (13,475,840 pixels) of the plume video data on LBNL supercomputers!

We develop a parallel implementation of two novel classification algorithms using OpenMP. We show OpenMP parallel and SIMD regions in combination with optimized library routines achieving almost ideal scaling and manyfold speedup over serial implementations. Although, we attain roughly 50% of the Roofline bound (no NUMA), we expect future optimizations for the transcendentals, the cache hierarchy, and NUMA to substantially improve performance. We also expect more performance optimization results on KNL "white boxes" (pre-release hardware) and the future Cori Phase II.

We investigate the task of discovering ego-motion categories from first-person videos. We deal with this problem in two steps. The first step is comparing two successive frames using the inverse compositional algorithm to extract signals containing motion and motion frequency information. Then we use unsupervised and semi-supervised clustering algorithms for classification. The semi-supervised graph based methods are particularly accurate using only 10% training data. We show promising results on both choreographed and real-world video data. This work is now being extended to many more classes with new methods for uncertainty qualification.

References

[1] Intel software development emulator: https://software.intel.com/en-us/articles/intel-software-development-emulator.

[2] Intel vtune official website: https://software.intel.com/en-us/intel-vtune-amplifier-xe.

[3] Roofline toolkit: https://bitbucket.org/berkeleylab/cs-roofline-toolkit.

[4] O. Akar, H. Chen, A. Dhillon, A. Song, and T. Zhou. Body worn video, 2017. Technical report from 2017 summer REU on classification of BWV from LAPD; A. L. Bertozzi, M. Haberland, H. Li, P. J. Brantingham, and M. Roper faculty mentors.

[5] Samuel M Allen and John W Cahn. A microscopic theory for antiphase boundary motion and its application to antiphase domain coarsening. *Acta Metallurgica*, 27(6):1085–1095, 1979.

[6] C. Anderson. A Rayleigh-Chebyshev procedure for finding the smallest eigenvalues and associated eigenvectors of large sparse Hermitian matrices. *Journal of Computational Physics*, 229:7477–7487, 2010.

[7] P. Arias, V. Caselles, and G. Sapiro. A variational framework for non-local image inpainting. *Energy Minimization Methods in Computer Vision and Pattern Recognition*, pages 345–358, 2009.

[8] Egil Bae and Ekaterina Merkurjev. Convex variational methods for multiclass data segmentation on graphs. submitted, 2016.

[9] Simon Baker and Iain Matthews. Equivalence and efficiency of image alignment algorithms. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–I. IEEE, 2001.

[10] Simon Baker and Iain Matthews. Lucas-Kanade 20 years on: A unifying framework. *International journal of computer vision*, 56(3):221–255, 2004.

[11] G. Barles and C. Georgelin. A simple proof of convergence for an approximation scheme for computing motions by mean curvature. *SIAM Journal on Numerical Analysis*, 32(2):484–500, 1995.

[12] H Clark Barrett, Peter M Todd, Geoffrey F Miller, and Philip W Blythe. Accurate judgments of intention from motion cues alone: A cross-cultural study. *Evolution and Human Behavior*, 26(4):313–331, 2005.

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

[14] M. Bertalmio, L. Vese, G. Sapiro, and S. Osher. Simultaneous structure and texture image inpainting. *IEEE Transactions on Image Processing*, 12(8):882–889, 2003.

[15] Andrea L Bertozzi and Arjuna Flenner. Diffuse interface models on graphs for classification of high dimensional data. *Multiscale Modeling & Simulation*, 10(3):1090–1118, 2012.

[16] Andrea L Bertozzi and Arjuna Flenner. Diffuse interface models on graphs for classification of high dimensional data. *SIAM Review*, 58(2):293–328, 2016.

[17] José M Bioucas-Dias and José M P Nascimento. Hyperspectral subspace identification. *IEEE Transactions on Geoscience and Remote Sensing*, 46(8):2435–2445, 2008.

[18] Dankmar Böhning. Multinomial logistic regression algorithm. *Annals of the Institute of Statistical Mathematics*, 44(1):197–200, 1992.

[19] Bernhard E Boser, Isabelle M Guyon, and Vladimir N Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, pages 144–152. ACM, 1992.

[20] Patrick Bouthemy, Marc Gelgon, and Fabrice Ganansia. A unified approach to shot change detection and camera motion characterization. *IEEE transactions on circuits and systems for video technology*, 9(7):1030–1044, 1999.

[21] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(11):1222 –1239, 2001.

[22] Yuri Boykov, Olga Veksler, and Ramin Zabih. Fast approximate energy minimization via graph cuts. *IEEE Transactions Pattern Analysis and Machine Intelligence*, 23(11):1222–1239, 2001.

[23] X. Bresson, T. Laurent, D. Uminsky, and J. von Brecht. Convergence and energy landscape for Cheeger cut clustering. *Advances in Neural Information Processing Systems*, 25:1394–1402, 2012.

[24] Xavier Bresson, Thomas Laurent, David Uminsky, and James H von Brecht. Convergence and energy landscape for Cheeger cut clustering. In *Advances in Neural Information Processing Systems*, pages 1385–1393, 2012.

[25] Xavier Bresson, Thomas Laurent, David Uminsky, and James H von Brecht. Multiclass total variation clustering. In *Advances in Neural Information Processing Systems*, pages 1421–1429, 2013.

[26] Xavier Bresson, Thomas Laurent, David Uminsky, and James H von Brecht. An adaptive total variation algorithm for computing the balanced cut of a graph. *arXiv preprint arXiv:1302.2717*, 2013.

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

[28] Joshua B Broadwater, Diane Limsui, and Alison K Carr. A primer for chemical plume detection using lwir sensors. *Technical Paper, National Security Technology Department, Las Vegas, NV*, 2011.

[29] Lorenzo Bruzzone, Mingmin Chi, and Mattia Marconcini. A novel transductive svm for semisupervised classification of remote-sensing images. *IEEE Transactions on Geoscience and Remote Sensing*, 44(11):3363–3373, 2006.

[30] A. Buades, B. Coll, and J.-M. Morel. A review of image denoising algorithms, with a new one. *Multiscale Modeling & Simulation*, 4(2):490–530, 2005.

[31] Antoni Buades, Bartomeu 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)*, volume 2, pages 60–65. IEEE, 2005.

[32] T. Bühler and M. Hein. Spectral clustering based on the graph p-Laplacian. *International Conference on Machine Learning*, pages 81–88, 2009.

[33] J.-F. Cai, R.H. Chan, and Z. Shen. A framelet-based image inpainting algorithm. *Applied and Computational Harmonic Analysis*, 24(2):131–149, 2008.

[34] Gustavo Camps-Valls and Lorenzo Bruzzone. Kernel-based methods for hyperspectral image classification. *IEEE Transactions on Geoscience and Remote Sensing*, 43(6):1351–1362, 2005.

[35] Gustavo Camps-Valls, Luis Gomez-Chova, Jordi Muñoz-Marí, Joan Vila-Francés, and Javier Calpe-Maravilla. Composite kernels for hyperspectral image classification. *IEEE Geoscience and Remote Sensing Letters*, 3(1):93–97, 2006.

[36] Tony F Chan and Luminita A Vese. Active contours without edges. *IEEE Transactions on Image Processing*, 10(2):266–277, 2001.

[37] O. Chapelle, B. Schölkopf, and A. Zien. *Semi-supervised learning*, volume 2. MIT Press, 2006.

[38] Olivier Chapelle, Bernhard Schölkopf, and Alexander Zien, editors. *Semi-Supervised Learning*. MIT Press, Cambridge, MA, 2006.

[39] Olivier Chapelle, Bernhard Scholkopf, and Alexander Zien. Semi-supervised learning (chapelle, o. et al., eds.; 2006)[book reviews]. *IEEE Transactions on Neural Networks*, 20(3):542–542, 2009.

[40] Wen-Yen Chen, Yangqiu Song, Hongjie Bai, Chih-Jen Lin, and Edward Y Chang. Parallel spectral clustering in distributed systems. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(3):568–586, 2011.

[41] Yi Chen, Nasser M Nasrabadi, and Trac D Tran. Sparse representation for target detection in hyperspectral imagery. *IEEE Journal of Selected Topics in Signal Processing*, 5(3):629–640, 2011.

[42] Yi Chen, Nasser M Nasrabadi, and Trac D Tran. Hyperspectral image classification using dictionary-based sparse representation. *IEEE Transactions on Geoscience and Remote Sensing*, 49(10):3973–3985, 2011.

[43] Fan RK Chung. *Spectral graph theory*, volume 92. American Mathematical Soc., 1997.

[44] 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(7):1384–1399, 2011.

[45] C. Couprie, L. Grady, H. Talbot, and L. Najman. Combinatorial continuous maximum flow. *SIAM Journal on Imaging Sciences*, 4(3):905–930, 2011.

[46] Camille Couprie, Leo Grady, Laurent Najman, and Hugues Talbot. Power watershed: A unifying graph-based optimization framework. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(7):1384–1399, 2011.

[47] James W Demmel. *Applied numerical linear algebra*. SIAM, 1997.

[48] X. Desquesnes, A. Elmoataz, and O. Lezoray. Pdes level sets on weighted graphs. *IEEE International Conference on Image Processing*, pages 3377–3380, 2011.

[49] J.A. Dobrosotskaya and A.L. Bertozzi. A wavelet-Laplace variational technique for image deconvolution and inpainting. *IEEE Transactions on Image Processing*, 17(5):657–663, 2008.

[50] Doug Doerfler. Understanding application data movement characteristics using intel vtune amplifier and software development emulator tools. *Intel Xeon Phi User Group (IXPUG)*, 2015.

[51] P. Drineas and M.W. Mahoney. On the Nyström method for approximating a Gram matrix for improved kernel-based learning. *The Journal of Machine Learning Research*, 6:2153–2175, 2005.

[52] Richard O Duda, Peter E Hart, and David G Stork. *Pattern classification*. John Wiley & Sons, 2012.

[53] 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(7):1047–1060, 2008.

[54] S. Esedoḡlu, S. Ruuth, and R. Tsai. Threshold dynamics for high order geometric motions. *Interfaces and Free Boundaries*, 10(3):263–282, 2008.

[55] S. Esedoḡlu, S. Ruuth, and R. Tsai. Diffusion generated motion using signed distance functions. *Journal of Computational Physics*, 229(4):1017–1042, 2010.

[56] Selim Esedoglu, Steven J Ruuth, and Richard Tsai. Threshold dynamics for high order geometric motions. *Interfaces and Free Boundaries*, 10(3):263–282, 2008.

[57] Selim Esedoḡlu and Yen-Hsi Richard Tsai. Threshold dynamics for the piecewise constant Mumford-Shah functional. *Journal of Computational Physics*, 211(1):367–384, 2006.

[58] L.C. Evans. Convergence of an algorithm for mean curvature motion. *Indiana University Mathematics Journal*, 42(2):533–557, 1993.

[59] G. Facciolo, P. Arias, V. Caselles, and G. Sapiro. Exemplar-based interpolation of sparsely sampled images. *Energy Minimization Methods in Computer Vision and Pattern Recognition*, pages 331–344, 2009.

[60] Michael D Farrell Jr and Russell M Mersereau. On the impact of PCA dimension reduction for hyperspectral detection of difficult targets. *IEEE Geoscience and Remote Sensing Letters*, 2(2):192–195, 2005.

[61] Xiaobing Feng and Andreas Prohl. Numerical analysis of the Allen-Cahn equation and approximation for mean curvature flows. *Numerische Mathematik*, 94(1):33–65, 2003.

[62] C. Fowlkes, S. Belongie, F. Chung, and J. Malik. Spectral grouping using the Nyström method. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(2):214–225, 2004.

[63] C. Fowlkes, S. Belongie, and J. Malik. Efficient spatiotemporal grouping using the Nyström method. *Computer Society Conference on Computer Vision and Pattern Recognition*, 1:I–231, 2001.

[64] C. Frohn-Schauf, S. Henn, and K. Witsch. Nonlinear multigrid methods for total variation image denoising. *Computing and Visualization in Science*, 7(3-4):199–206, 2004.

[65] Shenghua Gao, Ivor Wai-Hung Tsang, and Liang-Tien Chia. Kernel sparse representation for image classification and face recognition. In *European Conference on Computer Vision*, pages 1–14. Springer, 2010.

[66] C. Garcia-Cardona, A. Flenner, and A.G. Percus. Multiclass diffuse interface models for semi-supervised learning on graphs. *International Conference on Pattern Recognition Applications and Methods*, 2013.

[67] Cristina Garcia-Cardona, Ekaterina Merkurjev, Andrea L Bertozzi, Arjuna Flenner, and Allon G Percus. Multiclass data segmentation using diffuse interface methods on graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(8):1600–1613, 2014.

[68] 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*, pages 87430J–87430J. International Society for Optics and Photonics, 2013.

[69] G. Gilboa and S. Osher. Nonlocal operators with applications to image processing. *Multiscale Modeling & Simulation*, 7(3):1005–1028, 2008.

[70] Guy Gilboa and Stanley Osher. Nonlocal linear image regularization and supervised segmentation. *Multiscale Modeling & Simulation*, 6(2):595–630, 2007.

[71] L. Grady. Multilabel random walker image segmentation using prior models. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 763–770, 2005.

[72] L. Grady. Random walks for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(11):1768–1783, 2006.

[73] L. Grady and J.R. Polimeni. *Discrete calculus: applied analysis on graphs for computational science*. 2010.

[74] L. Grady, T. Schiwietz, S. Aharon, and R. Westermann. Random walks for interactive alpha-matting. *Visualization, Imaging and Image Processing*, pages 423–429, 2005.

[75] J Anthony Gualtieri and Robert F Cromp. Support vector machines for hyperspectral remote sensing classification. In *27th AIPR Workshop: Advances in Computer-Assisted Recognition*, pages 221–232, 1999.

[76] M. Hein, J. Audibert, and U. Von Luxburg. From graphs to manifolds - weak and strong pointwise consistency of graph Laplacians. *Conference on Learning Theory*, pages 470–485, 2005.

[77] M. Hein and S. Setzer. Beyond spectral clustering - tight relaxations of balanced graph cuts. *Advances in Neural Information Processing Systems*, pages 2366–2374, 2011.

[78] 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*, pages 67–78. International Society for Optics and Photonics, 2004.

[79] Huiyi Hu, Thomas Laurent, Mason A Porter, and Andrea L Bertozzi. A method based on total variation for network modularity optimization using the mbo scheme. *SIAM Journal on Applied Mathematics*, 73(6):2224–2246, 2013.

[80] 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*, pages 209–222. Springer, 2015.

[81] Chih-Cheng Hung, Sameer Kulkarni, and Bor-Chen Kuo. A new weighted fuzzy c-means clustering algorithm for remotely sensed image classification. *IEEE Journal of Selected Topics in Signal Processing*, 5(3):543–553, 2011.

[82] Tâm Huynh, Mario Fritz, and Bernt Schiele. Discovery of activity patterns using topic models. In *Proceedings of the 10th international conference on Ubiquitous computing*, pages 10–19. ACM, 2008.

[83] Anil K Jain. Data clustering: 50 years beyond k-means. *Pattern recognition letters*, 31(8):651–666, 2010.

[84] Luis O Jimenez and David A Landgrebe. Hyperspectral data analysis and supervised feature reduction via projection pursuit. *IEEE Transactions on Geoscience and Remote Sensing*, 37(6):2653–2667, 1999.

[85] Sinthop Kaewpijit, Jacqueline Le Moigne, and Tarek El-Ghazawi. Automatic reduction of hyperspectral imagery using wavelet spectral analysis. *IEEE Transactions on Geoscience and Remote Sensing*, 41(4):863–871, 2003.

[86] Vahid Kiani and Hamid Reza Pourreza. Robust gme in encoded mpeg video. In *Proceedings of the 9th International Conference on Advances in Mobile Computing and Multimedia*, pages 147–154. ACM, 2011.

[87] Jae-Gon Kim, Hyun Sung Chang, Jinwoong Kim, and Hyung-Myung Kim. Efficient camera motion characterization for mpeg video indexing. In *Multimedia and Expo, 2000. ICME 2000. 2000 IEEE International Conference on*, volume 2, pages 1171–1174. IEEE, 2000.

[88] Kris M Kitani, Takahiro Okabe, Yoichi Sato, and Akihiro Sugimoto. Fast unsupervised ego-action learning for first-person sports videos. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 3241–3248. IEEE, 2011.

[89] Robert V Kohn and Peter Sternberg. Local minimisers and singular perturbations. *Proceedings of the Royal Society of Edinburgh: Section A Mathematics*, 111(1-2):69–84, 1989.

[90] Balaji Krishnapuram, Lawrence Carin, Mario A T Figueiredo, and Alexander J Hartemink. Sparse multinomial logistic regression: Fast algorithms and generalization bounds. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(6):957–968, 2005.

[91] Da Kuang, Alex Gittens, and Raffay Hamid. Hardware compliant approximate image codes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 924–932, 2015.

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

[93] A. Levin, A. Rav-Acha, and D. Lischinski. Spectral matting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(10):1699 –1712, 2008.

[94] Jun Li, José M Bioucas-Dias, and Antonio Plaza. Semisupervised hyperspectral image segmentation using multinomial logistic regression with active learning. *IEEE Transactions on Geoscience and Remote Sensing*, 48(11):4085–4098, 2010.

[95] Jun Li, José M Bioucas Dias, and Antonio Plaza. Spectral–spatial hyperspectral image segmentation using subspace multinomial logistic regression and Markov random fields. *IEEE Transactions on Geoscience and Remote Sensing*, 50(3):809–823, 2012.

[96] Y. Lou, X. Zhang, S. Osher, and A.L. Bertozzi. Image recovery via nonlocal operators. *Journal of Scientific Computing*, 42(2):185–197, 2010.

[97] Bruce D Lucas, Takeo Kanade, et al. An iterative image registration technique with an application to stereo vision. 1981.

[98] Paul Lukowicz. Human computer interaction in context aware wearable systems. *Artificial Intelligence in Medicine*, pages 7–10, 2005.

[99] Xiyang Luo and Andrea L Bertozzi. Convergence analysis of the graph Allen-Cahn scheme. submitted, 2016.

[100] Li Ma, Melba M Crawford, and Jinwen Tian. Local manifold learning-based-nearest-neighbor for hyperspectral image classification. *IEEE Transactions on Geoscience and Remote Sensing*, 48(11):4099–4109, 2010.

[101] James MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297. Oakland, CA, USA., 1967.

[102] J. Mairal, M. Elad, and G. Sapiro. Sparse representation for color image restoration. *Image Processing, IEEE Transactions on*, 17(1):53–69, 2008.

[103] Dimitris Manolakis, Thristina Siracusa, and Gary Shaw. Adaptive matched subspace detectors for hyperspectral imaging applications. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 5, pages 3153–3156, 2001.

[104] Walterio W Mayol and David W Murray. Wearable hand activity recognition for event summarization. In *Wearable Computers, 2005. Proceedings. Ninth IEEE International Symposium On*, pages 122–129. IEEE, 2005.

[105] Farid Melgani and Lorenzo Bruzzone. Classification of hyperspectral remote sensing images with support vector machines. *IEEE Transactions on Geoscience and Remote Sensing*, 42(8):1778–1790, 2004.

[106] Zhaoyi Meng, Alice Koniges, Yun (Helen) He, Samuel Williams, Thorsten Kurth, Brandon Cook, Jack Deslippe, and Andrea 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.

[107] Zhaoyi Meng, Ekaterina Merkurjev, Alice Koniges, and Andrea L Bertozzi. Hyperspectral image classification using graph clustering methods. *Image Processing On Line*, 7:218–245, 2017.

[108] Zhaoyi Meng, Javier Sánchez, Jean-Michel Morel, and Andrea L Bertozzi. Ego-motion classification for body-worn videos. In *Imaging, vision and learning based on optimization and PDEs, accepted*, 2017.

[109] Ekaterina Merkurjev, Egil Bae, Andrea L Bertozzi, and Xue-Cheng Tai. Global binary optimization on graphs for classification of high-dimensional data. *Journal of Mathematical Imaging and Vision*, 52(3):414–435, 2015.

[110] Ekaterina Merkurjev, Andrea Bertozzi, Xiaoran Yan, and Kristina Lerman. Modified Cheeger and Ratio cut methods using the Ginzburg-Landau functional for classification of high-dimensional data. *Inverse Problems*, 33(7):074003, 2017.

[111] Ekaterina Merkurjev, Cristina Garcia-Cardona, Andrea L Bertozzi, Arjuna Flenner, and Allon G Percus. Diffuse interface methods for multiclass segmentation of high-dimensional data. *Applied Mathematics Letters*, 33:29–34, 2014.

[112] Ekaterina Merkurjev, Tijana Kostic, and Andrea L Bertozzi. An MBO scheme on graphs for classification and image processing. *SIAM Journal on Imaging Sciences*, 6(4):1903–1930, 2013.

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

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

[115] B. Merriman, J.K. Bence, and S.J. Osher. Motion of multiple functions: a level set approach. *Journal of Computational Physics*, 112(2):334–363, 1994.

[116] B. Merriman and S.J. Ruuth. Diffusion generated motion of curves on surfaces. *Journal of Computational Physics*, 225(2):2267–2282, 2007.

[117] Barry Merriman, James K Bence, and Stanley J Osher. Motion of multiple junctions: A level set approach. *Journal of Computational Physics*, 112(2):334–363, 1994.

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

[119] David Mumford and Jayant Shah. Optimal approximations by piecewise smooth functions and associated variational problems. *Communications on Pure and Applied Mathematics*, 42(5):577–685, 1989.

[120] M.M. Naeini, G. Dutton, K. Rothley, and G. Mori. Action recognition of insects using spectral clustering. *IAPR Conference on Machine Vision Applications*, pages 1–4, 2007.

[121] Saeid Niazmardi, Saeid Homayouni, and Abdolreza 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(2):831–839, 2013.

[122] S. Osher and J. Shen. Digitized PDE method for data restoration. *Analytic-Computational Methods in Applied Mathematics*, 698, 2000.

[123] P. Perona and L. Zelnik-Manor. Self-tuning spectral clustering. *Advances in Neural Information Processing Systems*, 17:1601–1608, 2004.

[124] G. Peyré, S. Bougleux, and L.D. Cohen. Non-local regularization of inverse problems. *Inverse Problems and Imaging*, 5(2):511–530, 2011.

[125] Hamed Pirsiavash and Deva Ramanan. Detecting activities of daily living in first-person camera views. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2847–2854. IEEE, 2012.

[126] Xiaofeng Ren and Chunhui Gu. Figure-ground segmentation improves handled object recognition in egocentric video. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 3137–3144. IEEE, 2010.

[127] J. Rubinstein, P. Sternberg, and J.B. Keller. A simple proof of convergence for an approximation scheme for computing motions by mean curvature. *SIAM Journal of Applied Mathematics*, 49(1):116–133, 1989.

[128] S.J. Ruuth. Efficient algorithms for diffusion-generated motion by mean curvature. *Journal of Computational Physics*, 144(2):603–625, 1998.

[129] Javier Sánchez. The inverse compositional algorithm for parametric registration. *Image Processing On Line*, 6:212–232, 2016.

[130] Javier Sánchez and Jean-Michel Morel. Motion smoothing strategies for video stabilization. *In Preparation*.

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

[132] Hong Shi, Yi Shen, and Zhiyan Liu. Hyperspectral bands reduction based on rough sets and fuzzy c-means clustering. In *Proceedings of the 20th IEEE Instrumentation and Measurement Technology Conference*, volume 2, pages 1053–1056, 2003.

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

[134] Ekaterina H Spriggs, Fernando De La Torre, and Martial Hebert. Temporal segmentation and activity classification from first-person sensing. In *Computer Vision and Pattern Recognition Workshops, 2009. CVPR Workshops 2009. IEEE Computer Society Conference On*, pages 17–24. IEEE, 2009.

[135] Thad Starner, Joshua Weaver, and Alex Pentland. Real-time american sign language recognition using desk and wearable computer based video. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(12):1371–1375, 1998.

[136] Li Sun, Ulrich Klank, and Michael Beetz. EYEWATCHME-3D hand and object tracking for inside out activity analysis. In *Computer Vision and Pattern Recognition Workshops, 2009. CVPR Workshops 2009. IEEE Computer Society Conference on*, pages 9–16. IEEE, 2009.

[137] Sudeep Sundaram and Walterio W Mayol Cuevas. High level activity recognition using low resolution wearable vision. In *Computer Vision and Pattern Recognition Workshops, 2009. CVPR Workshops 2009. IEEE Computer Society Conference on*, pages 25–32. IEEE, 2009.

[138] Justin Sunu, Jen-Mei Chang, and Andrea L Bertozzi. Simultaneous spectral analysis of multiple video sequence data for LWIR gas plumes. In *SPIE conference on Defense, Security, and Sensing*, 2014.

[139] Richard Szeliski. *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010.

[140] A. Szlam and X. Bresson. A total variation-based graph clustering algorithm for Cheeger ratio cuts. *International Conference on Machine Learning*, pages 1039–1046, 2010.

[141] Yuliya Tarabalka, Jón Atli Benediktsson, Jocelyn Chanussot, and James C Tilton. Multiple spectral–spatial classification approach for hyperspectral data. *IEEE Transactions on Geoscience and Remote Sensing*, 48(11):4122–4132, 2010.

[142] 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.

[143] Y. van Gennip and A.L. Bertozzi. Gamma-convergence of graph Ginzburg-Landau functionals. *Advanced in Differential Equations*, 17(11–12):1115–1180, 2012.

[144] Yves Van Gennip, Nestor Guillen, Braxton Osting, and Andrea L Bertozzi. Mean curvature, threshold dynamics, and phase field theory on finite graphs. *Milan Journal of Mathematics*, 82(1):3–65, 2014.

[145] Vladimir Vapnik. *The nature of statistical learning theory*. Springer Science & Business Media, 2013.

[146] 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(3):271–293, 2002.

[147] Pascal Vincent and Yoshua Bengio. Kernel matching pursuit. *Machine Learning*, 48(1-3):165–187, 2002.

[148] Ulrike Von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, 2007.

[149] J. Wang, T. Jebara, and S.F. Chang. Graph transduction via alternating minimization. *International Conference on Machine Learning*, pages 1144–1151, 2008.

[150] Jing Wang and Chein-I Chang. Independent component analysis-based dimensionality reduction with applications in hyperspectral image analysis. *IEEE Transactions on Geoscience and Remote Sensing*, 44(6):1586–1600, 2006.

[151] Samuel Williams, Andrew Waterman, and David Patterson. Roofline: an insightful visual performance model for multicore architectures. *Communications of the ACM*, 52(4):65–76, 2009.

[152] Xiao-Tong Yuan, Xiaobai Liu, and Shuicheng Yan. Visual classification with multitask joint sparse representation. *IEEE Transactions on Image Processing*, 21(10):4349–4360, 2012.

[153] Liangpei Zhang, Yanfei Zhong, Bo Huang, Jianya Gong, and Pingxiang Li. Dimensionality reduction based on clonal selection for hyperspectral imagery. *IEEE Transactions on Geoscience and Remote Sensing*, 45(12):4172–4186, 2007.

[154] T.F. Zhang, X.and Chan. Wavelet inpainting by nonlocal total variation. *Inverse Problems and Imaging*, 4(1):191–210, 2010.

[155] D. Zhou, O. Bousquet, T.N. Lal, J. Weston, and B. Schölkopf. Learning with local and global consistency. *Advances in Neural Information Processing Systems*, 16:321–328, 2004.

[156] D. Zhou and B. Schölkopf. A regularization framework for learning from graph data. *International Conference on Machine Learning*, 2004.

[157] Dengyong Zhou, Olivier Bousquet, Thomas N. Lal, Jason Weston, and Bernhard Schölkopf. Learning with local and global consistency. In Sebastian Thrun, Lawrence K. Saul, and Bernhard Schölkopf, editors, *Advances in Neural Information Processing Systems 16*, pages 321–328. MIT Press, Cambridge, MA, 2004.

[158] Dengyong Zhou, Thomas Hofmann, and Bernhard Schölkopf. Semi-supervised learning on directed graphs. In *Advances in neural information processing systems*, pages 1633–1640, 2004.

[159] Dengyong Zhou, Jiayuan Huang, and Bernhard Schölkopf. Learning from labeled and unlabeled data on a directed graph. In *Proceedings of the 22nd international conference on Machine learning*, pages 1036–1043. ACM, 2005.

[160] Dengyong Zhou and Bernhard Schölkopf. Regularization on discrete spaces. In *Joint Pattern Recognition Symposium*, pages 361–368. Springer, 2005.

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

[162] X. Zhu. Semi-supervised learning literature survey. *Computer Sciences Technical Report 1530, University of Wisconsin-Madison*, 2005.