Title
Graph-Based Data Fusion Methods

Author
Iyer, Geoffrey

Publication Date
2018

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA
Los Angeles

Graph-Based Data Fusion Methods

A dissertation submitted in partial satisfaction
of the requirements for the degree
Doctor of Philosophy in Mathematics

by

Geoffrey Sankar Iyer

2018

ABSTRACT OF THE DISSERTATION

Graph-Based Data Fusion Methods

by

Geoffrey Sankar Iyer
Doctor of Philosophy in Mathematics
University of California, Los Angeles, 2018
Professor Andrea Bertozzi, Chair

As data of all kinds becomes more readily available, there is an increasing demand for algorithms that can jointly process data from multiple sources, also called *modalities*. In the current state-of-the-art the are many such algorithms, but for the most part each method is made with a specific application in mind. In this work we aim to create more general and data-driven methods for various multimodal machine learning problems. We approach this difficult problem from the standpoint of graph methods, as graph representations are historically robust to many possible data formats, while maintaining sufficient information to produce state-of-the-art results.

The first problem considered here is a segmentation problem in the case of co-registered, multimodal datasets. Here we perform data fusion on the level of graph representations, specifically concentrating on finding and using the unique information that each modality may bring to the overall scene. From the fused graph we implement standard image segmentation techniques.

We also consider here a matching problem between arbitrary datasets. Once again, graph representations are used to preserve relevant topological information while filtering out specific formatting details. We then match graph nodes using spectral information. By using the Nyström extension eigensolver to quickly calculate approximate graph eigenfunctions, and by using a hierarchical matching algorithm narrow the matching search space, we obtain a runtime and space complexity that is, to the best of our knowledge, superior among the

state-of-the-art.

The dissertation of Geoffrey Sankar Iyer is approved.

Luminita Aura Vese

Stanley J Osher

Christopher R Anderson

Jocelyn Chanussot

Andrea Bertozzi, Committee Chair

University of California, Los Angeles

2018

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I would like to thank my thesis advisors, Andrea Bertozzi and Jocelyn Chanussot. Andrea, when I was unsure about my research topic and struggling to find a place here at UCLA, you put in the time to show me all the wonderful things that can be done in applied math and helped me to find my subject for this dissertation. Jocelyn, your help and advice brought me to more places than I ever thought possible. Thank you especially for the wonderful year in Grenoble, and more generally for the invaluable advice and guidance you have given me throughout our time together.

I also would like to thank my father, Hariharan Iyer, for all of our discussions over the years.

This dissertation includes content from the previously published work [ICB17] , available online at http://www.math.ucla.edu/ bertozzi/papers/ICIP_2017_final.pdf

This work was supported by NSF grant DMS-1118971, ONR grant N00014-16-1-2119, NSF grant DMS-1417674, European Research Council (Grant no. 320684 - CHESS project), and CNRS (Grant no. PICS-USA 263484).

This material is based upon research supported by the Chateaubriand Fellowship of the Office for Science & Technology of the Embassy of France in the United States.

VITA

2012	B.S. (Mathematics), University of Michigan, Ann Arbor.
2013-2017	Teaching Assistant, Mathematics Department, UCLA
2012-2017	Research Assistant, Mathematics Department, UCLA
2017-2018	Visiting Scientist, GIPSA Lab, Université Grenoble Alpes

PUBLICATIONS

G. Iyer, J. Chanussot, A. L. Bertozzi. "A GRAPH-BASED APPROACH FOR FEATURE EXTRACTION AND SEGMENTATION OF MULTIMODAL IMAGES" In International Conference on Image Processing, 2017

G. Iyer, J. Chanussot, A. L. Bertozzi. "A GRAPH-BASED APPROACH FOR DATA FUSION AND SEGMENTATION OF MULTIMODAL IMAGES", currently in revision with IEEE Transactions on Geoscience and Remote Sensing

CHAPTER 1

Introduction

For any event or scene of interest, there are a wide variety of possible sensors, measurement techniques, or experimental setups with which one may gain relevant data. However, it is rare for any single acquisition method to provide sufficient information to fully explain the process in question. Therefore, in pursuit of a more perfect understanding, it is generally desirable to collect information from many different sources, also called *modalities*, and create algorithms that can jointly process all of the input data. For example, in the area of speech recognition, integrating audio data with a video of the speaker results in a much more accurate classification [PNG03, SBR16]. Similarly, in medicine, it is possible to fuse the results of two different types of brain imaging to create a final image with better resolution than either of the originals [LVY12, SSJ16].

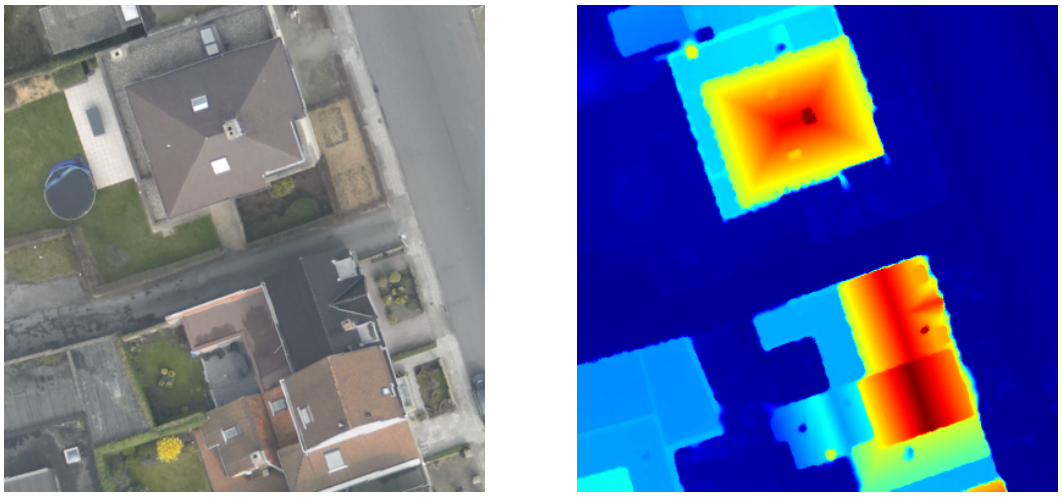
To properly take advantage of a more robust set of data, multimodal methods must in some way tackle the difficulty of correlating information between modalities that may not share a common output format. This space of research is commonly referred to as *data fusion*, and is the overarching topic of this dissertation. In the current state of the literature, the majority of data fusion algorithms are created with specific applications, models, or datasets in mind, and are therefore often not useful in a more general sense [LAJ15]. In this work we instead focus on more data-driven fusion methods through use of graphs. The graph structure is particularly well-suited to this type of problem, as it gives a representation of the data that preserves all relevant topological and geometric information, while at the same time removing much of the specific formatting. Thus, as we will show in throughout this dissertation, we are able to approach many common fusion problems in a very general setting, allowing for straightforward application to a diverse group of datasets.

The original work in this thesis is derived from one publication and two submitted manuscripts. In chapter 3 we present our work on segmentation of co-registered multimodal datasets [ICB17] [ICB18b], which contains a novel method of fusing data based on graph weights, as well as an application of previously developed segmentation methods to this new setting. Chapter 4 discusses the manuscript [ICB18a], covering a new hierarchical algorithm for graph matching that makes significant improvements in performance over the current state-of-the-art. In addition to this original work, we also present a background of common concepts in spectral graph theory in chapter 2.

1.1 Co-registered Image Segmentation

As a first foray into multimodal data fusion, we focus on the case of *co-registered* datasets, with applications towards segmentation. By *co-registered*, we mean that each modality contains the same number of observations, and these obversations share a common indexing. For example, this is often the case in multimodal timeseries data, as in most experiments all sensors will capture new data at the same time. Another common example of co-registered data, and the one we will focus on most in this paper, is in multimodal images, where the same scene may be captured by many different sensors. In figure 1.1 we show an example multimodal dataset from the 2015 IEEE Data Fusion Challenge [CRG16] (abbreviated as DFC2015), which consists of an optical and a lidar (elevation) image of a residential neighborhood in Belgium. This particular dataset is interesting because of the large amount of non-redundancy between the two images. By using the lidar data, one can easily differentiate the roofs of the buildings from the adjacent streets, even though they are roughly the same color. Conversely, the optical data allows one to separate the many different objects at ground-level, even though they appear the same in the lidar modality. Therefore one would expect that an algorithm that processes the two sources together would produce much more accurate segmentation results than could be obtained by dealing with the modalities separately. We will revisit this dataset in section 3.3 to show that this is indeed the case.

Our method first creates a graph representation of each separate modality, then merges



(a) DFC2015 optical data

(b) DFC2015 lidar data

Figure 1.1: DFC2015 Input Data

these representations using the co-registration assumption (3.2.1). From this, we get a single graph that constitutes a fusion of the original input information. We then proceed to perform feature extraction and segmentation on this graph using various well-established methods. Specifically, we extract features of the graph by finding the eigenvectors of the graph Laplacian (2.2), then use these features as inputs to the Spectral Clustering (3.2.2) and Graph MBO (3.2.3) algorithms. Finally, in 3.3 we show the results of the method applied to several optical/lidar datasets in various different contexts.

1.2 Fast Graph Matching

In addition to considering data fusion in the co-registered above, in this thesis we are also interested in problems where the data comes with minimial or no pre-defined registration. In pursuit of this goal, we first formulate and solve a matching problem based on graph representations of our data. A fundamental problem in theoretical computer science, graph matching has been applied to many different topics in computer vision and pattern recognition, such as object recognition [BMP02, BBM05], shape matching [BBM05, SSD98, HH99], and video indexing [SBV00].

The study of graph matching originally began with the graph isomorphism problem, which attempts to find an exact isomorphism between two given graphs. This problem is interesting in that it is one of the few problems in NP not known to be in P or NP-complete, with the recent state-of-the-art solving the problem in quasipolynomial time [Bab15,ABD17]. However, as most real-world applications involve graphs where no exact isomorphism exists, the majority of attention over the last 30 years has shifted towards inexact matching problems. The most general of these is the *inexact subgraph matching problem*, which searches for an embedding of a small graph into a larger that minimizes the difference between the original graph and it's image. Much more difficult that the graph isomorphism problem, all subgraph problems problem are clearly NP-complete, as they can be thought of as a generalization of the Hamiltonian cycle problem. It is on this style of problem that we focus our attention in this paper, as the greater generality of inexact subgraph matching allows for wider applications.

While all graph matching problems involve optimizations over discrete constraints, most state-of-the-art methods attack the problem via some continuous relaxation. Each algorithm then finishes with a discretization step, recovering the sought after matching. However, this discretization step presents a significant problem in terms of time complexity, as the most common tool in the discretization of continuous solutions is the *Hungarian Algorithm* [W55], which solves a linear assignment problem in $\mathcal{O}(N^3)$ time. Although these methods have proven to be both robust and accurate, the issue of runtime provides some serious restrictions in application to modern datasets. For example, current state-of-the-art graph representations of human brains use $\mathcal{O}(10^6)$ nodes and $\mathcal{O}(10^8)$ edges [RKM13], a figure that is entirely intractible for current matching methods.

To combat this issue, we introduce in this paper two hierarchical algorithm for graph matching, called Hierararchical One-to-One Spectral Matching (HOSM), and Hierarchical Many-to-Many Spectral Matching(HMSM). The crux of our algorithm involves dividing the full graph matching problem into many subproblems, thereby circumventing the runtime issues of classical matching methods. In the end we are able to obtain a time and space complexity that significantly outperforms the current state-of-the-art, while still maintaining

reasonable accuracy. After a review of other graph matching techniques in section 4.1, we explain our method in section 4.2, and give experimental results both on synthetic and real-world data in section 4.3. Lastly, we show in section 4.4 several applications of this algorithm to common machine learning problems.

CHAPTER 2

Background

In this chapter we present a general survey of the theory most relevant to our work. In each of our applications, we represent data via a graph $G = (V, E)$, where $V = \{v_1, \ldots, v_N\}$ is the set of graph nodes and E is the set of edges. In this paper, all graphs will be undirected and weighted. We represent this via a symmetric $|V| \times |V|$ weight matrix W, also called the *similarity matrix*, where w_{ij} is the weight of the edge joining the nodes v_i, v_j. For each node $v_i \in V$ we define the *degree* of the node

$$d_i = \sum_j w_{ij}, \tag{2.1}$$

and let D be the diagonal matrix consisting of the d_i.

2.1 Graph Similarity Matrix

Given a graph $G = (V, E)$, we create the weight matrix W based on some weight function $w : V \times V \to \mathbb{R}_{\geq 0}$. The goal of the weight function is to represent similarity between graph

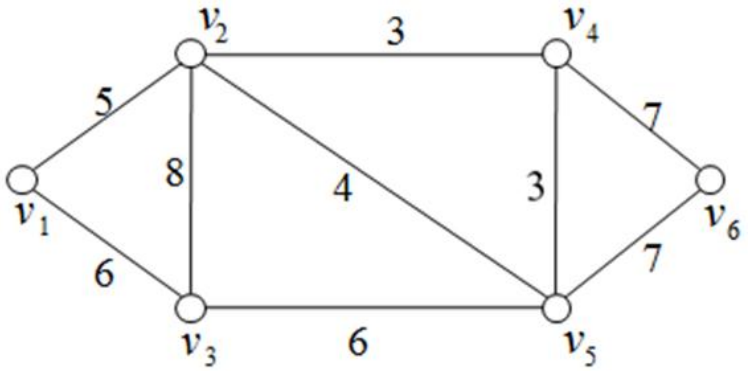


Figure 2.1: Example weighted graph

nodes. w_{ij} should be a large value if v_i, v_j are similar, and a small value if not. A common choice for weight function is a radial basis function of type

$$w(x, y) = \exp\left(-\frac{d(x, y)^2}{\sigma^2}\right), \tag{2.2}$$

where $d(x, y)$ is a distance measure on the set of nodes V, and σ is a scaling parameter. For example, if the graph nodes V are derived from points in a Euclidean space $\mathbb{R}^n$, then a reasonable choice for $d(x, y)$ would be the standard Euclidean distance. As another example, when creating graphs based on hyperspectral data, or another high-dimensional input source, a vector angle measure could give a better representation of the underlying scene. In chapter 3 we discuss how to extend common choices for graph weight functions to the case of multimodal data.

Note that it is not necessary to have a *complete graph*, that is, a graph with nonzero weights on all edges. Many applications use sparse similarity matrices in order to decrease computation time. For example, a *k-nearest neighbors graph* only gives a nonzero weight between vertices v_i, v_j if v_i is among the nearest neighbors of v_j, or vice versa. A *mutual k-nearest neighbors graph* instead includes an edge between v_i, v_j if both vertices are k-nearest neighbors of each other. Another possible method for sparsifying the graph is to apply a threshold to the edge weights, although this does require calculating the entire $|V| \times |V|$ weight matrix, which may become too computationally expensive.

2.2 Graph Laplacian

One major tool in the study of graphs is the *graph Laplacian*. There are many versions of the graph Laplacian considered in the literature, but we limit our focus here to the most common three:

- The unnormalized Laplacian $L = D - W$,
- The symmetric Laplacian $L_s = I - D^{-\frac{1}{2}} W D^{-\frac{1}{2}}$,
- The random-walk Laplacian $L_{rw} = I - D^{-1} W$.

One common way to view the graph Laplacian is as an approximation to the standard Laplace-Beltrami operator on manifolds. Given a smooth manifold, one can choose a discretization and define the graph Laplacian using the manifold distance on these points. Under reasonable assumptions on the discretization structure, as well as the weight function $w(x, y)$, each of the Laplacians above has been shown to converge to the Laplace-Beltrami operator as the mesh size goes to zero [Xu04, DRW10].

The graph Laplacian L can easily be shown to have the following properties [Lux07], [Chu97]:

1. L is symmetric and positive semi-definite.
2. The smallest eigenvalue of L is 0, with corresponding eigenvector $\mathbf{1}_{|L|}$.
3. The multiplicity of the eigenvalue 0 is equal to the number of connected components of the graph G.

In addition, the Laplacians L_s, L_{rw} have the following properties

1. L_s is symmetric and positive semi-definite.
2. L_{rw} is positive semi-definite.
3. λ is an eigenvalue of L_{rw} with eigenvector u if and only if λ is an eigenvector of L_s with eigenvector $D^{\frac{1}{2}}u$.
4. The multiplicity of the eigenvalue 0 is equal to the number of connected components of the graph G.

2.2.1 Laplacian Eigenmap Embedding

One common use of the graph Laplacian is as a tool for dimension reduction and data representation. Given a graph $G = (V, E)$ with weight matrix W as above, we consider the problem of embedding the graph into $\mathbb{R}^m$ for some $m \geq 1$, with the goal of having similar

graph nodes stay as close together as possible. To this end we define the following objective function

$$\frac{1}{2}\sum_{i,j}\left\|y_i - y_j\right\|^2 w_{ij} \tag{2.3}$$

where $y_i \in \mathbb{R}^m$ is the embedded image of the i-th graph node v_i. In the special case where $m = 1$ (an embedding onto a line), we have that

$$\begin{aligned}
\frac{1}{2}\sum_{i,j}\left\|y_i - y_j\right\|^2 w_{ij} &= \frac{1}{2}\sum_{i,j}\left(y_i - y_j\right)^2 w_{ij} && (2.4)\\
&= \frac{1}{2}\sum_{i,j}\left(y_i^2 + y_j^2 - 2y_i y_j\right) w_{ij} && (2.5)\\
&= \frac{1}{2}\left(\sum_i y_i^2 d_i + \sum_j \left\|y_j\right\|^2 d_j - 2\sum_{i,j} y_i y_j w_{ij}\right) && (2.6)\\
&= y^T\left(D - W\right) y && (2.7)\\
&= y^T L y, && (2.8)
\end{aligned}$$

where $y = \left(y_1, \ldots, y_{|V|}\right)$ is the vector giving the full embedding.

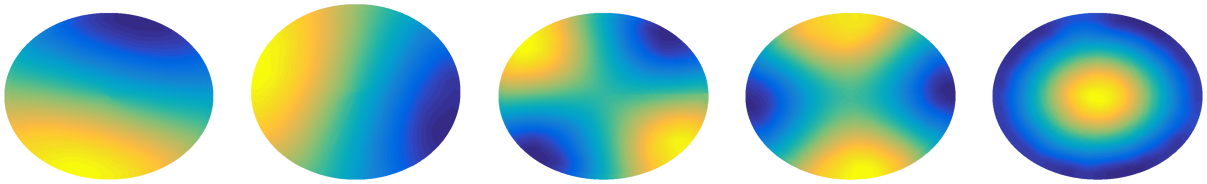
Based on this, we define the minimization problem for the case $m = 1$ as

$$\underset{\substack{y^T D y = 1\\ y^T D\mathbf{1} = 0}}{\operatorname{argmin}} y^T L y, \tag{2.9}$$

with the generalization to arbitrary m being

$$\underset{\substack{Y^T D Y = I\\ Y^T D\mathbf{1} = 0}}{\operatorname{argmin}} \operatorname{tr}\left(Y^T L Y\right), \tag{2.10}$$

where $Y \in \mathbb{R}^{|V|\times m}$ is the matrix with y_i as the i-th row. Here the $y^T D y = 1$ requirement serves the dual purpose of removing an arbitrary scaling factor from the solution (preventing the trivial solution $y = 0$), as well as assigning importance to individual graph vertices based on their degree. The second constraint $y^T D\mathbf{1} = 0$ removes the other trivial solution of collapsing all vertices to a single point by forcing the embeddings to be orthogonal to the constant vector. In a sense, this constraint can be thought of as removing a translational invariance from y. And of course, the corresponding constraints in the arbitrary m case perform the same task in this higher dimensional setting.



(a) v_2 (b) v_3 (c) v_4 (d) v_5 (e) v_6

Figure 2.2: Laplace-Beltrami eigenfunctions on the disc

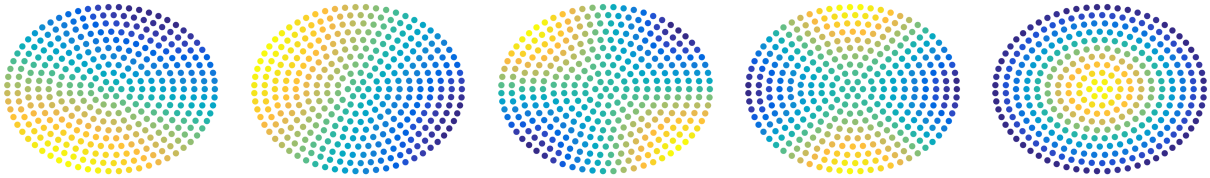
It is then shown in [BN03] that this energy is minimized by letting the solution matrix Y be the matrix of eigenvectors corresponding to the smallest non-zero eigenvalues of the generalized eigenproblem

$$Ly = \lambda Dy. \tag{2.11}$$

It is further shown in [Lux07] that this is the same as the eigenvectors of the random-walk graph Laplacian

$$L_r y = \lambda y. \tag{2.12}$$

In many ways, the applications of the graph Laplacian resemble those of the classical Laplace-Beltrami operator on manifolds. Just as Laplace-Beltrami eigenfunctions contain information about the symmetries and geometry of the underlying manifold, the graph Laplacian has the same use in the discrete case. Similar to classical Fourier analysis, the associated eigenvalue quantifies the "frequency" of the eigenfunction. In manifolds without boundary or with appropriate boundary conditions the first eigenfunction is always constant, with eigenvalue 0, and the low-value eigenfunctions correspond to the low-frequency symmetries of the manifold. In figures 2.2 and 2.3 we show a comparison of the first few non-trivial eigenfunctions of the Laplace-Beltrami operator on the unit disc, and of the graph Laplacian on a discretization. As we will explain further in chapter 3, these eigenfunctions are very useful in graph segmentation and classification problems.



(a) v_2 (b) v_3 (c) v_4 (d) v_5 (e) v_6

Figure 2.3: Graph Laplacian eigenfunctions on discrete disc

2.3 Nyström Extension

As discussed above, the eigenfunctions of the graph Laplacian are deeply connected with the geometric information of the graph. However, solving an eigenproblem on a complete graph with N nodes is an $O(N^3)$ process, which greatly limits the applicability of this theory. To combat this issue, we apply the Nyström Extension to find approximate eigenfunctions in significantly reduced time.

The classiscal Nyström Extension [Nys30, PTV92, Art79] is a technique for finding numerical approximations to eigenfunctions of the form

$$\int_a^b W(x,y)\phi(y)dy = \lambda\phi(y). \tag{2.13}$$

Approximating this integral by discretizing to a set of n evenly-spaced points $\xi_1, \ldots, \xi_n$ yields the equation

$$\frac{b-a}{n}\sum_{j=1}^{n} W(x,\xi_j)\hat{\phi}(\xi_j) = \lambda\hat{\phi}(x), \tag{2.14}$$

where $\hat{\phi}$ here is our approximation to the eigenfunction ϕ. Setting $x = \xi_i$ for $i = 1, \ldots, n$ gives a system of linear equations

$$\frac{b-a}{n}\sum_{j=1}^{n} W(\xi_i,\xi_j)\hat{\phi}(\xi_j) = \lambda\hat{\phi}(\xi_i) \quad i = 1, \ldots, n. \tag{2.15}$$

This system of equations can be reframed as an eigenvalue problem of size $n \times n$

$$\boldsymbol{W}\hat{\boldsymbol{\Phi}} = n\hat{\boldsymbol{\Phi}}\boldsymbol{\Lambda}, \tag{2.16}$$

where $\boldsymbol{W}_{ij} = W(\xi_i, \xi_j)$. Solving this eigenproblem gives us n approximate eigenfunctions $\hat{\phi}_1, \ldots, \hat{\phi}_n$ defined on the subdomain $\{\xi_1, \ldots, \xi_n\}$. We then substitute back into (2.14) to extend these functions to the full domain

$$\hat{\phi}_i(x) = \frac{1}{n\lambda_i} \sum_{j=1}^{n} W(x, \xi_j)\hat{\phi}(\xi_j) \tag{2.17}$$

to obtain n approximate eigenfunctions for the original statement of the problem in (2.13).

Following the example of [WS01, BF12, FBC04, BFC02], we adapt the theory above to the discrete case for use in our graph problems. Specifically, we can calculate n approximate eigenfunctions of an $N \times N$ weight matrix W by solving an $n \times n$ eigenproblem and using the Nyström extension. As above, we choose n sample graph nodes, and re-index the matrix W as

$$W = \begin{bmatrix} A & B \\ B^T & C, \end{bmatrix} \tag{2.18}$$

where $A \in \mathbb{R}^{n \times n}$ gives the graph weights within the n sample nodes, $B \in \mathbb{R}^{n \times (N-n)}$ is the weights between the n sample nodes and the remaining $N-n$ nodes, and $C \in \mathbb{R}^{(N-n)\times(N-n)}$ is the weights within the unsampled nodes. If we diagonalize $A = U\Lambda U^T$, then the approximate eigenfunctions from (2.17) are

$$\bar{U} = \begin{bmatrix} U \\ B^T U \lambda^{-1}. \end{bmatrix} \tag{2.19}$$

Using these approximation of W associated to $\bar{U}$ is given by

$$\begin{aligned} \hat{W} &= \bar{U}\Lambda\bar{U}^T \\ &= \begin{bmatrix} U \\ B^T U \lambda^{-1} \end{bmatrix} \Lambda \begin{bmatrix} U^T & \Lambda^{-1} U^T B \end{bmatrix} \\ &= \begin{bmatrix} A & B \\ B^T & B^T A^{-1} B \end{bmatrix} \end{aligned} \tag{2.20}$$

In particular, using the Nyström extension, we avoid calculating C, and approximate it with $B^T A^{-1} B$. The quality of this approximation is defined by the extent to with C is spanned by the rows of B.

There remains one detail to finish calculating the approximate eigenfunctions, as the matrix $\bar{U}$ is not orthogonal. Here we follow the method presented in [FBC04]. Let $A^{1/2}$ denote the symmetric positive definite square root of A, define $S = A + A^{-1/2}BB^TA^{-1/2}$, and diagonalize it as $S = U_S\Lambda_S U_S^T$. Then $\hat{W}$ is diagonalized by the matrix

$$V = \begin{bmatrix} A \\ B^TA^{-1/2} \end{bmatrix} U_S\Lambda_S^{-1/2}, \tag{2.21}$$

and $VV^T = I$, as desired.

In this paper, we are specifically interested in the eigenfunctions of the symmetric graph Laplacian $L_s = I - D^{-\frac{1}{2}}WD^{-\frac{1}{2}}$. For this we apply the Nyström theory above to the normalized weight matrix $D^{-\frac{1}{2}}WD^{-\frac{1}{2}}$. It is, however, very computationally expensive to find the degree matrix D, as this involves calculating each of the N^2 graph weights. Instead we approximate D using $\hat{W}$ from (2.20). That is, we let $\hat{D}$ be the diagonal matrix

$$\begin{aligned} \operatorname{diag}(\hat{D}) &= \hat{W}\mathbf{1}_N \\ &= \begin{bmatrix} A\mathbf{1}_n + B\mathbf{1}_{N-n} \\ B^T\mathbf{1}_n + B^TA^{-1}B\mathbf{1}_{N-n}, \end{bmatrix} \end{aligned} \tag{2.22}$$

and perform Nyström on the normalized matrix

$$\left(D^{-\frac{1}{2}}WD^{-\frac{1}{2}}\right)_{ij} = \frac{W_{ij}}{\sqrt{d_id_j}} \tag{2.23}$$

There are many other possible methods to quickly calculate graph Laplacian eigenvectors over large datasets. One highly-effective method is to sparsify the graph, as explained in section 2.1, and apply a sparse eigensolver such as the Lanczos method [Lan50, Saa11], or the related Rayleigh-Chebyshev method [And10].

CHAPTER 3

Co-registered Image Segmentation

In this chapter we present our algorithm for segmentation of co-registered datasets. This algorithm inputs a group of co-registered datasets along with a relevant distance metric on each individual set, then performs data fusion and feature extraction on the input. Then using the newly calculated features we apply several segmentation algorithms to achieve the final result. The graph representation of data is in particular quite useful here, as we can make comparisons between graphs in situations where the raw data may not be directly comparable.

3.1 Related Work

One very simple algorithm for multimodal image fusion is to simply take a weighted average of the different modes. Unfortunately, this method is often too naive to produce meaningful results. In many cases there are various objects and regions that occur in multiple images but with opposite contrast, which would cancel out in an averaged image. However, this basic idea is still worth consideration, so long as the blending step is treated with more care. In [MLY17] the authors use structural patch decomposition to perform roughly the same task, but with much better results, and in [STC12] the authors address the same problem with probabilistic methods. In each of these cases, the end product is an image that contains the most relevant features from each modality. Classical segmentation algorithms can then be performed on this fused image to create the desired results. Somewhat related is the multimodal kmeans method presented in [YTL12], which minimizes a weighted average of a standard kernel K-means energy on each modality to achieve a segmentation on the data.

Another common way to fuse images is to transform each modality with some processing algorithm, then merge the data in the new feature space. In [Pie03] the authors follow this methodology, using a multiresolution (MR) transformation to process information in each modality. The benefit of this algorithm is that the transformation is fully invertible, meaning that once the data has been synthesized in the feature space, the inverse transformation can be applied to recover the fused image. In [CBC07, MS07] the authors follow the same overall strategy, using Independent Component Analysis (ICA) as the initial processing algorithm.

Each of the above methods first fuses the different modalities (into either a new image, or into a new set of features), then uses this fused data to create a final segmentation. But another valid method is to instead segment each modality first, then combine the different classifications into a final result. Both [TDC15] and [RKD15] create a hierarchical segmentation of each modality (a chain of segmentations ranging from very coarse to very fine), then blend these segmentations using some decision algorithm. A related field of study is segmentation combination. Given multiple segmentations of the same image (possibly obtained from different modalities), the goal is to obtain a consensus segmentation by somehow fusing the different inputs. In [FAV11] the authors accomplish this through general ensemble clustering methods, and in [WJR08] this is done by using probabilistic methods and random walks.

In regard to spectral graph theory, these methods have been very successfully applied to data clustering problems and image segmentation [CBS05, GS06, SM00]. Graph-cut algorithms are quite flexible. All that is required is a well-chosen affinity function to describe the similarity between different graph nodes. In [LPB15, DMH14], the authors create a sparse graph by using a k-nearest neighbors jointly over each input modality, and use an RBF SVM classifier on the resulting eigenvectors to achieve a final classification. In [EKB15] the authors create one graph Laplacian matrix for each modality, and find a single set of eigenvectors that approximately diagonalizes all Laplacians simultaneously. In general, graph cuts can even be used to minimize a wide variety of energy functions [KZ04], allowing for the use of unsupervised [HSB15, WMB14] or semi-supervised methods [MKB13]. The standard theory behind this is described in [Moh91], with a tutorial on spectral clustering given in [Lux07].

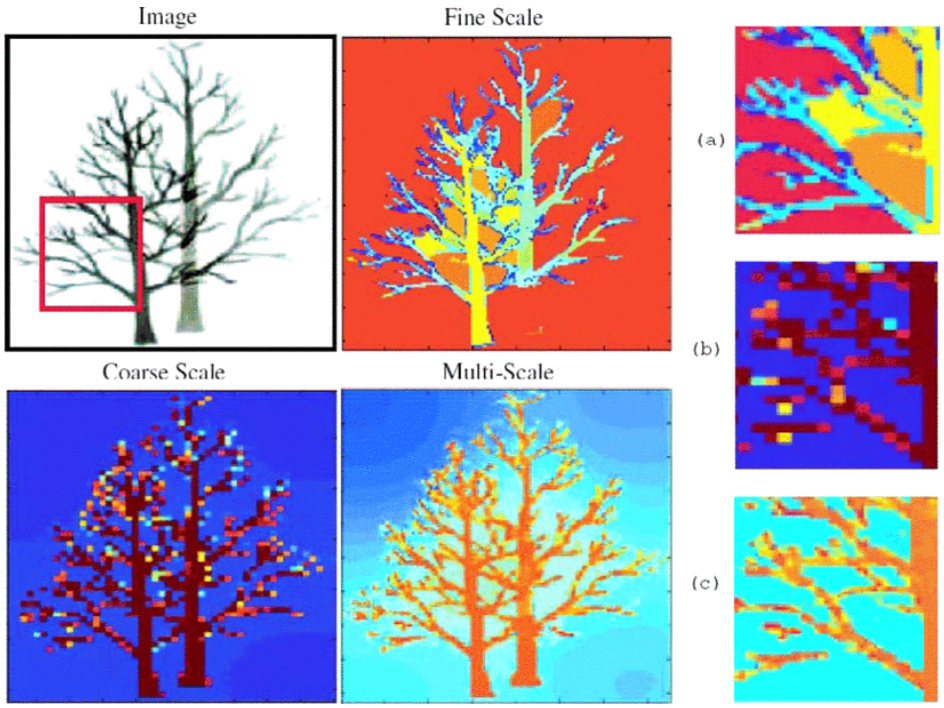


Figure 3.1: [CBS05] Spectral segmentation with multiscale graph decomposition.

In figure 3.1 we show an example result from [CBS05], where multiple graph cuts at different scales are simultaneously created then merged to better compromise between the overall representation and the fine detail of an image.

Although not entirely related to the work presented here, we give special mention to several domain adaptation and manifold learning techniques that depend on similar graph theory as our research. Given several input modalities viewed as $\mathbb{R}$-manifolds, the goal of these algorithms is to create maps from each manifold to a common $\mathbb{R}$-space - called a *latent space* - where structurally similar data in the original manifolds will map to nearby points in the latent space. These algorithms are well covered in the survey article [LPC14], but we shall mention a few that are based on graph spectral information. In [WM11] the authors create a large graph with similarity information from every pair of data points, both within and between modalities, and solve an eigenproblem on this graph to create the latent space embedding. In both this paper and in [YC13], SVM is then used on the latent space to transfer a known classification on one modality to the others. [TC15] applies a similar framework, adding kernel method to the similarity calculation to create a method more

robust to nonlinearities.

3.2 The Method

In this section we explain the theory behind the algorithm. First, in section 3.2.1 we cover the fusion step of the algorithm, in which we process each input modality together to create one weighted graph representing the entire dataset. We then exhibit two segmentation methods that we apply to the graph object. The first, *spectral clustering* 3.2.2, is an unsupervised method that can be used to quickly obtain a reasonable set of "proof-of-concept" results. The second, *graph MBO* 3.2.3, is a semisupervised method that more carefully handles the energy minimization to obtain a stronger final result.

3.2.1 Multimodal Graph Weights

Let k be the number of input modalities. For each $1 \leq \ell \leq k$, we have a dataset, which we will label X^ℓ, as well as a relevent distance function dist_ℓ. From this we create a graph-representation of each X^ℓ using the standard Gaussian-based weight function described in (2.2)

$$w_l(x,y) = \exp\left(-\frac{\text{dist}_\ell(x,y)^2}{\sigma_\ell^2}\right), \tag{3.1}$$

giving us a weight matrix W_ℓ associated to each input modality. To ensure the different weight matrices are reasonably comparable, we choose the scaling factor σ_ℓ based on the standard deviation of distances in the corresponding modality.

$$\sigma_\ell = \text{std}\left(\text{dist}_\ell(x,y) \mid x,y \in X^\ell\right). \tag{3.2}$$

Having created the individual graph representations of our modalities, we then use the co-registration assumption to form a single fused graph representing the entire input. For notation, let $N = |X^1| = \cdots = \left|X^k\right|$, and have x_i^ℓ denote the element of index i in X^ℓ. Furthermore, denote the full collection of elements of index i as $x_i = \left\{x_i^\ell \mid 1 \leq \ell \leq k\right\}$. We then define the fused graph similarity matrix by creating a notion of distance the collections

x_i

$$\text{dist}\left(x_i, x_j\right) = \max\left(\frac{\text{dist}_1\left(x_i^1, x_j^1\right)}{\sigma_1}, \cdots, \frac{\text{dist}_k\left(x_i^k, x_j^k\right)}{\sigma_k}\right), \tag{3.3}$$

and using this notion of distance in the standard Gaussian scheme (2.2)

$$W_{ij} = \exp\left(-\text{dist}\left(x_i, x_j\right)^2\right). \tag{3.4}$$

Note that this is equivalent to comparing the individual graph weight matrices and choosing the minimum value for each edge weight

$$W_{ij} = \min\left(W_{ij}^\ell \mid 1 \leq \ell \leq k\right). \tag{3.5}$$

The purpose of choosing the maximum of input distances to combine the individual dist_ℓ is to emphasize the unique information that each dataset brings. By using the maximum of all distances (i.e. the minimum of all similarities) two data points x_i, x_j are considered similar only when they are similar in every dataset. For example, in figure 1.1 the gray road and the gray rooftops are considered very similar in the RGB modality, but quite different in the lidar modality. Therefore, under this norm the two areas will be given a low similarity score, as desired. Of course, there are many other choices for combining the individual dist_ℓ, but both through heuristics and via experiments we have found the maximum to be the most effective. With any graph-based method, the choice of graph weights is always one of the leading concerns. For this reason, we have studied several different variations of equation (3.3), and we present our results in the section 3.4.

Note that if each dist_ℓ on X^ℓ is a formal metric, then dist defined on X will be as well. Furthermore, if we assume that each each X^ℓ is embedded in some $\mathbb{R}$-space and that dist_ℓ is induced by a norm on X^ℓ, then $\text{dist}(\cdot, 0)$ will be a norm on the concatenated set $\left(X^1, \ldots, X^k\right)$. One benefit of this approach to forming fused graph weights is that it allows us to adapt the treatment of each modality based on our knowledge of that particular source. For example, if a given modality is known to be represented poorly in Euclidean space (the Swiss Roll data is a well-known example) we can choose a metric more suited to that particular manifold. In the context of the work presented here, which focuses primarily on image segmentation,

we have found that the standard Euclidean distance or the vector angle distance produces the best description of the input data in section 3.3.

3.2.2 Spectral Clustering

To implement the first segmentation method, *spectral clustering*, we rephrase the data clustering problem as a graph-cut-minimization problem of the similarity matrix W. A more detailed survey of the theory can be found in [Lux07]. Here we state only the results necessary to implement the algorithm.

Given a partition of the graph nodes V into subsets $A_1, A_2, \ldots, A_m$, we define the *normalized graph cut*, abbreviated as the NCut.

$$\text{NCut}(A_1, \ldots, A_m) = \frac{1}{2} \sum_{i=1}^{m} \frac{W(A_i, A_i^c)}{vol(A_i)}. \tag{3.6}$$

Where

$$W(A, B) = \sum_{i \in A, j \in B} w_{ij}, \tag{3.7}$$

$$vol(A) = \sum_{i \in A, j \in A} w_{ij} = W(A, A). \tag{3.8}$$

Heuristically, minimizing the NCut serves to minimize the connection between distinct A_i, A_j, while still ensuring that each set is of a reasonable size. Without the $vol(A_i)$ term, the optimal solution often contains one large set and $m - 1$ small sets.

Solving the graph min-cut problem is equivalent to finding an $N \times m$ indicator matrix h, where

$$h_{ij} = \begin{cases} 1 & \text{if } x_i \in A_j \\ 0 & \text{else} \end{cases}. \tag{3.9}$$

Here the columns of h correspond to the m different classes. Each row of h will contain a single 1, which represents the class given to that data point. It has been shown in [GH94] that explicitly solving this problem is an $O(|V|^{m^2})$ process. As this is infeasible in most cases, we instead introduce an approximation of the graph min-cut problem that we will

solve using the symmetric graph Laplacian (as presented in 2.2). The main tool here is the following fact (proven in [Lux07]).

Theorem 3.2.1. *For a given graph-cut $A_1, \dots, A_m$, define h as above, then*

$$NCut(A_1, \dots, A_m) = Tr\left(h^T L_s h\right). \tag{3.10}$$

As explained above, it is infeasible to find the h that minimizes the NCut. However, one may note the similarity between this energy minimization problem and the Laplacian embedding problem presented in (2.10). Up to some choices in scaling, the NCut problem can be thought of as a discretization of the constraints on the embedding problem. With this in mind, we make the choice to relax the problem to arbitrary orthogonal matrices. That is, we find

$$\text{argmin}_{Y \in \mathbb{R}^{n \times m}} \text{Tr}\left(Y^T L_s Y\right) \quad \text{where } Y^T Y = I. \tag{3.11}$$

Similar to the Laplacian embedding in section 2.2.1, this problem is solved by choosing Y to be the matrix containing the m eigenvectors of L_s corresponding to the m smallest eigenvalues. Thus our solution to the relaxed NCut problem is an embedding $V \to \mathbb{R}^m$. To discretize this to a solution for the original min-cut problem, we then implement some classification algorithm on the embedded images of the graph nodes. Specifically, for spectral clustering we use k-means on the eigenvectors Y to create our final classification into m classes h. Although k-means is unlikely to give an optimal classification, it is quite easy to implement, and the final results are strong enough to give a proof-of-concept.

3.2.3 Semisupervised Graph MBO

In this section we describe how to use eigenvectors of the graph Laplacian to segment data in a semisupervised setting. By "semisupervised", we mean that the final classification of a small amount of data points (roughly 5% of all data) is used as an input to the algorithm. Following the example set in [GMB14, MKB13, MGB14], we formulate the problem as a minimization of the Ginzburg-Landau functional.

For the definition of the energy function, we use an $N \times m$ assignment matrix h, similar to the h in (3.9). As before, the final output of the algorithm will be a matrix h where each value is either 0 or 1, with a single 1 in each row. However, for intermediate steps of the algorithm h will be real-valued. Heuristically, the value h_{ij} represents the strength of association between element x_i and class j. For notational convenience we let h_i represent the i-th row of h. With this notation, we define the energy function

$$\begin{aligned} E(u) &= \epsilon \cdot \operatorname{Tr}\left(h^T L_s h\right) \\ &\quad + \frac{1}{\epsilon} \sum_i W(h_i) \\ &\quad + \sum_i \frac{\mu}{2} \chi(x_i) \left\| h_i - \hat{h}_i \right\|_{L_2}^2 . \end{aligned} \tag{3.12}$$

The first term of (3.12) is Dirichlet Energy, similar to 3.2.2. The second term is the multiwell potential

$$W(u_i) = \prod_{k=1}^{m} \frac{1}{4} \left\| u_i - e_k \right\|_{L_1}^2 , \tag{3.13}$$

where e_k is the k-th standard basis vector. These two terms together produce an approximation of the classical real Ginzburg-Landau functional, and it has been shown in [GB12] that they converge to the (graph) total-variation norm

$$TV(h) = \sum_{i,j} w_{ij} \left| h_i - h_j \right| \tag{3.14}$$

as $\epsilon \to 0$. The last term includes the fidelity, where $\hat{h}$ represents the semisupervised input,

$$\chi(x_i) = \begin{cases} 1 & \text{if } x_i \text{ is part of fidelity input} \\ 0 & \text{else} \end{cases} , \tag{3.15}$$

and μ is a tuning parameter.

The gradient descent update associated to this energy is given by

$$\frac{\partial h}{\partial t} = -\epsilon L_s h - \frac{1}{\varepsilon} W'(h) - \mu \chi(x)(h - \hat{h}). \tag{3.16}$$

Similar to [MKB13, GMB14, MMK17], we propose to minimize this via an MBO algorithm. If h^n represents the n-th iterate, then to calculate h^{n+1} we first diffuse

$$\begin{aligned}\frac{h^{n+\frac{1}{2}} - h^n}{dt} = &- L_s h^{n+\frac{1}{2}} - \mu(h^{n+\frac{1}{2}} - \hat{h}) \\ &+ (1 - \chi(x))(h^n - \hat{h}).\end{aligned} \tag{3.17}$$

Then threshold each row

$$h_i^{n+1} = e_r \quad \text{where } r = \text{argmax}_j h_{ij}^{n+\frac{1}{2}}. \tag{3.18}$$

This method effectively splits the energy into two parts and minimizes each alternatively. The diffusion step (3.17) handles the semisupervised Dirichlet Energy (terms 1 and 3 in (3.12)), and the thresholding minimizes the potential function W (term 2 in (3.12)). Note that for the diffusion equation (3.17) we use an implicit method to guarantee stability, as can be more clearly seen after changing coordinates in equation (3.22). The stopping criterion for this algorithm is based on the difference between two consecutive iterates h^n, h^{n+1}. In section 3.3, we stop the algorithm when h^n and h^{n+1} agree on 99.99% of data points.

The diffusion calculation can be done very efficiently by using the eigendecomposition of L_s (the feature vectors described in 3.2.2). If we write

$$L_s = U \Lambda U^T \tag{3.19}$$

and change coordinates

$$h^n = U a^n \tag{3.20}$$

$$\chi(x)(h^n - \hat{h}) = U d^n \tag{3.21}$$

then the diffusion step reduces to solving for coefficients

$$a_k^{n+1} = \frac{(1 + \mu dt) a_k^n - \mu dt \cdot d_k^n}{1 + \mu dt + dt \lambda_k}. \tag{3.22}$$

where λ_k is the k-th eigenvalue of L_s, in ascending order. Note that because we have $\lambda_k \geq 0$ for all k, this update step is guaranteed to be stable, regardless of the choice of parameters μ, dt.

As already discussed in section 2.3, only a small number of leading eigenvectors and eigenvalues need to be calculated in order to achieve a good accuracy. Therefore, in the eigendecomposition (3.19), we choose a number $n \ll N$ of eigenvectors to use, and truncate U to a rectangular matrix. By using the Nyström extension we are then able to calculate an approximate U with significantly reduced computation time.

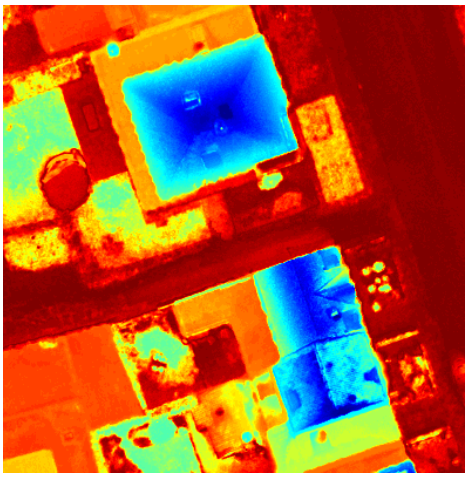
3.3 Experiment

3.3.1 Data Fusion Challenge 2015 Images

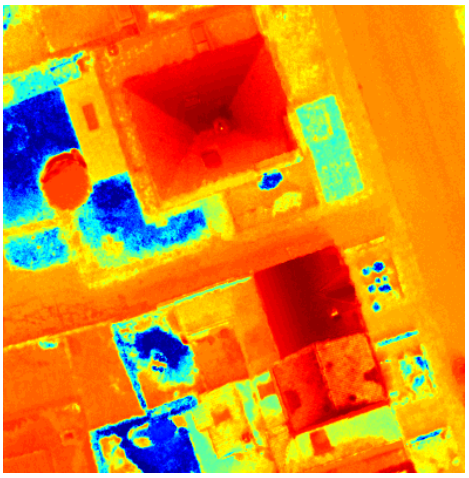
As a first test for the algorithm, recall the DFC2015 data presented in figure 1.1. This set consists of remote sensing images in both the optical and lidar modalities, and is interesting because of the unique information brought by each source.

In figures 3.2a, 3.2b, 3.2c, we show three example eigenvectors of the graph Laplacian. As explained in 3.2.2, these vectors can be thought of as feature of the dataset, and looking at them will give us a rough idea of the final segmentation. Notice how in 3.2a the dark-gray asphalt is distinct from both the nearby grass (which is at the same elevation), and the roofs of the buildings (which are a similar color). This shows at the feature level that the algorithm is successfully using both the optical and the lidar data when determining what pixels can be considered similar. Based on this example vector, the classification algorithm then separates those regions in the final results. One can note the similarities between each of the example eigenvectors and the final classifications 3.2f, 3.2e.

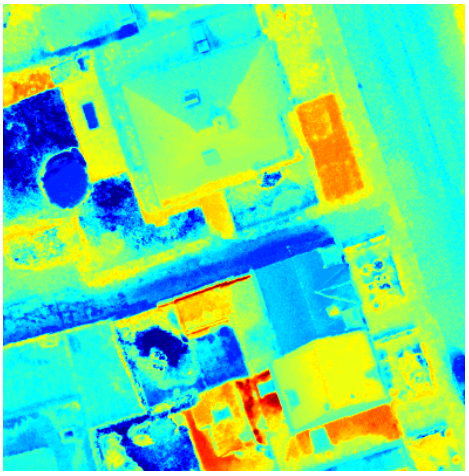
For this image, we choose to segment the data into 6 classes. As the data does not come with any ground truth attached, the number 6 was chosen based purely on personal opinion. The classes given in the semisupervised term (fig 3.2d) are roughly: tall buildings, mid-level buildings, asphalt (bright), asphalt (dark), white tiles, and grass. The exact choice of fidelity pixels was made by either manually choosing locations, or by characteristics of the data (ex: the 1% of pixels at highest elevation). Most importantly, these classes can all be separated using either color or lidar (or both).



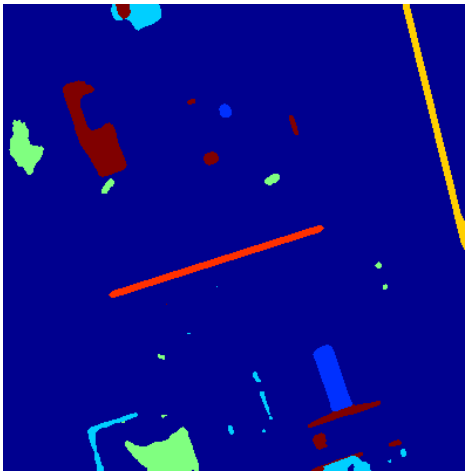
(a) Example eigenvector 1



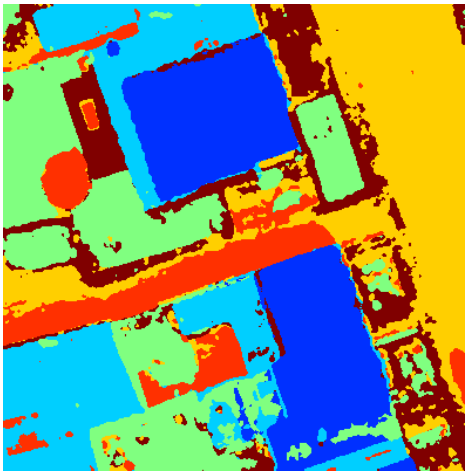
(b) Example eigenvector 2



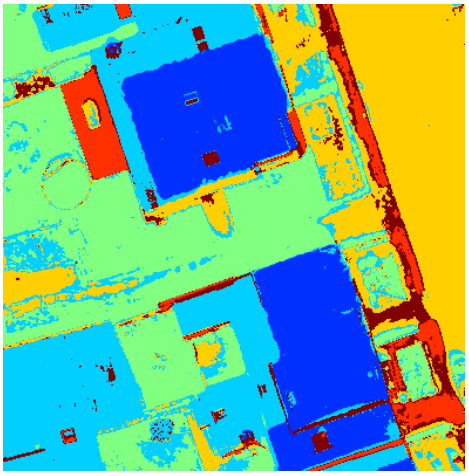
(c) Example eigenvector 3



(d) Semisupervised Input



(e) MBO segmentation



(f) Spectral clustering segmentation

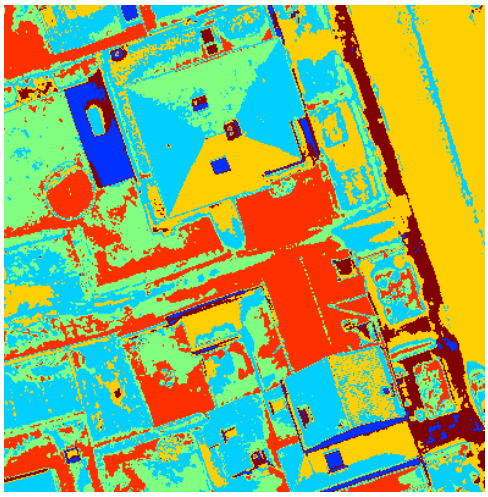
Figure 3.2: DFC2015 features and segmentations

As should be expected, the spectral clustering method (fig 3.2f) does not select exactly the same 6 classes that we have manually identified. As this algorithm is unsupervised, there is no way of encoding our human preference into the method. Therefore, the choice of exactly how to divide the different groups of pixels is made in accordance with only the graph min cut energy. In the end, this algorithm can still pick out the major features of the dataset, but the specific decisions of exactly which classes to combine and which to separate does not agree with our human intuition. By instead using a semisupervised algorithm such as graph MBO (fig 3.2e), we can input a small amount of information (in this case, 7% of total pixels) in order to align the energy minimization with our human expectations. Therefore, the final result aligns quite well with initial expectations.

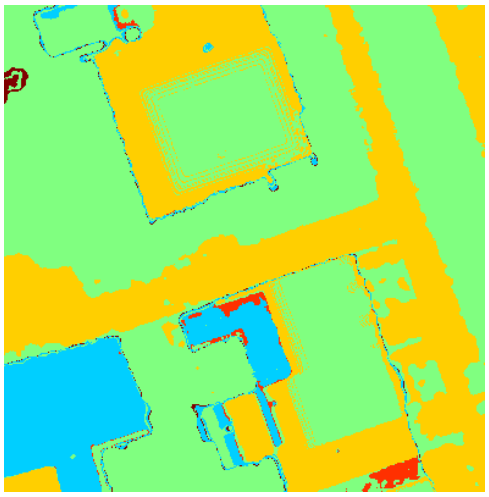
When choosing the exact parameters for the algorithm, there are two factors to consider. The choice of dt is a tradeoff between the runtime of the program and theaccuracy of the final result, and the choice of μ dictates our level of confidence in the semisupervised input (fig 3.2d). For this particular example, we choose $dt = 0.1$ and $\mu = 10^3$.

For comparison, we also show the results of several more naive algorithm. To emphasize the importance of using both modalities in the classification, we show in figure 3.3a the result of spectral clustering on the RGB data alone, and in figure 3.3b the similar result for the lidar data. As discussed earlier, neither modality contains sufficient information to separate the chosen classes alone, and this can be clearly seen in the results.

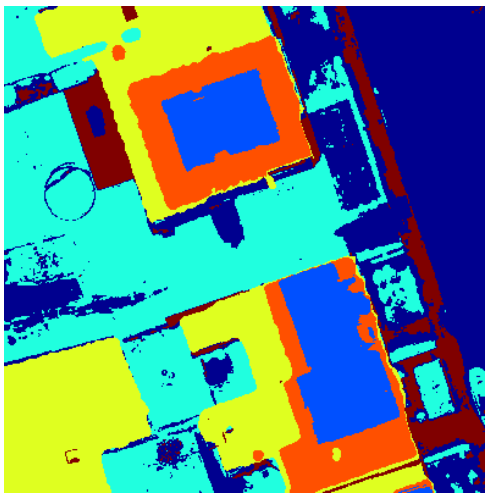
In figures 3.3c, 3.3d we display the result of two multimodal methods more naive than the one presented here. Figure 3.3c shows the result of k-means applied directly to the concatenated (4-dimensional) dataset, without any preprocessing. As can be seen from the result, a direct application of k-means is not well suited towards handling information from disparate sources. In this particular example, the segmentation overvalues the information from the lidar modality, and therefore overclassifies the buildings based on height. This in turn results in a poor classification of the different ground-level features, as the RGB information is not well-used. For figure 3.3d, we extract graph Laplacian eigenfunctions from each modality separately, then concatenate these eigenfunctions for the k-means step. Similar to our example with k-means, this method is insufficient to properly combine the



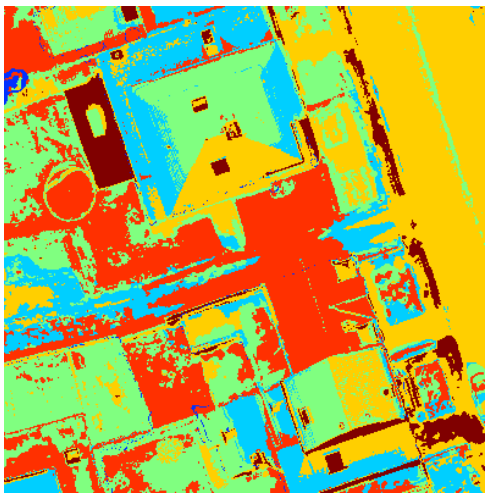
(a) RGB data only segmentation



(b) Lidar data only segmentation



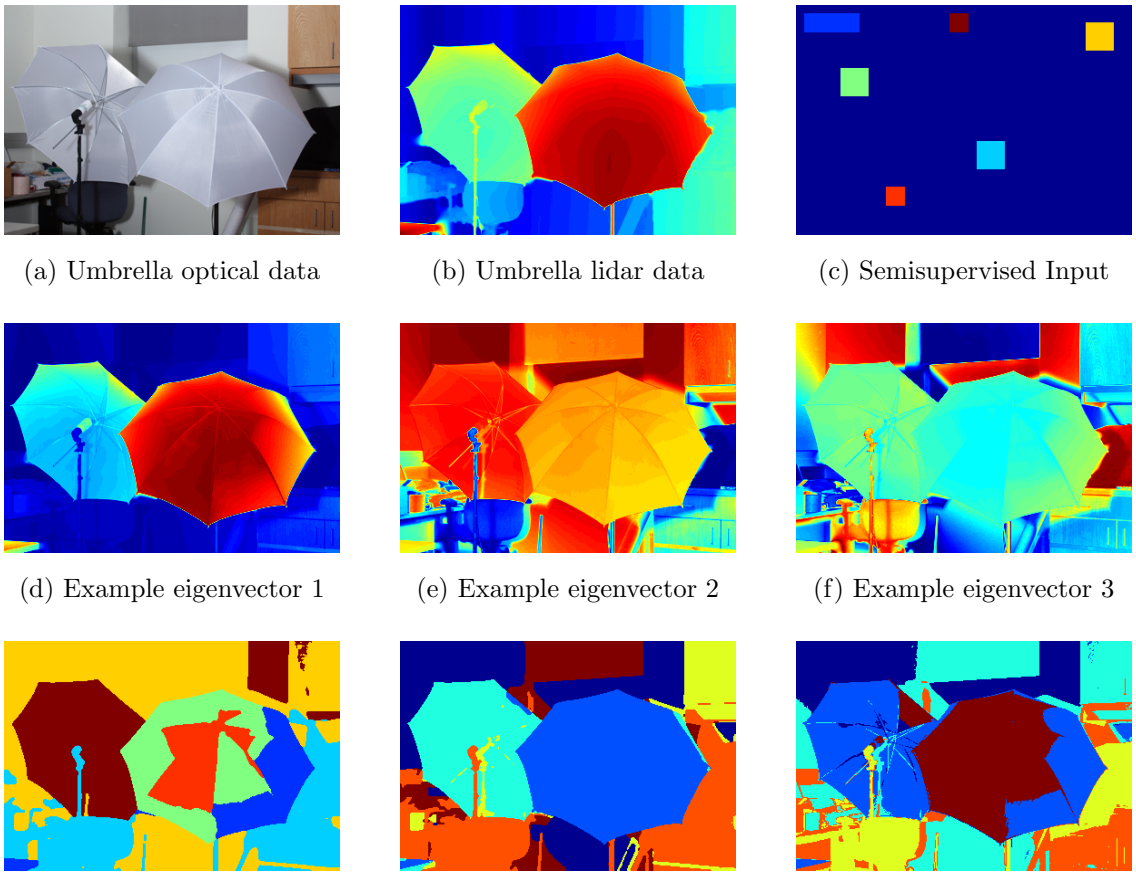
(c) Direct k-means (no preprocessing)



(d) Calculate eigenvectors separately

Figure 3.3: Other methods on DFC2015 for comparison. (a) Spectral clustering on RGB data, (b) Spectral clustering on lidar data, (c) k-means on concatenated (4-dimensional) dataset, (d) Calculate eigenvectors on RGB and lidar separately, and k-means on the concatenation.

RGB and lidar information. Unlike our max-norm method, concatenating the individual eigenvectors makes no effort to find which dataset is more relevant in separating the various objects. Therefore, while the dataset input to k-means technically has sufficient information to classify the data as desired, there is also enough superfluous information to result in a



(a) Umbrella optical data (b) Umbrella lidar data (c) Semisupervised Input

(d) Example eigenvector 1 (e) Example eigenvector 2 (f) Example eigenvector 3

(g) Spectral clustering segmentation (h) MBO segmentation (i) Direct k-means

Figure 3.4: Umbrella data results

poor classification.

3.3.2 Umbrella Data

In fig 3.4 we show the results of the method applied to another optical/lidar set (found in [SHK14]), which we will refer to as the umbrella data. Similar to the DFC2015 set, the umbrella data serves as a good example because it cannot be easily analyzed using one modality alone. The umbrellas and the background walls are nearly the same shade of white, and can only be distinguished in the lidar data. Meanwhile, the different pieces of the background all lie at nearly the same depth, and can only be separated by color. As was the

case with the DFC2015 data, the final classifications 3.4g, 3.4h can be understood by looking at the individual feature vectors. In figure 3.4d, we see very clearly the difference the major features of the dataset: the front umbrella, the back umbrella, and the background wall. Figures 3.4e, 3.4f show more of the small details of the data, separating the many different background objects.

As was the case with the DFC2015 data, we chose to segment this image into 6 classes based primarily on personal opinion. The classes represented in the semisupervised input are: the front umbrella, the back umbrella, the wooden cabinet in the corner, and various different colors of background objects (fig 3.4c). Similar to the results from the DFC2015 dataset, we can find many major features in the spectral clustering result (fig 3.4g), but the exact details of the classification do not match our expectations. In particular, the foremost umbrella of this set is overclassified, which in turn forces the algorithm to combine the background objects into a small number of classes. In the graph MBO result (fig 3.4h), we give include the class of 5% of pixels as part of the input, and as such the classification fits the original data much more closely.

In figure 3.4i we again show the result of applying k-means directly to the concatenated dataset. As seen before, this naive algorithm struggles to make use of all the information present in the different modalities. In this example, the issue can be seen most clearly in the failure to separate the two umbrellas. k-means succeeds in separating many objects based on their RGB value, but the fact that the two umbrellas are grouped into the same class shows that the lidar information is not properly valued.

3.3.3 Data Fusion Challenge 2018 Images

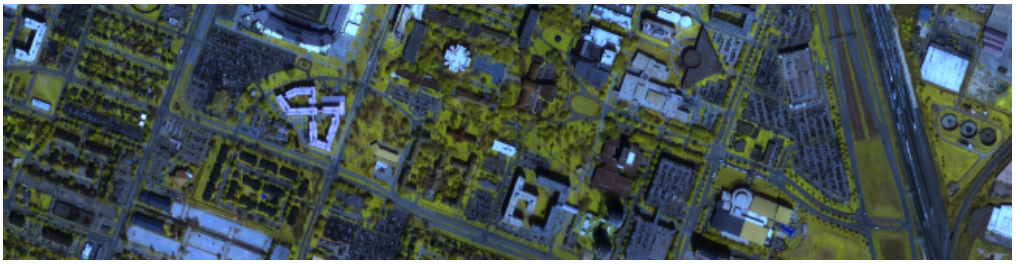
For a more in-depth test of the algorithm, we look at the images from the Data Fusion Challenge 2018 (abbreviated DFC2018). A much more robust dataset, the DFC2018 consists of 50 bands of hyperspectral data covering wavelengths 380-1050nm, Multispectral-LiDAR data at 3 different wavelengths (1550nm, 1064nm, and 532nm), intensity rasters for each LiDAR band, a Digital Surface Model (DSM) of the area, and finally, ground truth data for

roughly 30% of input pixels. The data covers an area of size $600m \times 2400m$ at a resolution of 0.25^m per pixel, for a total of nearly 6,000,000 pixels.

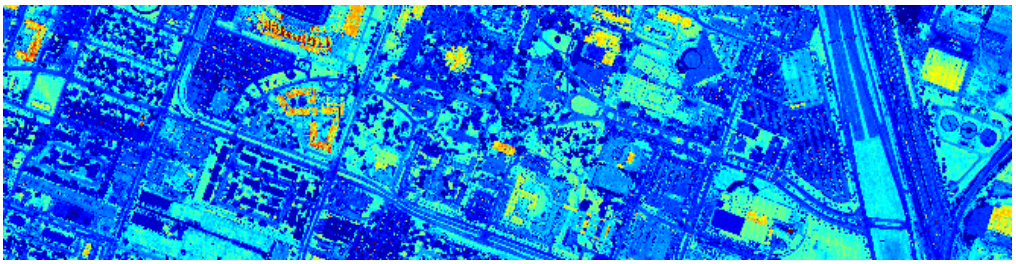
Featured in this dataset are a wide variety of different materials. The accompanying ground truth labels 20 different classes, including different plants, water, various types of pavement, and some metal objects. We merge these 20 original classes into 10 final classes (see table 3.6), as ground truth data separates some objects with remarkably comparable spectral signatures (for example, roads, major throughfares, and highways are considered different classes in the original input). Still, amongst the 10 final classes we are given the opportunity to show the strength of our algorithm in considering the entire input set while differentiating between classes, as there is no one modality that fully separates all 10 classes.

As explained in 2.3, one difficulty in dealing with such a diverse dataset is the efficacy of the Nyström eigenvectors. Recall that as part of the MBO update step (3.22) we perform a coordinate change using the Graph Laplacian eigenvectors U (3.20). As we only calculate (an approximation of) the most influential eigenvectors, rather than the entire matrix, this coordinate change $a^n = U^T h^n$ represents a loss of overall information. Essentially, we are projecting the full data onto the space spanned by the chosen eigenvectors. Therefore, it is important that each class is well represented in our choice of Nyström landmark nodes. To accomplish this, we use the ground truth labels given to select a few nodes from each class. We also increase the total number of Nyström eigenvectors from 100 to 200.

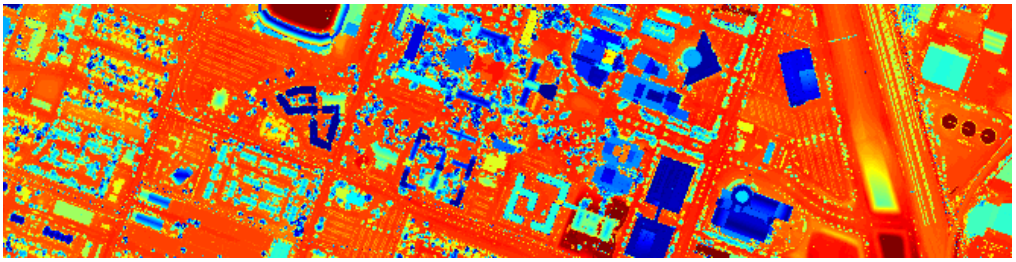
The application of our algorithm to the dataset is also complicated by the size of the image. This particular example is more than an order of magnitude bigger than the others, and the space complexity of building the graph becomes a big issue, even when using the Nyström method (section 2.3) to reduce the matrix size. We handle ths computation by running the algorithm many times over patches of size 100,000 pixels. To provide the semisupervised input for the graph MBO, we choose $\approx 0.1\%$ of the ground truth data (with an equal distribution between class labels) and add a copy of it to each run of the algorithm. For the input parameters of the MBO, we use $dt = 0.1$ and $\mu = 10^3$. Using the ground truth data, we can get a quantitative evaluation of the quality of our algorithm. For the pixels that are labeled in the ground truth, we correctly classify 80%.



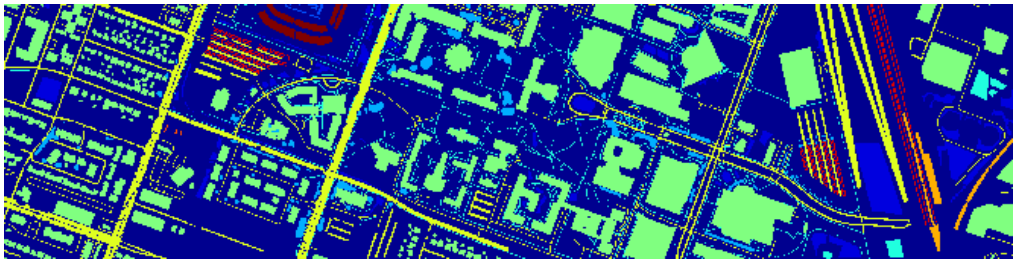
(a) Hyperspectral data (RGB bands)



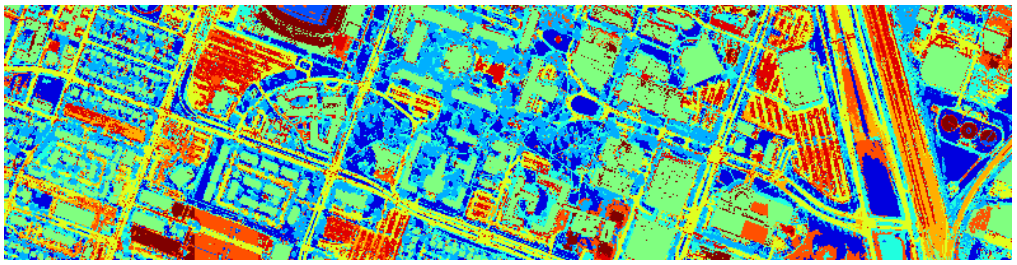
(b) 1064nm intensity raster



(c) Digital Surface Model



(d) Ground Truth



(e) MBO segmentation (using $\approx 0.1\%$ of ground truth as semisupervised input)

Figure 3.5: DFC2018 Data

Label	Name	Validation Set Size	Graph MBO Accuracy (%)
	Grass	169204	88.23
	Artificial turf	2736	100.00
	Trees	74494	97.61
	Sidewalks and bare earth	154099	71.19
	Buildings	1053764	83.76
	Roads and other paved areas	460150	65.45
	Railways	27748	96.33
	Unpaved parking lots	587	99.15
	Cars and trains	47768	89.66
	Stadium seats	27296	96.19

Figure 3.6: DFC2018 Classes and MBO Accuracy

3.3.4 Data Fusion Challenge 2013 Images

For a more quantitative evaluation of our method, we apply our multimodal graph MBO algorithm to the images from the Data Fusion Challenge 2013 [DMH14], and draw a comparison to the results of [LPB15]. The datasets distributed for the contest include a hyperspectral image and a lidar-derived digital surface model (DSM) of the University of Houston campus and its neighboring area, both at a spatial resolution of 2.5m. The hyperspectral image consists of 144 bands in the 380-1050 nm range (figure 3.7a), and corresponding co-registered lidar data represents the elevation in meters above sea level (figure 3.7b).

Along with the the input data, training and validation sets were created by the Data Fusion Technical Committee, labeling 2832 pixels for the training set, and 15029 pixels for the testing set (figures 3.7c, 3.7d, respectively). The pixels were separated into 15 classes, as is detailed in table 3.1. As shown, both land cover and land use classes were considered, including natural objects (e.g., grass, tree, soil, and water), and man-made objects (e.g., road, highway, and railway). Note that the Parking Lot 1 class included parking garages at ground level and in elevated areas, and Parking Lot 2 corresponded to parked vehicles.

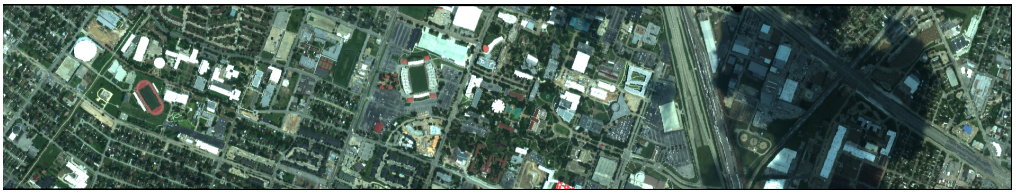
For each class, the size of training and validation sets was made constant (when possible) to include about 200 and 1000 samples, respectively. It is noteworthy that a large cloud shadow was present during the acquisition of the HSI; as a result, no training samples were selected in this region. However, a significant number of validation samples were collected to test the efficacy of various algorithms in dealing with cloud shadow.

We show in figure 3.7e the results of our multimodal graph MBO algorithm on this data, with per-class accuracies and a comparison against [LPB15] given in table 3.1. Although our algorithm generally has good results, we specifically struggle to classify a piece of highway near the right-hand side of the image. By comparing the hyperspectral and lidar data, one can see that this section of road is actually a bridge over railway below, and it is for this reason that our algorithm gives poor results in this area. To maintain generality and applicability to situations other than image processing, we do not use positional data in our algorithm. Therefore each of the higher-elevation pixels at the top of the bridge are treated as separate objects, and misclassified as buildings.

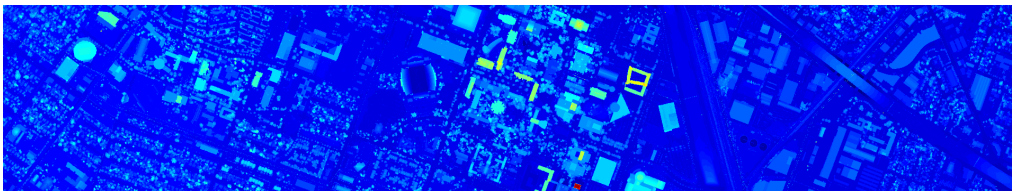
3.3.5 Jade Plant Data

Found in the same paper as the umbrella data [SHK14], we test the method against another optical/lidar scene of a jade plant, shown in figures 3.8a, 3.8b. As with all examples shown here, there is a large amount of non-redundancy between the two input images. In particular, in this example the optical image is quite homogeneous, as it is mostly composed of shades of brown. Therefore, one would expect the addition of the lidar data to greatly aid the segmentation.

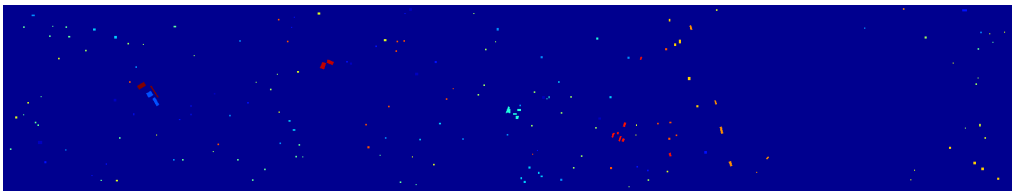
In figures 3.8c, 3.8d, we once again show a few example eigenvectors extracted from the input data. As before, we can see in each eigenvector some pieces of the final classifications 3.8e, 3.8f.



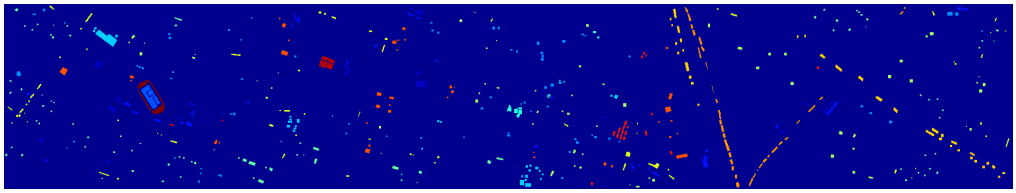
(a) Hyperspectral data (RGB bands)



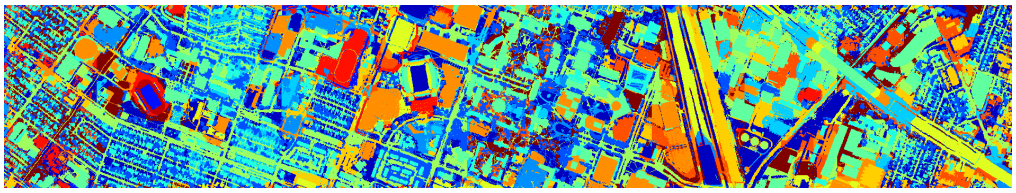
(b) Lidar data



(c) Training Samples



(d) Validation Samples



(e) MBO segmentation

Figure 3.7: DFC2013 Data

3.4 Choice of Norm to Combine Modalities

Recall from section 3.2.1, we create our multimodal edge weights by calculating the distance between nodes in each modality, applying the appropriate scaling, and choosing the largest distance found, and applying a radial basis function to this distance (equations (3.3) and

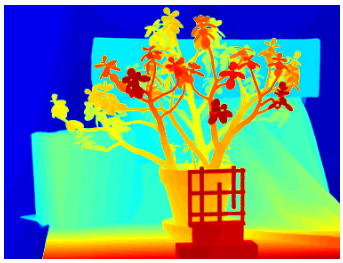
Label	Name	GGF [LPB15]	Our method
	Healthy Grass	82.91	80.74
	Stressed Grass	99.34	87.16
	Synthetic Grass	100.00	100.00
	Trees	99.34	98.95
	Soil	100.00	99.03
	Water	95.10	98.15
	Residential	90.86	90.14
	Commercial	95.63	92.68
	Roads	89.33	74.84
	Highway	92.76	69.68
	Railway	96.58	90.45
	Parking 1	91.93	85.81
	Parking 2	74.39	81.02
	Tennis Court	100.00	99.77
	Running Track	98.73	100.00
	Overall Accuracy (%)	94.00	88.56
	Average Accuracy (%)	93.79	89.90
	κ	0.935	0.877

Table 3.1: Results of our method and of [LPB15] on DFC2013 data

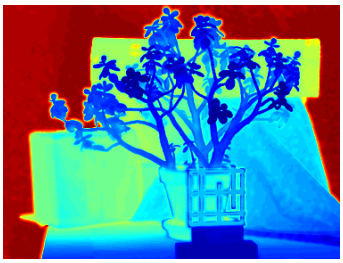
(3.4)). In this section we concern ourself specifically with the choice of function for combining the distances found in each modality. As the distance functions dist_ℓ, $1 \leq \ell \leq k$ are a part of the input to our algorithm, we then naturally have for each pair (x_i, x_j) k different distance values, one for each modality. Let

$$d_{ij} = \left(\frac{\text{dist}_1(x_i^1, x_j^1)}{\lambda_1}, \ldots, \frac{\text{dist}(x_i^k, x_j^k)}{\lambda_k} \right). \tag{3.23}$$

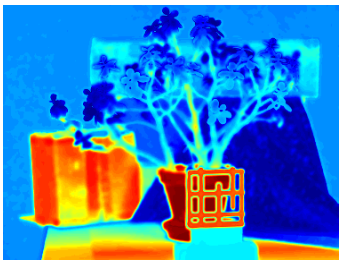
(a) Jade plant optical data



(b) Jade plant lidar data

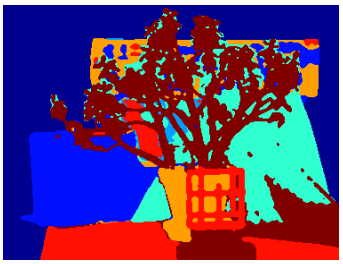


(c) Example eigenvector 1



(d) Example eigenvector 2

(e) Spectral clustering segmentation



(f) MBO segmentation

Figure 3.8: Jade plant data results

Note that choosing the maximum between the k different values is the same as applying the L^∞ norm in $\mathbb{R}^k$. In other words, we can change the notation of equation (3.3)

$$\text{dist}(x_i, x_j) = \|d_{ij}\|_\infty . \tag{3.24}$$

We explain in section 3.2.1 our reasoning for choosing this particular method of combining individual modality distances, but it is worth some investigation into other possible methods. One very natural generalization of the distance function chosen is to replace $\|\cdot\|_\infty$ with a different choice of L^p norm. That is, we could alter equation (3.3) and define

$$w_{ij} = \exp\left(-\|d_{ij}\|_p\right) \tag{3.25}$$

$$= \exp\left(-\left(\sum_\ell \left(\frac{\text{dist}_\ell(x_i, x_j)}{\sigma_\ell}\right)^p\right)^{1/p}\right), \tag{3.26}$$

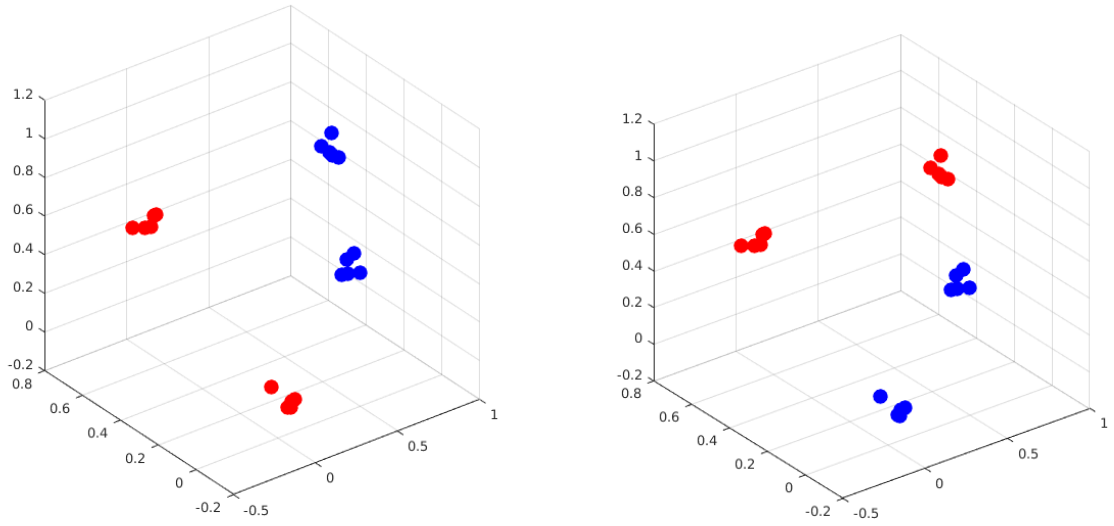
for some choice of $1 \leq p \leq \infty$. The question then arises, if we choose a different L^p norm, what difference should we expect in the final result? In this section we aim to answer this question, both on the level of heuristics, as well as with some more concrete observations.

The most obvious heuristic is that choosing a large p causes the edge weights to be heavily affected by individual outliers among modalities, whereas choosing a small p will provide more of an averaging effect over all the modalities. In this paper we choose to use the L^∞ norm because we expect each modality to separate some, but not all, of the objects. In other words, we believe that two pixels should be considered similar only if they are similar in every modality, and a single difference shown across all modalities is worth consideration. However, for a different application - for example, when working with noisy data - it may be desirable to a smaller value of p to obtain more of an average over the input data.

These heuristics give an idea of the quality of difference between two choices of norms, but it is also desirable to understand the quantity of difference. If we change our choice of p norm, how much change can we expect in the graph weights, and in the resulting graph cut? Unsurprisingly, this answer is highly dependent on the number of modalities. From classical measure theory in $\mathbb{R}^k$, we have that for $1 \leq p \leq q \leq \infty$

$$\|d_{ij}\|_p \leq \|d_{ij}\|_q \leq k^{1/p-1/q} \|d_{ij}\|_p . \tag{3.27}$$

That is, the difference between L^p norms here depends on the number of input modalities. So when working with relatively few modalities, a different choice of p norm will not make a large difference in the distance between points (and therefore the graph weights). However, as the number of modalities increases, it is possible to have large differences in the graph weights as a result of changing p. For example, we show in figures 3.9a, 3.9b we show a small synthetic dataset where the optimal graph cut changes drastically based on the norm used to create the graph weights. The dataset consists of 20 points in $3D$, representing 3 different modalities, grouped into 4 clusters of 5 points. In each image we segment the points into two classes based on the minimal graph NCut (calculated without approximations), where in figure 3.9a the graph weights are calculated using the 1-norm, and in 3.9b the weights are calculated using the ∞-norm. As can be seen in the results, it is possible for the grouping of points to change entirely based on the change of distance function.



(a) 1-norm Segmentation

(b) ∞-norm Segmentation

Figure 3.9: Optimal graph NCut of synthetic dataset under two different of norm for combining distances from individual modalities. Data represents input from 3 1-dimension modalities.

3.4.1 Theorem and Proof

In addition to a synthetic example, we also present a theoretical statement on the variability of graph cuts based on the distance function. We begin with some notation to simplify the statement of the theorem. Suppose we have n points $x_1, \ldots, x_N$ in some $\mathbb{R}$-space. We're interested in distances between points in different norms. So let

$$d_{ij}^p = \|x_i - x_j\|_p \tag{3.28}$$

$$D^p = \left\{ d_{ij}^p \,:\, 1 \leq i < j \leq N \right\} \tag{3.29}$$

In other words, for each choice of p there are $\binom{N}{2}$ values that we are interested in. More specifically, we are interested in the ordering under $\leq$ for each of these sets D^p, as the graph weights are depend on the relative size of the different d_{ij}^p rather than on the absolute size. As we show below, if the ambient dimension is large enough it is possible to simultaneously control the orderings in more than one D^p by properly choosing the $x_1, \ldots, x_N$.

Theorem 3.4.1. *For any $1 \leq p < q \leq \infty$, it is possible to choose the $x_1, \ldots, x_N \in \mathbb{R}^N$ to simultaneously produce any arbitrary ordering under $\leq$ on both D^p and D^q.*

Proof. It suffices to give a proof for the case $p = 1, q = \infty$, as the L^p norm on $\mathbb{R}^N$ is a decreasing function of p, and therefore any inequalities in the case $p = 1, q = \infty$ will hold in the general case $1 \leq p < q \leq \infty$ as well.

To construct the $x_1, \ldots, x_N$ that produce the distances we desire, we first begin with the standard basis vectors

$$x_i = e_i \quad 1 \leq i \leq N \tag{3.30}$$

then proceed to make small edits to the x_i to achieve the desired order. Note that before making any changes, $d^1_{ij} = 2, d^\infty_{ij} = 1$ for all i, j. To properly order the L^1 distances, we makes the following type of adjustment where necessary:

$$x_i^{new} = x_i^{old} + \epsilon \cdot e_j \quad \text{for some } \epsilon < 1, \tag{3.31}$$

as pictured in figure 3.10a. This decreases d^1_{ij} by ϵ, increases $d^1_{i\ell}$ by ϵ for $\ell \neq j$, and does not affect $d^\infty_{\ell k}$ for any ℓ, k.

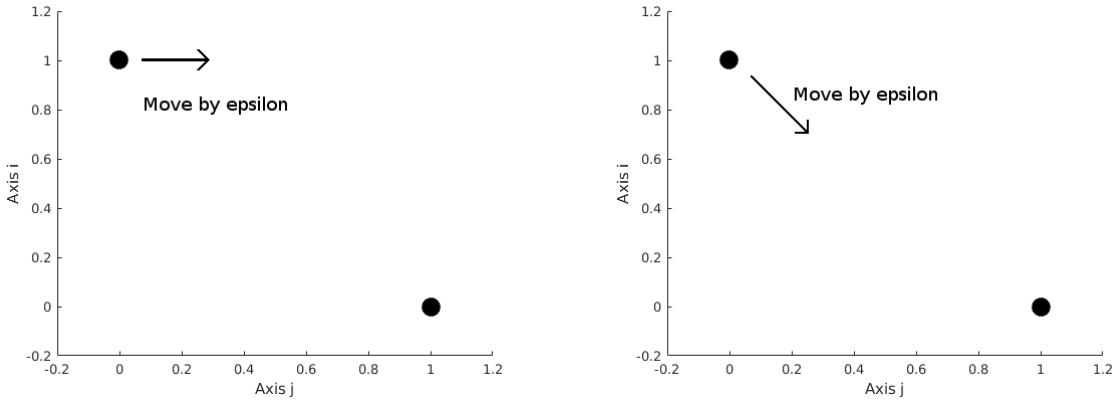
Once the L^1 distances are properly set, we can fix the L^∞ distances as follows:

$$x_i^{new} = x_i^{old} + \epsilon \cdot e_j - \epsilon \cdot e_i \quad \text{for some } \epsilon < 1, \tag{3.32}$$

as pictured in figure 3.10b. This decreases d^1_{ij} by $2 \cdot \epsilon$, decreases d^∞_{ij} by ϵ, and does not affect $d^\infty_{\ell k}$ for any ℓ, k.

Given these two possible moves, it's relatively simple to achieve the desired ordering of distances by using progressively decreasing values of epsilon. At each step one can choose epsilon sufficiently small so that the changes from the previous steps are not affected. For example, if one could use $\epsilon = 2^{-k}$ for the kth move.

□



(a) Move 1

(b) Move 2

Figure 3.10: Moves from 3.4.1

3.5 Summary

In conclusion, graph-based methods provide a straightforward and flexible method of combining information from multiple datasets. By considering the similarity between points in each individual dataset, we reduce the information from each modality into something more directly comparable. This in turn gives us a model that is more data-driven, using the information obtained from each modality without needing to know the details about the source from which the data was captured. Therefore the same algorithm could be applied in many different scenarios, with different types of data.

Once we have calculated and compared the different weight matrices, we can then create the graph Laplacian of the data and extract features in the form of eigenvectors. These features can then be used as part of many different data-segmentation algorithms. For this paper, we use k-means on the eigenvectors as a simple proof-of-concept, and graph MBO as a more in-depth approach. The main computational bottleneck is in calculation of the eigenvectors, for which we have made several approximations to improve the speed of our algorithm. After this step, there are many different viable classifications in the literature.

A future area of interest is to further investigate the possible choices for multimodal graph weights and to create a more solid theoretical background for these choices. When

choosing to use graphs in machine learning problems the foremost concern is the quality of the data representation in graph form, and therefore it is this segment of our algorithm that merits the most study.

CHAPTER 4

Fast Graph Matching

In this chapter we present our algorithms on graph matching. Given two undirected, weighted graphs, we search for a matching between nodes that minimizes the difference between corresponding edges. Although the formal statement of this problem is NP-hard, we use the Laplacian embedding from section 2.2.1 to approximate and quickly minimize the matching energy. Then, using our matching algorithms, we show several possible applications to real-world machine learning problems.

4.1 Related Work

Extensive surveys of graph matching methods can be found in [Ven15,FPV14,CFS04]. These algorithms are commonly categorized as exact or inexact based on the approach used. Exact methods search for a true isomorphism of (sub)graphs, using either heuristics or additional given information to narrow the search space [Ull76, SFS03]. As explained above, the difficulty of the exact matching problem forces these methods to have limited applications, falling well short of the needs of modern pattern recognition or computer vision. For this reason, the majority of research done in recent years is focused on inexact graph matching. Within the category of inexact graph matching there are a wide variety of approaches, such as tree searches [SSA04], or a reduction to bipartite graph matching [BW03]. However, the most common approach is to relax the discrete constraints of the problem to the continuous case in order to recast the problem as a non-convex optimization.

One common relaxation of constraints is to the set of *doubly stochastic matrices*, the convex hull of the set of permutation matrices. In this formulation, each potential match

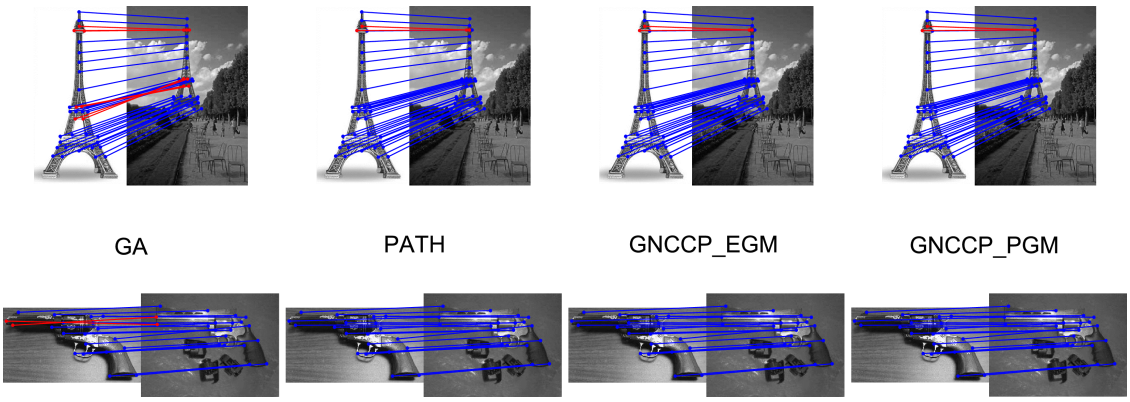


Figure 4.1: [LQ14] A comparison of several matching algorithms.

between graph nodes is given a probability rather than a strict choice of 0 or 1. After the cost function is minimized in this new set, a discretization step is then performed to obtain the final solution. In [GR96] the authors introduce the Graduated Assignment (GA) algorithm, minimizing the first-order Taylor series approximation to the graph cost function by iteratively solving the corresponding linear assignment problem via the softassign algorithm. Both [LQ14] and [VCL15] impliment a similar procedure, using a Franke-Wolfe method to minimize the Taylor approximation, with the Hungarian algorithm and a stardard linesearch for each iteration. This approach is extended to many-to-many graph matching problems in [ZBV10], and to hypergraph matching in [YQL17]. This same continuous relaxation is also used in several fixed point methods, such as [LHS09] where the authors project the partial solution back to the discrete space at each timestep, or [LHL16] where the authors define a partial doubly stochastic projection function that can quickly map the gradient to the desired solution space. Lastly, in [ZBV09] the authors introduce the PATH algorithm, in which the matching is formulated as a convex-concave programming problem which is solved by interpolating between two approximate simpler formulations. In figure 4.1 we reproduce a figure from [LQ14] comparing the matching results of GA ([GR96]), PATH, ([ZBV09]), and GNCCP ([LQ14]).

Another major group of inexact graph matching methods is spectral techniques, which can be thought of as a relaxation of the constraints to the space of orthogonal matrices. This approach was first introduced in [Ume88], where each potential match between nodes

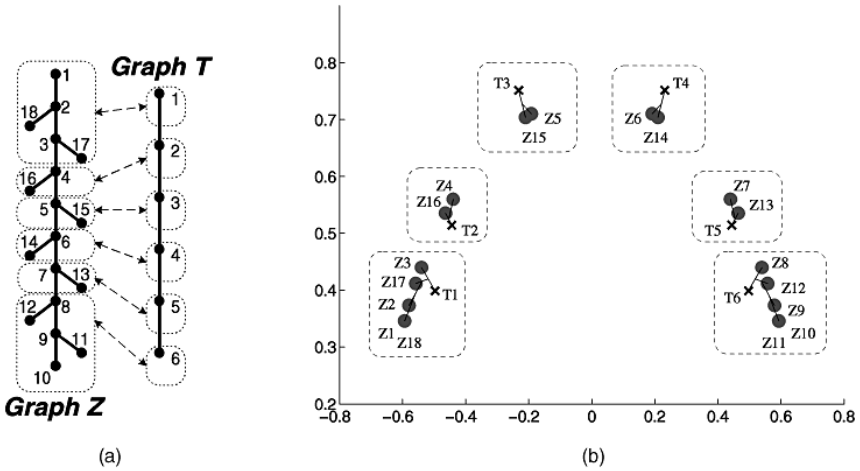


Figure 4.2: [CK04] Two graphs and a comparison of eigenspaces.

is given an affinity score by comparing their positions in each graph eigenspace. The final match can then be obtained by solving a linear assignment problem using the affinity scores. In [KC02, CK04] the authors extend this method to graphs of unequal size, and in [KSM09] the authors introduce an unsupervised method based on histrograms to better align the eigenvectors, ensuring that the eigenspaces are truly comparable. In figure 4.2 we show an example of two graphs and the corresponding eigenspace comparison from [CK04]. In [LH05, CSS07] the authors instead create affinity scores between edges of each graph, avoiding the issue of aligning different eigenspaces but squaring the size of the problem.

4.2 The Method

In this paper we approach the graph matching problem via spectral methods, similar to [Ume88, KSM09], but with considerable changes to the matching process to reduce both the time and space complexity of the problem. In this section we formally introduce the matching problem in 4.2.1, the spectral approach to a solution in 4.2.2, and our specific contributions in 4.2.3, 4.2.4, and 2.3. Finally, we give a theoretical computation of our algorithm runtime in 4.2.5. Short overviews of our HOSM and HMSM algorithms can be found in Algorithms 1, 2, respectively.

4.2.1 The Weighted Graph Matching Problem (WGMP)

Let $G_1 = (X_1, W_1)$, $G_2 = (X_1, W_2)$ be undirected, weighted graphs. Here X_1, X_2 represent the nodes of G_1, G_2 (respectively), and W_1, W_2 are the corresponding matrices of edge weights. Let $|X_1| = M$, and $|X_2| = N$. Without loss of generality we will assume $M \leq N$. The goal of the WGMP is to find an injection $\rho : X_1 \to X_2$ that minimizes the squared difference of edge weights:

$$\text{argmin}_\rho \sum_{i=1}^{M} \sum_{j=1}^{M} \left(W_1(i,j) - W_2(\rho(i), \rho(j))\right)^2 . \tag{4.1}$$

It is often easier to view the map ρ as an injection of the indices $\{1, 2, \ldots, M\} \to \{1, 2, \ldots, N\}$. Corresponding to this, there is an $N \times M$ permutation matrix P where $Pe_i = e_{\rho(i)}$ for each $1 \leq i \leq M$ (here e_i is the ith standard basis vector). With this notation we can rephrase the minimization problem as

$$\text{argmin}_P \left\| W_1 - P^T W_2 P \right\|_F^2 . \tag{4.2}$$

Finding an exact solution to this problem is NP-Hard [AKK12]. Instead, we look for an approximate solution via the methods presented in [Ume88, KSM09]. In particular, we use a form of the Laplacian embedding presented in section 2.2.1 to solve a relaxed formulation of the matching problem.

4.2.2 The relaxed WGMP and the graph Laplacian

As the WGMP is too difficult to solve exactly, we instead introduce a relaxed version of the problem which is much more reasonable. Specifically, instead of choosing P a permutation matrix as in 4.2, we look for an orthogonal matrix

$$Q^* = \text{argmin}_{QQ^T = I} \left\| W_1 - Q^T W_2 Q \right\|_F^2 . \tag{4.3}$$

This problem was solved theoretically in [Ume88] using the Laplacian embedding. For each graph G_ℓ ($\ell \in \{1, 2\}$) notate the graph Laplacian

$$L_\ell = D_\ell - W_\ell, \tag{4.4}$$

and let $L_\ell = U_\ell \Lambda_\ell U_\ell^t$, be the eigendecomposition of the Laplacian, where the eigenvalues Λ_ℓ are monotone increasing along the diagonal. Then the spectral graph matching theorem from [Ume88] states that if each L_1, L_2 has distinct eigenvalues, the optimal Q from (4.3) is given by

$$Q^* = U_2 S U_1^T. \tag{4.5}$$

Here S is a diagonal matrix with values ± 1 to account for the sign ambiguity in eigenvectors between U_2 and U_1 (explained in detail in 4.2.3). This equation is also a slight abuse of notation, as the matrix dimensions do not properly correspond in the case where $|X_1| \neq |X_2|$. In fact, similar to chapter 3, for practical purposes we never calculate the full eigenvector matrices U_1, U_2. Instead, we choose a number $n \leq \min(|X_1|, |X_2|)$ and truncate both U_1 and U_2 to the n columns corresponding to the n smallest eigenvalues. In terms of the energy minimization, this approximation does not significantly change the quality of solution, and as explained in section 2.3 this allows us to vastly improve both the time and space complexity of our algorithm.

Note that this solution is easy to describe in the context of the Laplacian embedding (section 2.2.1. If we temporarily assume that $S = I$ in equation (4.5), then our optimal $Q^* = U_2 U_1^T$. From the viewpoint of our embeddings

$$\phi_1 : G_1 \to \mathbb{R}^n, \phi_2 : G_2 \to R^n, \tag{4.6}$$

this product $U_2 U_1^T$ compares the images of nodes in X_1, X_2 via the standard dot product in $\mathbb{R}^n$. Instead of an exact 0 or 1 representing exact matches, this gives a "similarity score" between each pair of nodes in $X_1 \times X_2$. In section 4.2.4, we explain how we convert this relaxed solution into an exact matching between graphs.

4.2.3 Eigenvector Alignment

To properly compare the Laplacian embeddings ϕ_1, ϕ_2 in (4.6), some work is required to ensure that the different eigenvectors are properly aligned. Because of the inherent sign ambiguity in the eigenvectors U_1, U_2, there are in fact 2^n possibilities for each of the Laplacian

embeddings ϕ_1, ϕ_2. Furthermore, because of numerical instability and small perturbations between datasets, the ordering of eigenvalues based on $\leq$ may not provide the optimal match between eigenvectors. Therefore, to ensure a reasonable comparison between graphs we must take steps to align our eigenvector matrices. For this reason, we introduce an extra term to the theoretical solution to the relaxed WGMP (4.5) and include a possible permutation of columns for one eigenvector matrix.

$$Q^* = U_2 S P U_1^T. \tag{4.7}$$

Here the matrix S accounts for the possible sign difference between eigenvectors, and P accounts for any permutation of columns of U_1^T.

This problem of finding an optimal S, P has already been approached from multiple angles. In [Ume88] the authors omit S entirely by replacing each U_ℓ ($\ell \in \{1, 2\}$)) with $|U_\ell|$ in equation 4.5. This allows for a quick handling of issue, but also ignores a great deal of information by removing signs, and doesn't address the problem of P at all. In [KSM09] the authors consider the histograms of each eigenvector in U_1, U_2, as these are invariant under the ordering of graph nodes in X_1, X_2. Each pair of histograms is compared, both with and without a sign flip, and an optimal matching is chosen, thus deciding both S and P. The main advantage of this method is that it is completely unsupervised. Additionally, as the histograms can be constructed with any number of bins, it allows comparisons even in the case where $|X_1| \neq |X_2|$.

Here we propose a semi-supervised method of aligning eigenvectors. Given a small number of known matches, we can concretely compare different alignments and choose the optimal one. Suppose we are given J known matches between X_1 and X_2. For simplicity, we reindex the graphs so that the matched nodes are the first J in each set. Let $\hat{U}_1, \hat{U}_2$ be the $J \times n$ matrices obtained by truncating U_1, U_2, respectively. The rows of these matrices give the Laplacian embeddings of our J matched points (before eigenvector alignment). We then calculate the similarity matrix

$$R = \hat{U}_2^T \hat{U}_1. \tag{4.8}$$

Here, the ijth element of $|R|$ gives the similarity between eigenvector i of $\tilde{U}_2$ and eigenvector j of U_1, and the sign of R_{ij} is the same as the sign difference between these eigenvectors. From this similarity matrix, we create a final alignment of eigenvectors by applying the Hungarian algorithm, which in $\mathcal{O}(n^3)$ time finds the bijection $\psi : \{1, \ldots, n\} \to \{1, \ldots, n\}$ that maximizes

$$\sum_{i=1}^{M} \left| R_{i,\psi(i)} \right|. \tag{4.9}$$

For our final choice of alignment, let P be the permutation matrix associated to this ψ, and have S be the diagonal matrix of ± 1 corresponding to the signs of $R_{i,\psi(i)}$.

Algorithm 1: HOSM algorithm overview

Data: Graphs G_1, G_2

Result: A one-to-one matching $\rho : G_1 \to G_2$

Calculate graph Laplacian eigenvectors U_1, U_2 for G_1, G_2 via Nyström's method, section 2.3

Find eigenvector alignment matrices S, P, section 4.2.3

Create subsampled graphs $\tilde{G}_1, \tilde{G}_2$, section 4.2.4

Let $\tilde{U}_1, \tilde{U}_1$ be the rows of U_1, U_2 corresponding to nodes $\tilde{G}_1, \tilde{G}_2$

Define similarity scores between subsampled graphs $\tilde{Q} = \tilde{U}_2 S P \tilde{U}_1^T$

Create match $\tilde{\rho} : \tilde{G}_1 \to \tilde{G}_2$ via Hungarian algorithm on $\tilde{Q}$

forall *nodes* $\tilde{x} \in \tilde{G}_1$ **do**

- Let $H_1 \subseteq G_1$ the nodes in G_1 corresponding to $\tilde{x}$. Similar for $H_2 \subseteq G_2$ and $\tilde{\rho}(\tilde{x})$
- Have $\hat{U}_1, \hat{U}_2$ the rows of U_1, U_2 corresponding to H_1, H_2
- Calculate the similarity scores $Q^* = \hat{U}_2 S P \hat{U}_1^T$
- Create match ρ on the subset H_1 via Hungarian algorithm on Q^*

Combine individual matches into final result $\rho : G_1 \to G_2$

Algorithm 2: HMSM algorithm overview

Data: Graphs G_1, G_2

Result: Matching functions $\rho_{X_1 \to X_2} : G_1 \to G_2$, $\rho_{X_2 \to X_1} : G_2 \to G_1$

Calculate graph Laplacian eigenvectors U_1, U_2 for G_1, G_2 via Nyström's method, section 2.3;

Find eigenvector alignment matrices S, P, section 4.2.3;

Create subsampled graphs $\tilde{G}_1, \tilde{G}_2$, section 4.2.4;

Let $\tilde{U}_1, \tilde{U}_1$ be the rows of U_1, U_2 corresponding to nodes $\tilde{G}_1, \tilde{G}_2$;

Define similarity scores between subsampled graphs $\tilde{Q} = \tilde{U}_2 S P \tilde{U}_1^T$;

forall *nodes* $\tilde{x} \in \tilde{G}_1$ **do**

  Choose a set of several nodes $A_{\tilde{x}} \subseteq \tilde{G}_2$ that are the most similar to $\tilde{x}$ (according to $\tilde{Q}$);

  Let $H_1 \subseteq G_1$ the nodes in G_1 corresponding to $\tilde{x}$;

  Let $H_2 = \cup_{\tilde{y} \in A_{\tilde{x}}}$ (nodes in G_2 corresponding to $\tilde{y}$) .;

  Have $\hat{U}_1, \hat{U}_2$ the rows of U_1, U_2 corresponding to H_1, H_2;

  Calculate the similarity scores $Q^* = \hat{U}_2 S P \hat{U}_1^T$;

  forall *nodes* $x \in H_1$ **do**

    Let $\rho_{X_1 \to X_2}(x)$ be the node in $y \in H_2$ with the largest similarity score (according to Q);

Repeat algorithm with X_1, X_2 interchanged to create $\rho_{X_2 \to X_1}$;

4.2.4 Hierarchical Matching

Based on sections 2.2, 4.2.3, we have created an $N \times M$ matrix

$$Q^* = U_2 S P U_1^T \tag{4.10}$$

where Q^*_{ij} represents the similarity between node i of X_2 and node j of X_1. To create a final match between graphs based on this Q^*, the classical solution is to apply the Hungarian algorithm to find the injection $\rho : \{1, 2, \ldots, M\} \to \{1, 2, \ldots, N\}$ that solves the maximization

problem

$$\rho = \text{argmax}_{\psi} \sum_{i=1}^{M} Q^*_{\psi(i),i}. \tag{4.11}$$

The major benefit of using the Hungarian algorithm is that it results in the optimal one-to-one matching based on the input data, but the $\mathcal{O}(N^3)$ runtime presents a major problem when dealing with larger datasets. To address this we introduce a hierarchical algorithm for graph matching that solves the problem by making many smaller matches, thereby circumventing the N^3 issue. The key step in the hierarchical matching algorithm is the creation of smaller subsampled graphs $\tilde{G}_1, \tilde{G}_2$ of size $K \ll N$. The goal is for $\tilde{G}_1, \tilde{G}_2$ to represent the same geometrical structure as G, H, with significantly reduced size. One common way to this is to iteratively choose the graph node of highest degree, then use a nearest-neighbors algorithm to delete all nearby nodes. However, even this method requires an $\mathcal{O}(N^2)$ calculation of graph node degrees, which is still quite slow. Instead, we use here the approximate node degrees we already calculated as a step in the Nyström extension, given in equation (2.22).

Using the subsampled graphs $\tilde{G}_1, \tilde{G}_2$, we then apply one of two matching algorithms: Hierararchical One-to-One Spectral Matching (HOSM), and Hierarchical Many-to-Many Spectral Matching(HMSM). For HOSM, we first select the part of Q^* related to our two subsampled graphs $\tilde{G}_1, \tilde{G}_2$ and form a match on this level using the Hungarian algorithm to convert these similarity scores into an exact match. Then, for each match $i \to j$ in the subsampled graphs, we run the classical graph matching algorithm between the corresponding clusters in the original graph. So in total the HOSM method applies the Hungarian algorithm $K+1$ times. We create 1 match of size K, and K matches of size $\frac{N}{K}$.

In HMSM, we sacrifice the injectivity property of the graph match in order to increase the speed of the method. Instead of forming an exact match between subsampled graphs $\tilde{G}_1, \tilde{G}_2$, we select a few matches with largest similarity scores for each node in each graph. Similar to HOSM above, these can be thought of as matches between clusters on the level of the original graphs G_1, G_2. To create a final match, we compare each node in G_1 to all the nodes in the clusters it is matched to, and choose the match with the largest similarty

score, and similar for each node in G_2. In the end, we have two functions

$$\rho_{X_1 \to X_2} : G_1 \to G_2 \quad \rho_{X_2 \to X_1} : G_2 \to G_1 \tag{4.12}$$

giving for each node in G_1 one highly-correlated node in G_2, and vice-versa.

Overall, the hierarchical method is significantly faster than the original $\mathcal{O}(N^3)$ algorithm, and allows us to work with much larger datasets. We provide a more in depth analysis of the computational complexity of the algorithm in 4.2.5, as well as experimental results in 4.3.2.

4.2.5 Theoretical runtime computation

Combining the material presented above, we now give a theoretical bound on the time complexity of our HOSM algorithm, proving that our method outperforms $\mathcal{O}(N^2)$. To simplify our calculations, we will assume in this section that both graphs are of size N. As a quick reminder of notation, we will let L denote the number of graph Laplacian eigenvectors to be calculated (using the Nyström method in 2.3), and K denote the size of the subsampled graphs detailed in 4.2.4.

Based on [FBC04], the runtime of the Nyström method is

$$\mathcal{O}(n^3) + \mathcal{O}(L \cdot N). \tag{4.13}$$

Here, the optimization of the choice of n is based on a choice of speed vs. accuracy. In our practical applications (section 4.3) we have found that it suffices to calculate only a small number of eigenvectors to achieve reasonable results, and even in large cases such as $N = \mathcal{O}(10^5)$ we often have $n \approx 100$.

In the graph subsampling step, we use a basic k-nearest neighbors calculation for each cluster to determine which points it should contain. This reduces to sorting a list of size N once for each of K clusters, giving a runtime of

$$\mathcal{O}(K \cdot N \log(N)). \tag{4.14}$$

Lastly, the matching method uses the Hungarian algorithm to create create 1 match of size K, and K matches of size $\frac{N}{K}$, resulting in a runtime of

$$\mathcal{O}\left(K^3 + \frac{N^3}{K^2}\right). \tag{4.15}$$

Without putting too much effort into understanding the details of the implied constants in each of these bounds, we can obtain a very simple bound on the overall runtime by choosing $K = \mathcal{O}\left(N^{\frac{3}{5}}\right)$, giving us a final runtime of

$$\begin{aligned} &\mathcal{O}(L^3) + \mathcal{O}(L \cdot N) + \mathcal{O}(N^{\frac{8}{5}} \log(N)) + \mathcal{O}(N^{\frac{9}{5}}) \\ &= \mathcal{O}(N^{\frac{9}{5}}), \end{aligned} \tag{4.16}$$

where in the last equation we assume that L is small relative to N.

A bound on the runtime of HMSM involves very similar calculation. The only change is in the matching algorithm, where we replace the Hungarian algorithm (4.15) with a simple maximum, resulting in a runtime of

$$\begin{aligned} &\mathcal{O}(n^3) + \mathcal{O}(n \cdot N) \\ &+ \mathcal{O}(K \cdot N \log(N)) + \mathcal{O}\left(K^2 + \frac{N^2}{K}\right). \end{aligned} \tag{4.17}$$

Choosing $K = \mathcal{O}(\sqrt{N})$ here gives an asymptotic runtime of $\mathcal{O}(N^{\frac{3}{2}} \log(N)$.

4.3 Experimental Results

In this section, we evaluate the performance in large graph matching of our HOSM and HMSM algorithms. In 4.3.1 we show through a simple example the ability of our method to recognize and match salient features in datasets, and in 4.3.2 we experimentally calculate the algorithm runtime, as well as draw comparisons in matching error against other leading algorithms. Finally, in section 4.3.3 we show the results of our algorithm on a real-world dataset.

4.3.1 Toy example of hierarchical match

We begin by showing the results of our graph matching algorithm on a synthetic dataset, which is pictured in figure 4.3. Here the nodes of the graphs G_1, G_2 are represented by 2-dimensional datasets X_1, X_2, and the weight matrices W_1, W_2 are determined via a classical RBF kernel applied to the 2-norm, as in 2.2.

$$E_1(i,j) = \|X_1(i) - X_1(j)\|_2^2 \tag{4.18}$$

$$W_1(i,j) = -\exp\left(\frac{E_1(i,j)}{std(E_1)^2}\right), \tag{4.19}$$

and similarly for W_2. The matching is then calculated using the our HOSM method in 4.2.4, with the match on the subsampled level begin shown in figures 4.3d, 4.3e, 4.3f. This example has datasets of size $N = 1500$, with $K = 50$, so that the subsampled data has size $\left|\tilde{G}_1\right| = \left|\tilde{G}_2\right| = 30$.

The purpose of this example is to show that the matching algorithm can recognize similar shapes in data that has been altered in a smooth-enough manner. Both sets X_1, X_2 contain a tight cluster of points, as well as longer line segment. In figures 4.3c, 4.3f we see the final result of the algorithm, where each match is represented by a line connecting the two points in question. As we can see in the figure, the algorithm successfully matches the objects based on their shape, as we desired.

4.3.2 Runtime and error calculation

In this set of experiments we implement our algorithm on random synthetic graphs, giving a practical estimation of runtime to complement our theoretical bound in section 4.2.5, as well as comparing the resulting graph matching error against established methods.

First, we show our results on algorithm runtime in figures 4.4, 4.5 over a selection of random graphs of a variety of sizes. For each graph size, we run the method 60 times and give the average time in seconds in figures 4.4a, 4.5a. We also present the same data in a log-log plot in figures 4.4b, 4.5b to give an estimate of the asymptotic complexity of the method. Supporting our ideas from 4.2.5, our experiment estimates the runtime for HOSM

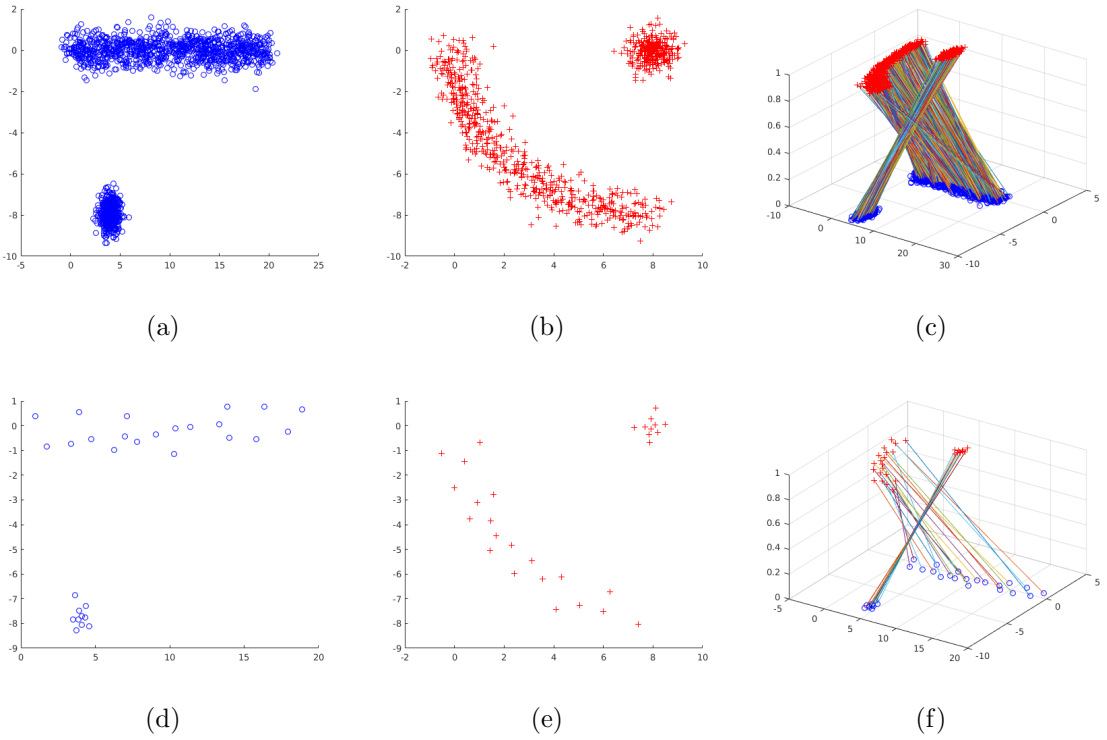


Figure 4.3: Example hierarchical matching on synthetic data. (a) Dataset X_1. (b) Dataset X_2. (c) Hierarchical match result. (d) Subsampled data $\tilde{X}_1$. (e) Subsampled data $\tilde{X}_2$. (f) Classical match on subsampled data

at roughly $\mathcal{O}(N^{1.6})$, and for HMSM at $\mathcal{O}(N^{1.4})$. For a comparison, we present in table 4.1 a short review of other state-of-the-art methods and their associated runtimes.

We next present in table 4.2 a comparison of matching error with several established methods. Similar to section 4.3.1 we create example graphs from synthetic 2-dimensional datasets X_1, X_2 via an RBF kernel, as in equations (4.18), (4.19). We then apply 7 different methods for graph matching, and for each we calculate the matching energy as in equation (4.2). As computing the matching error this way is an expensive operation, we limit the size of our graphs to $N = 300$ in each of these experiments.

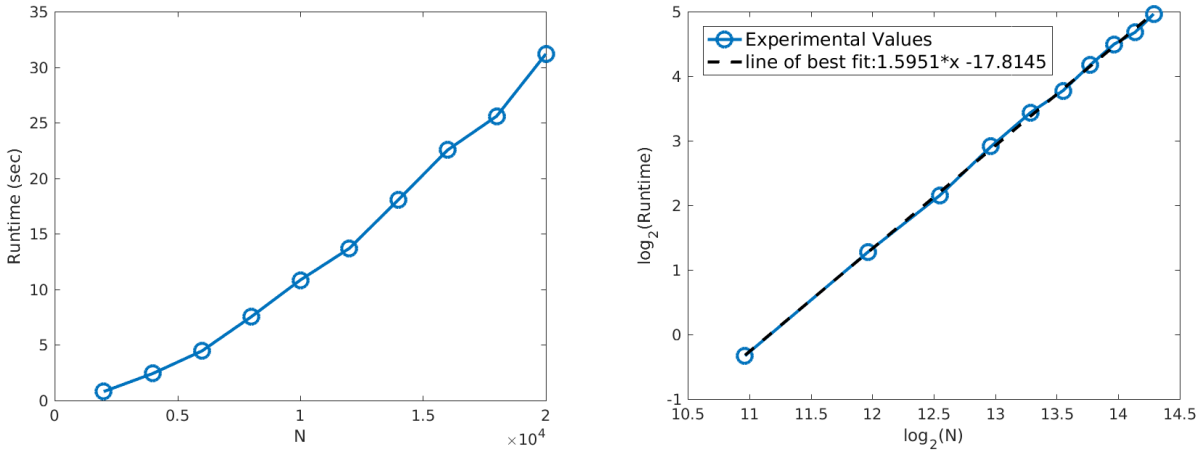
The methods used are as follows: *Rand* chooses the match uniform randomly, providing a basis for comparison. *PATH* is a convex-concave algorithm presented in [ZBV09], and *FastPFP* is a fixed-point method shown in [LHL16]. The remaining 4 models are ours, with

Method	Asymptotic runtime	Graph size considered
IPFP [LHS09]	$\mathcal{O}(N^4)$	$\mathcal{O}(10^1)$
FastGA [GR96]	$\mathcal{O}(N^3)$	$\mathcal{O}(10^2)$
Umeyama [Ume88]	$\mathcal{O}(N^3)$	$\mathcal{O}(10^2)$
PATH [ZBV09]	$\mathcal{O}(N^3)$/iteration	$\mathcal{O}(10^2)$
FAQ [VCL15]	$\mathcal{O}(N^3)$/iteration	$\mathcal{O}(10^3)$
FastPFP [LHL16]	$\mathcal{O}(N^3)$/iteration	$\mathcal{O}(10^3)$
Knossow[1,2] [KSM09]	$\mathcal{O}(N^2)$	$\mathcal{O}(10^4)$
HOSM	$\mathcal{O}(N^{1.8})$	$\mathcal{O}(10^4)$ or $\mathcal{O}(10^5)$
HMSM[1]	$\mathcal{O}(N^{1.5}\log(N))$	$\mathcal{O}(10^5)$

[1] Not a 1-to-1 match.

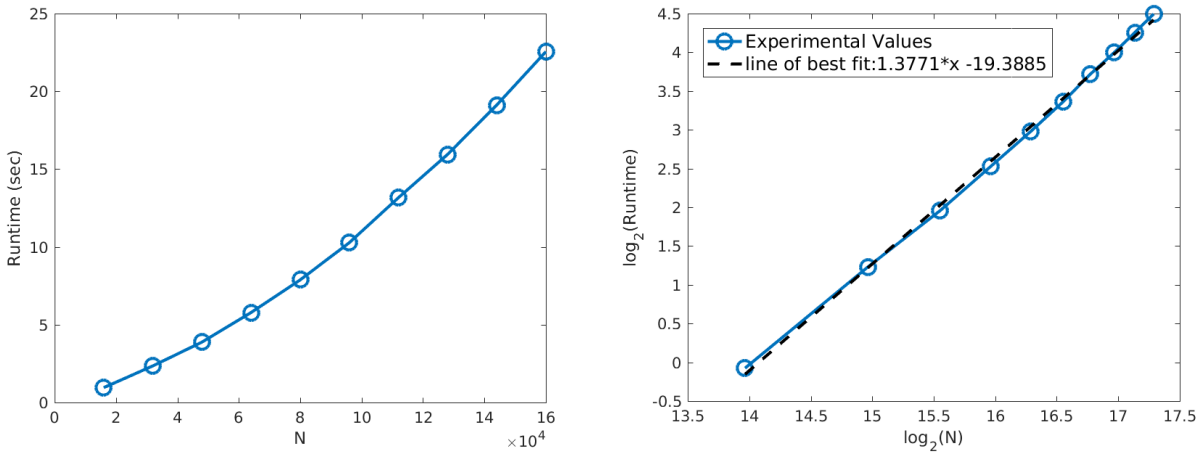
[2] Only considers sparse graphs.

Table 4.1: Theoretical runtime and largest size of graph considered over current state-of-the-art large graph matching algrotihms.



(a) (b)

Figure 4.4: Average runtime of HOSM over 60 trials for graphs of size 2,000 to 20,000. (a) Raw data. (b) Log-log plot with line of best fit.



(a) (b)

Figure 4.5: Average runtime of HMSM over 60 trials for graphs of size 16,000 to 160,000. (a) Raw data. (b) Log-log plot with line of best fit.

different choices of the hierarchical matching method from section 4.2.4. Note that choosing $K = N$ eschews the hierarchical portion of the algorithm and directly creates and uses the similarity matrix Q^* from (4.7).

Our first experiment uses a similar "line segment + cluster" dataset as in figure 4.3. We use this experiment to measure the ability of algorithms to match graphs that are similar topologically, but not truly isomorphic. In the next experiment each graph contains two clusters of similar composition, testing the eigenvector alignment step of the spectral methods (section 4.2.3). We then give three examples covering the case of exactly isomorphic graphs with various levels of added noise. Lastly, we address the case of subgraph isomorphism, where $|G_1| = 150, |G_2| = 300$, and G_2 contains as a subgraph an exact copy of G_1.

As can be seen from the data, our algorithms are competitive with the state-of-the-art in each experiment. Most importantly, for many of the experiments the hierarchical matching algorithms (those with $K = N/10$) show comparable error rate with the other algorithms, proving that our choice of approximations preserves the quality of the final match.

Experiment	Rand	PATH	FastPFP	HOSM $K=N$	HOSM $K=N/10$	HMSM $K=N$ *	HMSM $K=N/10$ *
Line segment + cluster	1.390e+4	639.61	1381.6	585.36	1023.4	328.62	632.76
Two clusters	3.059e+4	27.28	6.538	6.1398	20.176	4.9154	8.1589
Isomorphic graphs	1.460e+4	5.2652	14.527	0.000	364.49	0.000	119.28
Isomorphism + 10% perturbation	1.441e+4	94.706	83.413	72.206	535.05	76.272	155.26
Isomorphism + 20% perturbation	1.426e+4	363.13	287.54	278.29	826.99	292.02	451.21
Subgraph isomorphism	1.260e+4	786.80	785.35	6.4151	193.76	39.406	151.68

* Not a 1-to-1 match

Table 4.2: Matching error comparison over several experiments. Methods implemented are (a) Rand, a uniform random match, (b) PATH, presented in [ZBV09], (c) FastPFP, presented in [LHL16], (d,e) HOSM, presented in this paper, with various choices of subsampled graph size, (f,g) HMSM, presented in this paper, with various choices of subsampled graph size.

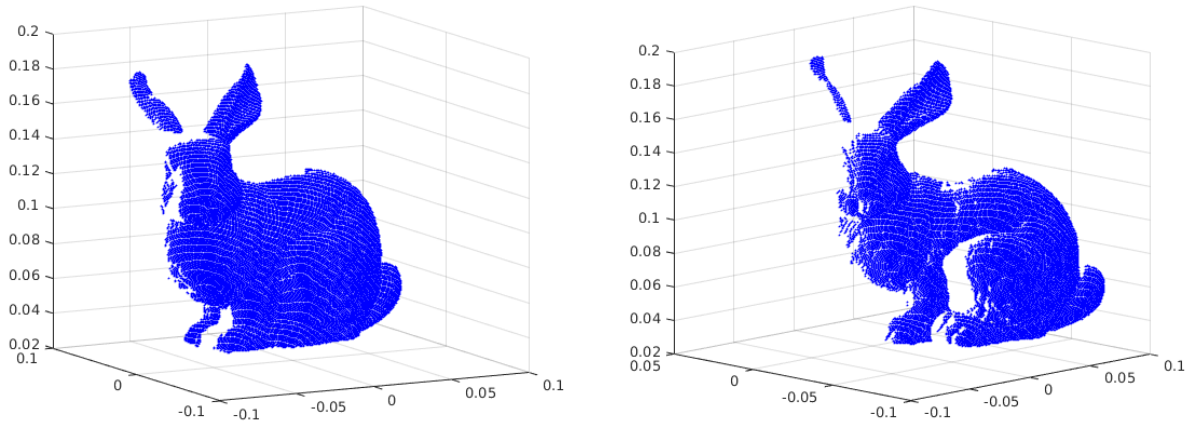
4.3.3 Stanford Bunny experiment

In this section we show the results of our algorithm applied to the Stanford Bunny dataset [TL94]. The set consists of several 3D point clouds showing the surface of a rabbit as viewed from different angles, and is publicly available (thanks to the Stanford Computer Graphics Library) at

$$http://graphics.stanford.edu/data/3Dscanrep/.$$

We show in figure 4.6 the 3D realization of two of these sets with a direct view (0°-angle) in figure 4.6a, and a 45°-angle view in figure 4.6b. These figures contain 40,256 and 40,097 data points, respectively, falling well within the reasonable range of our matching algorithms. We convert these point clouds into weighted graphs using the same RBF kernel as described in (4.19), then proceed to match the figures using our HOSM and HMSM methods. To visualize the results, we give in figures 4.7a, 4.8a, and 4.9a hand-made colorings of one angle of the bunny, then in figures 4.7b, and 4.8b, 4.9b we use the various matching functions created to transfer the coloring to the second angle.

Figures 4.7 and 4.8 give the two matching functions (neither of which is one-to-one) calculated by HMSM. In figure 4.7 we begin with a coloring on the 45°-angle dataset, then

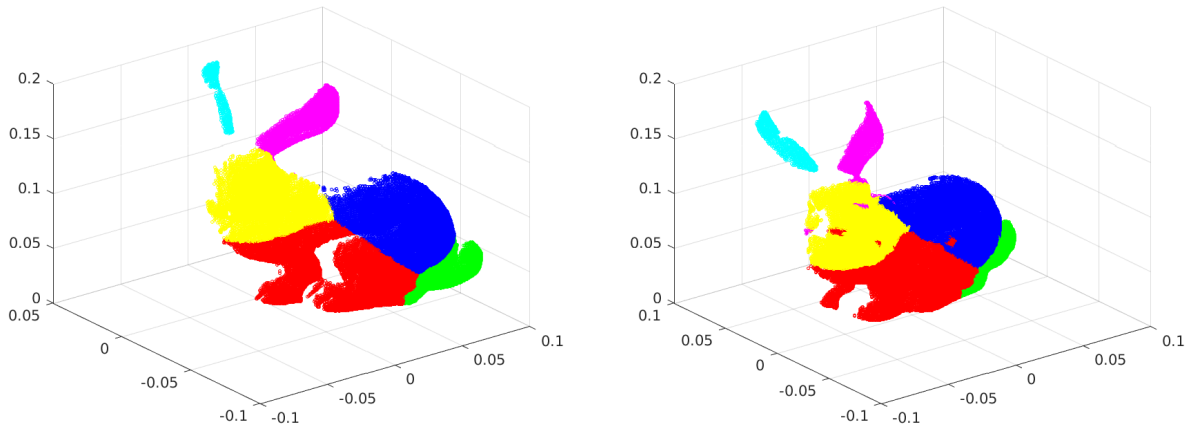


(a) (b)

Figure 4.6: Stanford Bunny Dataset. 3D point clouds representing the surface of a rabbit as viewed (a) from directly in front (angle of 0°), and (b) with a 45° rotation.

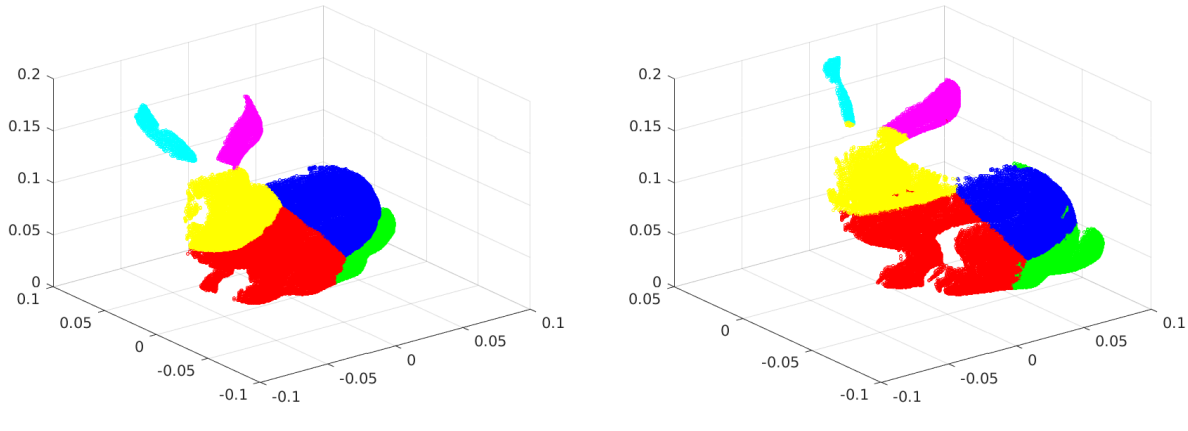
perform a match from 0° to 45°. The coloring on the 0° set is then determined by pulling back the coloring on the 45° through this match. Figure 4.8 shows the same operation performed in the opposite direction. As can be seen in the figure, the overall results of the match are quite good, with only small errors in each direction.

In figure 4.9 we show the one-to-one match found by HOSM. Compared to the results of the many-to-many method we see more errors in this case, which speaks to the increased difficulty of finding a reasonable one-to-one match in a situation such as this where no exact match exists. This difficulty is especially apparent when looking at the ears of the bunny. Each ear is a distinct feature of the figure, but between the two angles the number of data points per ear is quite different. In particular, there are are many fewer data points in the rear ear (light blue) of the 45°-angle set than in the 0°-angle set. Since we require the match to be one-to-one, there not enough available blue points to properly color the back ear in the target data, which is the major cause of the erroneous coloring here.



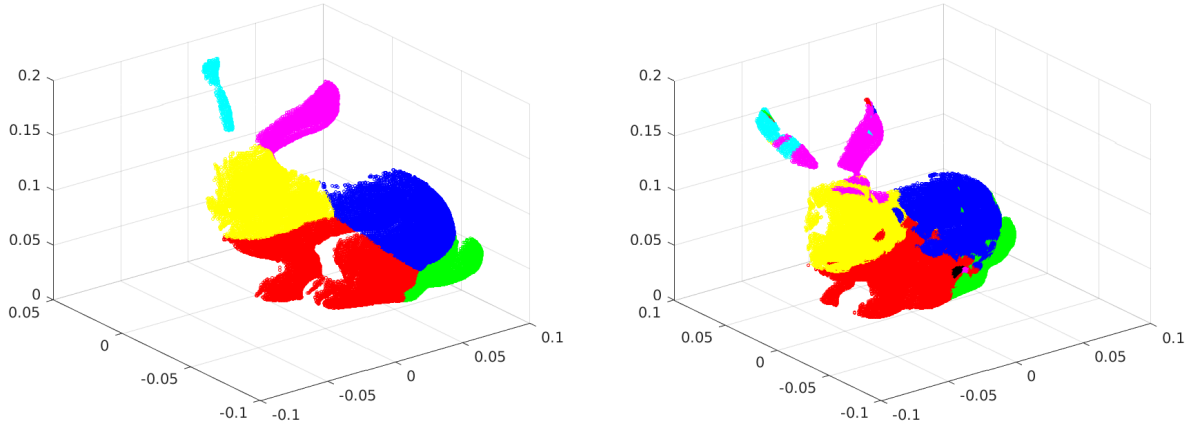
(a) (b)

Figure 4.7: Representation of HMSM match on Stanford Bunny. (a) A coloring of the 45°-angle dataset. (b) The transfer of the coloring onto the 0°-angle dataset via $color(i) = color(\rho_{0^\circ \to 45^\circ}(i))$.



(a) (b)

Figure 4.8: Representation of HMSM match on Stanford Bunny. (a) A coloring of the 0°-angle dataset. (b) The transfer of the coloring onto the 0°-angle dataset via $color(i) = color(\rho_{45^\circ \to 0^\circ}(i))$.



(a) (b)

Figure 4.9: Representation of HOSM match on Stanford Bunny. (a) A coloring of the 45°-angle dataset. (b) The transfer of the coloring onto the 0°-angle dataset via the match found.

4.4 Graph Matching Applications

In this section we show several application of our graph matching algorithm to real-world machine learning problems. In 4.4.1 we introduce a method for change detection on co-registered sets based on cparing the results of a graph match to the natural indexing on sets. In 4.4.2 we show an example of knowledge transfer derived from pulling any relevant information through a graph match.

4.4.1 Change detection using graph matching

One possible application of our graph matching algorithm is in change detection. Suppose we are given datasets X_1, X_2 representing the same event, captured at different times or with different sensors. One should expect that the two datasets represent roughly the same information, but an interesting question is to find when this is not the case, and in which specific parts of the data do we see this difference. Speaking more mathematically, it is reasonable to expect that the topology of the two dataset would be roughly similar, and

that any topological differences would be the points of interest for which we search. Graph representations of our data are particularly suitable for this purpose, as they preserve relevant topological information while simultaneously filtering out specific formatting details of each sensor. Based on these ideas, we haved formed a change detection algorithm for *co-registered* datasets - that is, datasets that share a common indexing. By comparing the matching found by our graph algorithm to the inherent indexing, we are able to locate and highlight the regions where the two datasets do not agree.

The exact algorithm is as follows: given datasets X_1, X_2 of size N, we create graphs G_1, G_2 by choosing relevant weight functions for each datasets. We then apply our matching algorithm to find a permutation

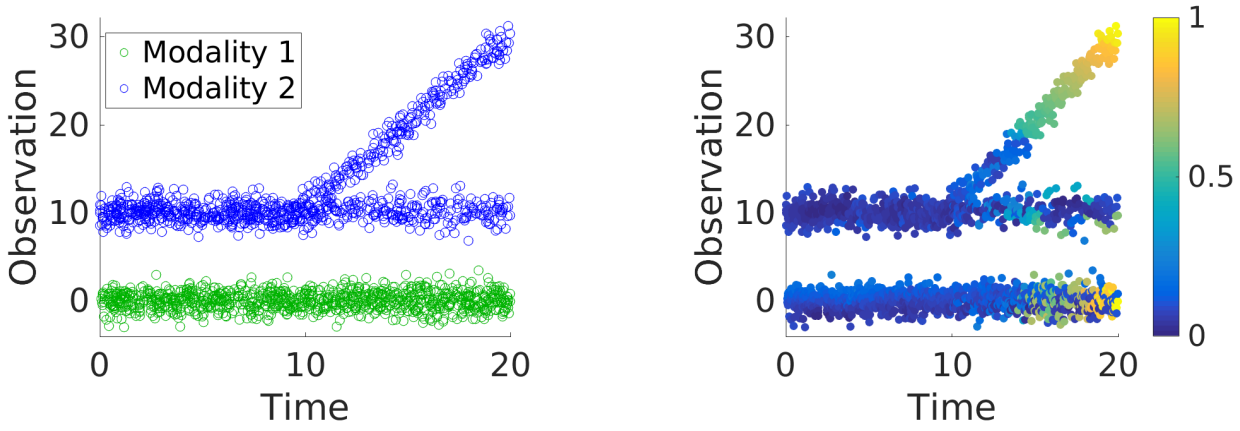
$$\rho : \{1, 2, \dots, N\} \to \{1, 2, \dots, N\} \tag{4.20}$$

We can then measure the degree of change at each graph node by comparing the inherent indexing against the graph match permutation.

$$\begin{aligned} &\text{Change in } X_1 \text{ at node } i = \|X_1(i) - X_1(\rho(i))\| \\ &\text{Change in } X_2 \text{ at node } i = \left\|X_2(\rho^{-1}(i)) - X_2(i)\right\| \end{aligned} \tag{4.21}$$

As a first example, we show in figure 4.10 a synthetic dataset consisting of two one-dimensional timeseries. Following our discussion above, we create the two sets with enough common structure to make graph matching a reasonable option, but with one significant difference between the two sets. After performing a graph match, we calculate the quantity of change at each node via (4.21) and display the results in fig 4.10b. As desired, the new feature added to modality 2 is highlighted as a change.

We next apply the algorithm to a real-life dataset with a synthetic change, with the results shown in figure 4.11. Here we use for the dataset X_1 (fig 4.11a) an image of an indoor scene of 261×378 pixels, and for the set X_2 (fig 4.11d) we apply a continuous transformation to each pixel, as well as add a significant artifact in the upper-left corner. This choice of X_2 mimics the discussion above, where the change in the majority of pixels represents the perturbations caused from using a different sensor, and the red square represents some



(a) Input data

(b) Scaled norms

Figure 4.10: Change detection appplied to synthetic timeseries data. (a) The input data, roughly equal for times $t \in [0, 10]$, and showing increasing change over $t \in [10, 20]$. (b) Differences as calculated in (4.21)

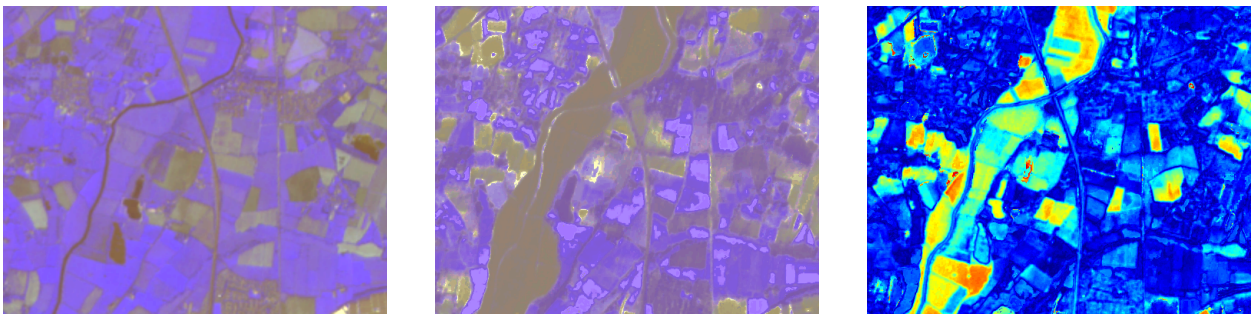
information detected by one sensor and not by the other. The intermediate steps of the algorithm in figures 4.11b, 4.11e show how the matching algorithm permutes the pixels of X_1 to agree with the topology of X_2, and vice versa. As expected there is very little change between the original input data and its permuted version, except for the added square in the corner. Therefore, in the final results in figures 4.11c, 4.11f, we see the square artifact highlighted, and relatively little other change found.

For a real-world example of this algorithm, we apply our method to the 2010 Data Fusion Contest data [LPG12], with results shown in figure 4.12. The dataset consists of SPOT satellite images of Gloucester, UK, taken before and after a flood event in November 2000. The SPOT satellite uses three channels to capture light from the spectral range $0.50\mu m$ to $0.89\mu m$ (green to near-infrared), which we display in false color in figures 4.12a (before flood), and 4.12d (after flood). As can be seen from the figures, there is a significant topographical change between the two datasets where the river overflowed and spilled onto the nearby land. This is evident in the intermediate figures 4.12b, 4.12e, as the algorithm is forced to make significant changes to permute the pixels of X_1 to resemble X_2, and vice versa. In our results in figures 4.12c and 4.12f we see that our algorithm successfully highlights these areas.

(a) Image X_1 (b) $X_1(\rho(i))$ (c) $\|X_1(i) - X_1(\rho(i))\|$

(d) Image X_2 (e) $X_2(\rho^{-1}(i))$ (f) $\left\|X_2(\rho^{-1}(i)) - X_2(i)\right\|$

Figure 4.11



(a) Image X_1 (before flood) (b) $X_1(\rho(i))$ (c) $\|X_1(i) - X_1(\rho(i))\|$

(d) Image X_2 (after flood) (e) $X_2(\rho^{-1}(i))$ (f) $\left\|X_2(\rho^{-1}(i)) - X_2(i)\right\|$

Figure 4.12: Example change detection on DFC 2010 data

4.4.2 Knowledge transfer via graph matching

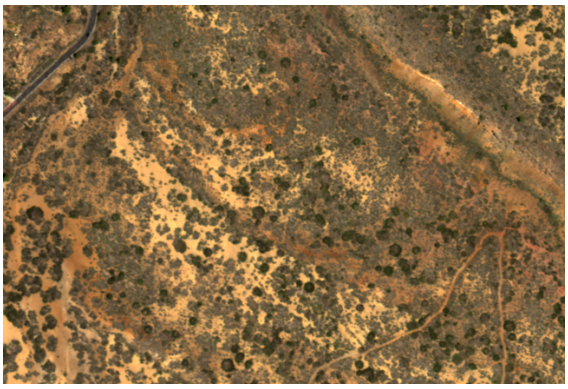
For another potential application of our graph matching algorithm, we present here method for knowledge transfer. Given two datasets with roughly similar topology, we are able to transfer any known knowledge about one set to the other by performing a graph match. Notate our two sets X_1, X_2, with pre-existing information on X_1. Using our HMSM method to create a matching function $\rho_{2\to 1} : X_2 \to X_1$, we can transfer the information on X_1 to X_2 by letting

$$\text{label}(X_2(i)) = \text{label}\left(X_1(\rho_{2\to 1}(i))\right) \tag{4.22}$$

A first example of this method has already been shown in this paper in the Stanford bunny experiment (section 4.3.3). In figures 4.7, 4.8 we begin with a pre-defined coloring on one dataset, and transfer it to the other. As a second example, we apply our algorithm in a remote sensing context, with results shown in figure 4.13. The input data consists of 68 bands of hyperspectral data taken of a desert area, split into several patches. We display the RGB bands of two of these patches, which we will call X_1, X_2, in figures 4.13a, 4.13b. Along with the hyperspectral data we are given a classification of image X_1 into 6 classes, shown in figure 4.13c. The results of the transfer is then shown in figure 4.13d.

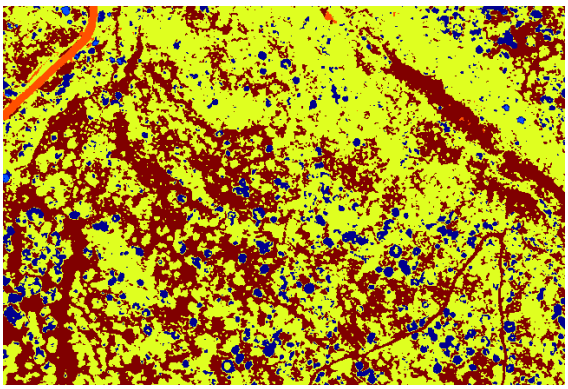
4.5 Summary

In conclusion, we have proposed a method for inexact matching of large graphs that significantly improves on the space and time complexity of existing methods, allowing us to work with much larger graphs. By applying a hierarchical scheme to our matching algorithm, we are able to ignore a large percent of unlikely matches. Furthermore, extensive experiments show that the approximations made in the course of our method do not significantly affect the accuracy of the match. Future plans include a more thorough investigation of the graph subsampling step in 4.2.4, as well as attempts to improve the flexibility of the hierarchical matching algorithm.

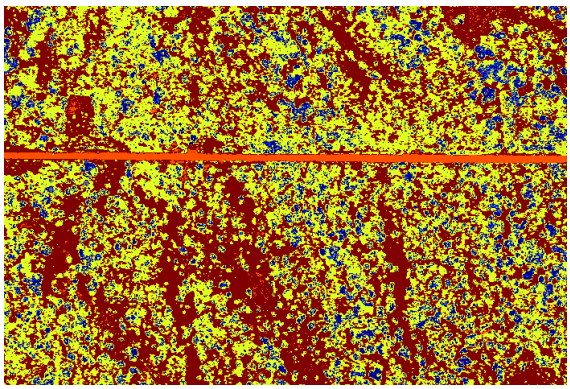


(a) Image X_1

(b) Image X_2



(c) Ground Truth classification of X_1



(d) Transfer to X_2

Figure 4.13: Knowledge transfer on remote sensing data

[ABD17] H. Andrés Helfgott, J. Bajpai, and D. Dona. "Graph isomorphisms in quasi-polynomial time." *ArXiv e-prints*, October 2017.

[AKK12] Vikraman Arvind, Johannes Köbler, Sebastian Kuhnert, and Yadu Vasudev. *Approximate Graph Isomorphism*, pp. 100–111. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[And10] C. R. 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, September 2010.

[Art79] D. W. Arthur. "C. T. H. Baker, The Numerical Treatment of Integral Equations (Clarendon Press; Oxford University Press, 1978), xiv 1034 pp., 2250." *Proceedings of the Edinburgh Mathematical Society*, **22**(1):6767, 1979.

[ATA15] S. Ben Ayed, H. Trichili, and A. M. Alimi. "Data fusion architectures: A survey and comparison." In *2015 15th International Conference on Intelligent Systems Design and Applications (ISDA)*, pp. 277–282, Dec 2015.

[Bab15] László Babai. "Graph Isomorphism in Quasipolynomial Time." *CoRR*, **abs/1512.03547**, 2015.

[BBM05] A. C. Berg, T. L. Berg, and J. Malik. "Shape matching and object recognition using low distortion correspondences." In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pp. 26–33 vol. 1, 2005.

[BF12] Andrea Bertozzi and Arjuna Flenner. "Diffuse Interface Models on Graphs for Classification of High Dimensional Data." **10**, 07 2012.

[BFC02] Serge Belongie, Charless Fowlkes, Fan Chung, and Jitendra Malik. *Spectral Partitioning with Indefinite Kernels Using the Nyström Extension*, pp. 531–542. Springer Berlin Heidelberg, Berlin, Heidelberg, 2002.

[BMP02] S. Belongie, J. Malik, and J. Puzicha. "Shape matching and object recognition using shape contexts." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **24**(4):509–522, Apr 2002.

[BN03] M. Belkin and P. Niyogi. "Laplacian Eigenmaps for Dimensionality Reduction and Data Representation." *Neural Computation*, **15**(6):1373–1396, June 2003.

[BW03] Andrew D. Bagdanov and Marcel Worring. "First order Gaussian graphs for efficient structure classification." *Pattern Recognition*, **36**(6):1311 – 1324, 2003.

[CBC07] N. Cvejic, D. Bull, and N. Canagarajah. "Region-Based Multimodal Image Fusion Using ICA Bases." *IEEE Sensors Journal*, **7**(5):743–751, May 2007.

[CBS05] T. Cour, F. Benezit, and J. Shi. "Spectral segmentation with multiscale graph decomposition." In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, pp. 1124–1131 vol. 2, June 2005.

[CFS04] Donatello Conte, Pasquale Foggia, Carlo Sansone, and Mario Vento. "Thirty Years Of Graph Matching In Pattern Recognition." *International Journal of Pattern Recognition and Artificial Intelligence*, **18**:265–298, 05 2004.

[Chu97] Fan R. K. Chung. *Spectral Graph Theory*. American Mathematical Society, 1997. *[SIGNATUR = 789.018.]*

[CK04] T. Caelli and S. Kosinov. "An eigenspace projection clustering method for inexact graph matching." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **26**(4):515–519, April 2004.

[CRG16] M. Campos-Taberner, A. Romero-Soriano, C. Gatta, G. Camps-Valls, A. Lagrange, B. Le Saux, A. Beaupre, A. Boulch, A. Chan-Hon-Tong, S. Herbin, H. Randrianarivo, M. Ferecatu, M. Shimoni, G. Moser, and D. Tuia. "Processing of Extremely High-Resolution LiDAR and RGB Data: Outcome of the 2015 IEEE GRSS Data Fusion Contest #8211;Part A: 2-D Contest." *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, **9**(12):5547–5559, Dec 2016.

[CSS07] Timothee Cour, Praveen Srinivasan, and Jianbo Shi. "Balanced Graph Matching." In B. Schölkopf, J. C. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*, pp. 313–320. MIT Press, 2007.

[DMH14] C. Debes, A. Merentitis, R. Heremans, J. Hahn, N. Frangiadakis, T. van Kasteren, W. Liao, R. Bellens, A. Piurica, S. Gautama, W. Philips, S. Prasad, Q. Du, and F. Pacifici. "Hyperspectral and LiDAR Data Fusion: Outcome of the 2013 GRSS Data Fusion Contest." *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, **7**(6):2405–2418, June 2014.

[DRW10] Tamal K. Dey, Pawas Ranjan, and Yusu Wang. *Convergence, Stability, and Discrete Approximation of Laplace Spectra*, pp. 650–663. 2010.

[EKB15] D. Eynard, A. Kovnatsky, M. M. Bronstein, K. Glashoff, and A. M. Bronstein. "Multimodal Manifold Analysis by Simultaneous Diagonalization of Laplacians." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **37**(12):2505–2517, Dec 2015.

[FAV11] Lucas Franek, Daniel Duarte Abdala, Sandro Vega-Pons, and Xiaoyi Jiang. *Image Segmentation Fusion Using General Ensemble Clustering Methods*, pp. 373–384. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

[FBC04] Charless Fowlkes, Serge Belongie, Fan Chung, and Jitendra Malik. "Spectral Grouping Using the Nystrom Method." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **26**(2), February 2004.

[FPV14] Pasquale Foggia, Gennaro Percannella, and Mario Vento. "Graph Matching and Learning in Pattern Recognition in the Last 10 Years." *International Journal of Pattern Recognition and Artificial Intelligence*, 2 2014.

[GB12] Yves van Gennip and Andrea L. Bertozzi. "Γ-convergence of graph Ginzburg-Landau functionals." *Adv. Differential Equations*, **17**(11/12):1115–1180, 11 2012.

[GH94] Olivier Goldschmidt and Dorit S. Hochbaum. "A Polynomial Algorithm for the k-Cut Problem for Fixed k." *Mathematics of Operations Research*, **19**(1):24–37, 1994.

[GMB14] C Garcia-Cardona, E Merkurjev, AL Bertozzi, A Flenner, and AG. Percus. "Multiclass data segmentation using diffuse interface methods on graphs." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **36**(8):1600–1613, 2014.

[GR96] S. Gold and A. Rangarajan. "A graduated assignment algorithm for graph matching." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **18**(4):377–388, Apr 1996.

[GS06] L. Grady and E. L. Schwartz. "Isoperimetric graph partitioning for image segmentation." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **28**(3):469–475, March 2006.

[HH99] Benoit Huet and Edwin R. Hancock. "Shape recognition from large image libraries by inexact graph matching." *Pattern Recognition Letters*, **20**(11):1259 – 1269, 1999.

[HSB15] Huiyi Hu, Justin Sunu, and Andrea L. Bertozzi. *Multi-class Graph Mumford-Shah Model for Plume Detection Using the MBO scheme*, pp. 209–222. Springer International Publishing, Cham, 2015.

[ICB17] Geoffrey Iyer, Jocelyn Chanussot, and Andrea Bertozzi. "A Graph-Based Approach for Feature Extraction and Segmentation of Multimodal Images." 2017.

[ICB18a] Geoffrey Iyer, Jocelyn Chanussot, and Andrea Bertozzi. "Fast Large Graph Matching via Spectral Methods." submitted for publication, 2018.

[ICB18b] Geoffrey Iyer, Jocelyn Chanussot, and Andrea Bertozzi. "A Graph-Based Approach for Data Fusion and Segmentation of Multimodal Images." submitted for publication, 2018.

[KC02] Serhiy Kosinov and Terry Caelli. "Inexact Multisubgraph Matching Using Graph Eigenspace and Clustering Models." In Terry Caelli, Adnan Amin, Robert P. W. Duin, Dick de Ridder, and Mohamed Kamel, editors, *Structural, Syntactic, and Statistical Pattern Recognition*, pp. 133–142, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg.

[KSM09] David Knossow, Avinash Sharma, Diana Mateus, and Radu Horaud. *Inexact Matching of Large and Sparse Graphs Using Laplacian Eigenvectors*, pp. 144–153. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[KZ04] V. Kolmogorov and R. Zabin. "What energy functions can be minimized via graph cuts?" *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **26**(2):147–159, Feb 2004.

[LAJ15] Dana Lahat, Tülay Adalı, and Christian Jutten. "Multimodal Data Fusion: An Overview of Methods, Challenges and Prospects." *Proceedings of the IEEE*, **103**(9):1449–1477, August 2015.

[Lan50] Cornelius Lanczos. "An iterative method for the solution of the eigenvalue problem of linear differential and integral.", 1950.

[LH05] M. Leordeanu and M. Hebert. "A spectral technique for correspondence problems using pairwise constraints." In *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, volume 2, pp. 1482–1489 Vol. 2, Oct 2005.

[LHL16] Yao Lu, Kaizhu Huang, and Cheng-Lin Liu. "A fast projected fixed-point algorithm for large graph matching." *Pattern Recognition*, **60**:971 – 982, 2016.

[LHS09] Marius Leordeanu, Martial Hebert, and Rahul Sukthankar. "An Integer Projected Fixed Point Method for Graph Matching and MAP Inference." In Y. Bengio, D. Schuurmans, J. D. Lafferty, C. K. I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems 22*, pp. 1114–1122. Curran Associates, Inc., 2009.

[LPB15] W. Liao, A. Piurica, R. Bellens, S. Gautama, and W. Philips. "Generalized Graph-Based Fusion of Hyperspectral and LiDAR Data Using Morphological Features." *IEEE Geoscience and Remote Sensing Letters*, **12**(3):552–556, March 2015.

[LPC14] D. Lunga, S. Prasad, M. M. Crawford, and O. Ersoy. "Manifold-Learning-Based Feature Extraction for Classification of Hyperspectral Data: A Review of Advances in Manifold Learning." *IEEE Signal Processing Magazine*, **31**(1):55–66, Jan 2014.

[LPG12] Nathan Longbotham, Fabio Pacifici, Taylor Glenn, Alina Zare, Michele Volpi, Devis Tuia, Emmanuel Christophe, Julien Michel, Jordi Inglada, Jocelyn Chanussot, and Qian Du. "Multi-modal change detection, application to the detection of flooded areas: outcome of the 2009-2010 data fusion contest." *IEEE J. Sel. Topics Appl. Earth Observ.*, **5**(1):331–342, Feb. 2012.

[LQ14] Z. Y. Liu and H. Qiao. "GNCCP #x2014;Graduated NonConvexityand Concavity Procedure." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **36**(6):1258–1267, June 2014.

[Lux07] Ulrike von Luxburg. "A tutorial on spectral clustering." *Statistics and Computing*, **17**(4):395–416, 2007.

[LVY12] X. Lei, P. A. Valdes-Sosa, and D. Yao. "EEG/fMRI fusion based on independent component analysis: integration of data-driven and model-driven methods." *J. Integr. Neurosci.*, **11**(3):313–337, Sep 2012.

[MGB14] 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.

[MKB13] Ekaterina Merkurjev, Tijana Kostic, and Andrea L Bertozzi. "An MBO scheme on graphs for classification and image processing." *SIAM Journal on Imaging Sciences*, **6**:1903–1930, October 2013.

[MLY17] K. Ma, H. Li, H. Yong, Z. Wang, D. Meng, and L. Zhang. "Robust Multi-Exposure Image Fusion: A Structural Patch Decomposition Approach." *IEEE Transactions on Image Processing*, **PP**(99):1–1, 2017.

[MMK17] 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.

[Moh91] Bojan Mohar. "The Laplacian Spectrum of Graphs." *Graph Theory, Combinatorics, and Applications*, **2**:871–898, 1991.

[MS07] Nikolaos Mitianoudis and Tania Stathaki. "Pixel-based and region-based image fusion schemes using {ICA} bases." *Information Fusion*, **8**(2):131 – 142, 2007. Special Issue on Image Fusion: Advances in the State of the Art.

[Nys30] E. J. Nystrm. "ber Die Praktische Auflsung von Integralgleichungen mit Anwendungen auf Randwertaufgaben." *Acta Math.*, **54**:185–204, 1930.

[Pie03] Gemma Piella. "A general framework for multiresolution image fusion: from pixels to regions." *Information Fusion*, **4**(4):259 – 280, 2003.

[PNG03] Gerasimos Potamianos, Chalapathy Neti, Guillaume Gravier, Ashutosh Garg, and Andrew W. Senior. "Recent advances in the automatic recognition of audiovisual speech." *Proceedings of the IEEE*, **91**(9):1306–1326, Sept 2003.

[PTV92] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C (2Nd Ed.): The Art of Scientific Computing*. Cambridge University Press, New York, NY, USA, 1992.

[RKD15] Jimmy Francky Randrianasoa, Camille Kurtz, Éric Desjardin, and Nicolas Passat. *Multi-image Segmentation: A Collaborative Approach Based on Binary Partition Trees*, pp. 253–264. Springer International Publishing, Cham, 2015.

[RKM13] W. Gray Roncal, Z. H. Koterba, D. Mhembere, D. M. Kleissas, J. T. Vogelstein, R. Burns, A. R. Bowles, D. K. Donavos, S. Ryman, R. E. Jung, L. Wu, V. Calhoun, and R. J. Vogelstein. "MIGRAINE: MRI Graph Reliability Analysis and Inference for Connectomics." In *2013 IEEE Global Conference on Signal and Information Processing*, pp. 313–316, Dec 2013.

[Saa11] Y. Saad. *Numerical Methods for Large Eigenvalue Problems*. Society for Industrial and Applied Mathematics, 2011.

[SBR16] Farnaz Sedighin, Massoud Babaie-Zadeh, Bertrand Rivet, and Christian Jutten. "Two Multimodal Approaches for Single Microphone Source Separation." In *24th European Signal Processing Conference (EUSIPCO 2016)*, pp. 110–114, Budapest, Hungary, September 2016.

[SBV00] Kim Shearer, Horst Bunke, and Svetha Venkatesh. "Video Indexing and Similarity Retrieval by Largest Common Subgraph Detection using Decision Trees." Idiap-RR Idiap-RR-15-2000, IDIAP, 0 2000.

[SFS03] M. De Santo, P. Foggia, C. Sansone, and M. Vento. "A large database of graphs and its use for benchmarking graph isomorphism algorithms." *Pattern Recognition Letters*, **24**(8):1067 – 1079, 2003. Graph-based Representations in Pattern Recognition.

[SHK14] Daniel Scharstein, Heiko Hirschmüller, York Kitajima, Greg Krathwohl, Nera Nešić, Xi Wang, and Porter Westling. "High-resolution stereo datasets with subpixel-accurate ground truth." In *Proceedings of the 36th German Conference on Pattern Recognition*, september 2014.

[SM00] Jianbo Shi and J. Malik. "Normalized cuts and image segmentation." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **22**(8):888–905, Aug 2000.

[SSA04] Alberto Sanfeliu, Francesc Serratosa, and Rene Alquezar. "SECOND-ORDER RANDOM GRAPHS FOR MODELING SETS OF ATTRIBUTED GRAPHS AND THEIR APPLICATION TO OBJECT LEARNING AND RECOGNITION." *International Journal of Pattern Recognition and Artificial Intelligence*, **18**(03):375–396, 2004.

[SSD98] K. Siddiqi, A. Shokoufandeh, S. J. Dickenson, and S. W. Zucker. "Shock graphs and shape matching." In *Sixth International Conference on Computer Vision (IEEE Cat. No.98CH36271)*, pp. 222–229, Jan 1998.

[SSJ16] S. Samadi, H. Soltanian-Zadeh, and C. Jutten. "Integrated Analysis of EEG and fMRI Using Sparsity of Spatial Maps." *Brain Topography*, **29**(5):661–678, 2016.

[STC12] M. Song, D. Tao, C. Chen, J. Bu, J. Luo, and C. Zhang. "Probabilistic Exposure Fusion." *IEEE Transactions on Image Processing*, **21**(1):341–357, Jan 2012.

[TC15] Devis Tuia and Gustau Camps-Valls. "Kernel Manifold Alignment for Domain Adaptation." Mar 2015.

[TDC15] Guillaume Tochon, Mauro Dalla Mura, and Jocelyn Chanussot. *Segmentation of Multimodal Images Based on Hierarchies of Partitions*, pp. 241–252. Springer International Publishing, Cham, 2015.

[TL94] Greg Turk and Marc Levoy. "Zippered Polygon Meshes from Range Images." In *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '94, pp. 311–318, New York, NY, USA, 1994. ACM.

[Ull76] J. R. Ullmann. "An Algorithm for Subgraph Isomorphism." *J. ACM*, **23**(1):31–42, January 1976.

[Ume88] S. Umeyama. "An Eigendecomposition Approach to Weighted Graph Matching Problems." *IEEE Transactions on Pattern Analysis Machine Intelligence*, **10**(5):695–703, September 1988.

[VCL15] JT Vogelstein, JM Conroy, V Lyzinski, LJ Podrazik, SG Kratzer, and ET Harley et al. "Fast Approximate Quadratic Programming for Graph Matching." *PLoS ONE*, **10**(4), 2015.

[Ven15] Mario Vento. "A long trip in the charming world of graphs for Pattern Recognition." *Pattern Recognition*, **48**(2):291 – 301, 2015.

[W55] Kuhn H. W. "The Hungarian method for the assignment problem." *Naval Research Logistics Quarterly*, **2**(12):83–97, 1955.

[WJR08] Pakaket Wattuya, Xiaoyi Jiang, and Kai Rothaus. *Combination of Multiple Segmentations by a Random Walker Approach*, pp. 214–223. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[WM11] Chang Wang and Sridhar Mahadevan. "Heterogeneous Domain Adaptation Using Manifold Alignment." In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence*, 2011.

[WMB14] J. T. Woodworth, G. O. Mohler, A. L. Bertozzi, and P. J. Brantingham. "Non-local crime density estimation incorporating housing information." *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, **372**(2028), 2014.

[WS01] Christopher K. I. Williams and Matthias Seeger. "Using the Nyström Method to Speed Up Kernel Machines." In T. K. Leen, T. G. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems 13*, pp. 682–688. MIT Press, 2001.

[Xu04] Guoliang Xu. "Discrete LaplaceBeltrami operators and their convergence." *Computer Aided Geometric Design*, **21**(8):767 – 784, 2004. Geometric Modeling and Processing 2004.

[YC13] Hsiuhan Lexie Yang and Melba M. Crawford. "Learning a Joint Manifold with Global-Local Preservation for Multitemporal Hyperspectral Image Classification." In *IEEE International Geoscience and Remote Sensing Symposium*, 2013.

[YQL17] Xu Yang, Hong Qiao, and Zhi-Yong Liu. "Point correspondence by a new third order graph matching algorithm." *Pattern Recognition*, **65**:108 – 118, 2017.

[YTL12] S. Yu, L. Tranchevent, X. Liu, W. Glanzel, J. A. K. Suykens, B. De Moor, and Y. Moreau. "Optimized Data Fusion for Kernel k-Means Clustering." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **34**(5):1031–1039, May 2012.

[ZBV09] M. Zaslavskiy, F. Bach, and J. P. Vert. "A Path Following Algorithm for the Graph Matching Problem." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **31**(12):2227–2242, Dec 2009.

[ZBV10] Mikhail Zaslavskiy, Francis Bach, and Jean-Philippe Vert. "Many-to-Many Graph Matching: A Continuous Relaxation Approach." In *Machine Learning and Knowledge Discovery in Databases*, pp. 515–530, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[Zhe15] Y. Zheng. "Methodologies for Cross-Domain Data Fusion: An Overview." *IEEE Transactions on Big Data*, **1**(1):16–34, March 2015.