3-7-2017

Compassionately Conservative Normalized Cuts for Image Segmentation

Tyler L. Hayes
tlh6792@rit.edu

Compassionately Conservative Normalized Cuts for Image Segmentation

by

Tyler L. Hayes

A Thesis Submitted in Partial Fulfillment of the Requirements
for the Degree of Master of Science in Applied and Computational Mathematics
School of Mathematical Sciences, College of Science

Rochester Institute of Technology
Rochester, NY
March 7, 2017

Committee Approval:

R·I·T
College of
Science

Nathan Cahill, D.Phil. Date
School of Mathematical Sciences
Thesis Advisor

Elizabeth Cherry, Ph.D. Date
School of Mathematical Sciences
Committee Member

John Hamilton Jr., Ph.D. Date
School of Mathematical Sciences
Committee Member

Sogol Jahanbekam, Ph.D. Date
School of Mathematical Sciences
Committee Member

Christopher Kanan, Ph.D. Date
Center for Imaging Science
Committee Member

Matthew Hoffman, Ph.D. Date
School of Mathematical Sciences
Director of Graduate Programs

Abstract

Image segmentation is a process used in computer vision to partition an image into regions with similar characteristics. One category of image segmentation algorithms is graph-based, where pixels in an image are represented by vertices in a graph and the similarity between pixels is represented by weighted edges. A segmentation of the image can be found by cutting edges between dissimilar groups of pixels in the graph, leaving different clusters or partitions of the data.

A popular graph-based method for segmenting images is the Normalized Cuts (NCuts) algorithm, which quantifies the cost for graph partitioning in a way that biases clusters or segments that are balanced towards having lower values than unbalanced partitionings. This bias is so strong, however, that the NCuts algorithm avoids any singleton partitions, even when vertices are weakly connected to the rest of the graph. For this reason, we propose the Compassionately Conservative Normalized Cut (CCNCut) objective function, which strikes a better compromise between the desire to avoid too many singleton partitions and the notion that all partitions should be balanced.

We demonstrate how CCNCut minimization can be relaxed into the problem of computing Piecewise Flat Embeddings (PFE) and provide an overview of, as well as two efficiency improvements to, the Splitting Orthogonality Constraint (SOC) algorithm previously used to approximate PFE. We then present a new algorithm for computing PFE based on iteratively minimizing a sequence of reweighted Rayleigh quotients (IRRQ) and run a series of experiments to compare CCNCut-based image segmentation via SOC and IRRQ to NCut-based image segmentation on the BSDS500 dataset. Our results indicate that CCNCut-based image segmentation yields more accurate results with respect to ground truth than NCut-based segmentation, and IRRQ is less sensitive to initialization than SOC.

Acknowledgments

First and foremost, I would like to express my sincere gratitude to my advisor, Dr. Nathan Cahill, for his continuous support of my Master's study and research, his patience, wisdom, motivation, and his utmost passion for research. He has helped not only guide me, but push me in all aspects to grow as a researcher, and for this I am extremely grateful.

Moreover, I would like to thank my committee members, Dr. Elizabeth Cherry, Dr. John Hamilton Jr., Dr. Sogol Jahanbekam, and Dr. Christopher Kanan for their immense knowledge, time, helpful comments, and thought-provoking questions. Additionally, I would like to thank Renee Meinhold for her theoretical and experimental contributions detailed in this thesis. I would also like to thank the RIT School of Mathematical Sciences for allowing me to pursue two degrees within their department and for providing me with the invaluable resources to grow as a mathematician.

I must also express my profound gratitude to my family and friends for their unfailing love and support. In particular, I must thank my mom for continually believing in me and encouraging me to chase my passion. Finally, a special thanks to James Arnold whose unconditional love, support, and patience have provided me with constant motivation and encouragement throughout my college career.

Contents

List of Figures

List of Tables

1 Introduction

In computer vision, image segmentation is the process by which a machine automatically segments an image into regions with similar characteristics. These segmented images often have diverse applications ranging from helping doctors identify tumors to serving as a pre-processing step in most modern computer vision and machine learning algorithms. As a pre-processing step, image segmentation is useful in providing initial estimates for tasks such as detection, recognition, and tracking of objects in both images and videos. While automating the task of image segmentation is known to be a challenging problem, automation can alleviate the need for human analysts with the added benefit of improved segmentation accuracy.

Image segmentation algorithms fall into a wide variety of categories, but one of the most popular is graph-based segmentation algorithms. A graph-based image segmentation algorithm is one that treats an image as a graph with vertices representing pixels and weighted edges representing the similarity between pixels. The algorithm then cuts edges between dissimilar groups of pixels, leaving different segments or partitions of the original image. One such algorithm is the Normalized Cuts (NCuts) algorithm [19, 20], which minimizes a normalized sum of weighted edges removed between groups of pixels in a graph. By normalizing the cut cost by the total degrees of each partition, the user eliminates trivial segmentations of the data, where a partition consists of a single pixel.

We argue that in some cases the NCuts algorithm goes too far in keeping partitions *balanced*, which often yields segments that look unnatural. In this thesis, we present a new cut cost which strikes a compromise between the desire to avoid too many singleton partitions and the notion that *all* partitions should be balanced. This new cut cost, coined *Compassionately Conservative Normalized Cuts* (CCNcuts), normalizes the original cut cost by the *square root* of the total degrees of each partition. The CCNCut is more conservative in its normalization scheme than the original NCuts algorithm, which normalized by the total degrees of each partition.

Computing solutions that minimize these various cuts is NP-hard. For NCuts, one way to efficiently approximate the solution is to relax the discrete minimization problem into a continuous one; the relaxed problem can then be transformed into a generalized eigenvalue problem. The relaxed problem is equivalent to the Laplacian Eigenmaps (LE) problem [4], which is a well-known technique for computing representations of data on manifolds. We will show in Chapter 3 that CCNCuts can be relaxed in a similar manner, yielding a continuous relaxation that coincides with the Piecewise Flat Embedding (PFE) problem [27]. The PFE problem is a weighted ℓ_1-minimization subject to a quadratic (orthogonality) constraint.

To compute an approximation to the PFE problem, Yu et al. [27] introduced an iterative scheme based on the Splitting Orthogonality Constraint (SOC) algorithm [13]. While this algorithm demonstrated promising results on the BSDS500 dataset [1], we found that it was not computationally feasible for large images. However, we were able to propose two improvements that are described in Chapter 4 [16]*. The first improvement involves reformulating the original PFE problem to allow multiple linear algebra computations to be performed in parallel. The second improvement utilizes the preconditioned conjugate gradient iterative linear solver to quickly solve a succession of linear least-squares problems.

While the technique for solving PFE based on the SOC algorithm yielded promising results, there are some limitations to this approach. First, the algorithm relies on nested iterations of two algorithms that each have their own set of parameters that must be tuned. Second, due to the nested iterations approach, the algorithm does not strictly enforce the orthogonality constraint in the second loop, which means that the SOC algorithm is not actually approximating a solution to the exact PFE problem. Finally, the SOC algorithm requires an initial estimate of the embedding and can be highly sensitive to initialization.

Due to these limitations, in Chapter 5, we propose an alternative approach for solving the PFE problem inspired by the Iteratively Reweighted Least Squares (IRLS) algorithms commonly used to solve ℓ_1-minimization problems [9]. IRLS algorithms perform ℓ_1-minimization by iteratively solving a succession of weighted least-sqaures (ℓ_2-minimization) problems, with the weights updated at each iteration to reduce the impact of large residual errors. The advantage of IRLS algorithms is that they do not require an initial estimate of the embedding, although they can certainly be provided with one, and in specific cases [9], they have provable convergence guarantees. We show that the solution to the PFE problem can be approximated by iteratively solving a succession of weighted Rayleigh Quotient minimization problems, and thus we term this new algorithm *Iteratively Reweighted Rayleigh Quotient* (IRRQ) minimization.

In Chapter 6, we demonstrate results of minimizing CCNCuts by solving the relaxed (PFE) problem using the IRRQ algorithm on the BSDS500 dataset [1] in a series of experiments. We then provide a detailed comparison of the original SOC algorithm to our proposed IRRQ algorithm. Finally, in Chapter 7 we provide conclusions as well as a list of open questions pertaining to the PFE problem.

*The proposed computational improvements to the SOC algorithm for computing PFE that are described in Chapter 4 of this thesis were done in collaboration with Renee Meinhold.

2 Prior Work and Research Aims

2.1 Introduction to Image Segmentation Methods

Although the main objective of image segmentation is to group pixels into regions with similar characteristics, there are a wide variety of methods to choose from when performing segmentation. At the highest level, these methods could be considered threshold-based, edge-based, region-based, energy-based, or graph-based, with some algorithms combining several of these techniques.

Threshold-based methods are most commonly used for binary or gray-scale image segmentation. In these methods, a threshold is chosen based on some criteria, and pixels are assigned discrete labels based on whether they fall above or below this value. Often, a histogram of pixel values is generated, and large 'peaks' in the histogram counts distinguish the threshold values that should be chosen. One of the most common thresholding algorithms for segmentation is k-means clustering where initial cluster centers can be chosen based on thresholding a histogram of intensity values.

Edge-based methods form segmentation boundaries based on local or global edge and contour information in an image. Three of the most common contour or edge-based segmentation methods are the watershed algorithm [22], snakes [12], and level set methods such as fast marching methods [18]. These methods are useful in locating boundaries on images in an iterative fashion and, as such, are often coined *active contour* algorithms [21]. In Chapter 4, we introduce the global probability of boundary (gPb) method [1] which yields a probability distribution of boundaries over an image.

Region merging techniques for segmentation fall into two main categories: region splitting techniques and region merging techniques. Region splitting, or divisive clustering, techniques treat the initial image as a single cluster, and use either thresholding or cost function optimization to partition an image into smaller clusters. Region merging, or agglomerative clustering, techniques initially treat each pixel as a single cluster, and iteratively merge pixels with similar characteristics into larger clusters.

The similarity between each of these methods is in their ability to produce segmentations with small intra-pixel variability among clusters. If we add the restriction that pixels must be grouped together when they have small relative distances, then we have added a constraint to the segmentation problem. This constraint allows the segmentation problem to be reformulated as an energy function optimization problem, which can be formed using either a variational

formulation or a Markov random field (MRF). In the next section, we introduce the notion of graph-based segmentation, where an image is represented as a graph, and a partitioning of the data is found by optimizing a cost function involving the relationships between vertices in the graph.

2.2 Prior Graph-Based Work

Graph-based image segmentation algorithms have been of interest to the computer vision community due to their ability to model the relationship between pixels within a given neighborhood of an image. In these algorithms, the original image is modeled as an undirected, weighted graph with pixels represented by nodes and the similarity between pixels represented by weighted edges.

Consider an undirected weighted graph $\mathcal{G} = (V, \mathcal{E})$ that we wish to partition into two disjoint subgraphs, $\mathcal{G}_A = (A, \mathcal{E}_A)$ and $\mathbf{G}_B = (B, \mathcal{E}_B)$ where $A \bigcup B = V$. This partitioning can be achieved by removing or *cutting* the edges connecting A and B. The associated cost of this partition of $\mathcal{G}$ is known as the cut cost and is defined as the total weight of the edges that have been removed:

$$\text{Cut}\,(A, B) = \sum_{v_i \in A, v_j \in B} W_{i,j} \ , \tag{2.1}$$

where $V = \{v_1, v_2, ..., v_n\}$ is the vertex set and $\mathbf{W}$ is the weighted adjacency matrix of the graph $\mathcal{G}$.

One method to find an optimal partitioning of $\mathcal{G}$ would be to minimize (2.1) to find the minimum cut. The issue with the minimum cut, however, is that oftentimes the optimal partitioning leaves one subgraph with a single vertex [23]. A demonstration of this can be observed in Figure (1), in which the normalized cut, a cut that will be introduced in (2.3), yields a more desirable partioning of the data than the minimum cut due to its ability to maintain balanced partitions. To determine a more balanced partition, other cut costs have been proposed, including the Ratio Cut (RCut) [6]

$$\text{RCut}\,(A, B) = \frac{\text{Cut}\,(A, B)}{(|A| \cdot |B|)} \ , \tag{2.2}$$

and the Normalized Cut [19, 20]

$$\text{NCut}\,(A, B) = \frac{\text{Cut}\,(A, B)}{\text{Assoc}\,(A, V)} + \frac{\text{Cut}\,(A, B)}{\text{Assoc}\,(B, V)} \ , \tag{2.3}$$

where

$$\text{Assoc}\,(S, V) = \sum_{v_i \in S, v_j \in V} W_{i,j} \tag{2.4}$$

is the total connection of all vertices in S to all vertices in $\mathcal{G}$.

In [20], it is shown that minimizing (2.3) is equivalent to solving the discrete minimization problem:

$$\begin{aligned} \min_{\mathbf{y}} \quad & \frac{\mathbf{y^T}(\mathbf{D}-\mathbf{W})\mathbf{y}}{\mathbf{y^T}\mathbf{D}\mathbf{y}} \\ \text{subject to} \quad & y_i \in \{1, -b\}, i = 1, 2, ..., n, \\ & \mathbf{y^T}\mathbf{D}\mathbf{1} = \mathbf{0} \ , \end{aligned} \tag{2.5}$$

where $\mathbf{D}$ is the diagonal weighted degree matrix of the graph $\mathbf{G}$ defined componentwise by $d_i = D_{i,i} = \sum_j W_{i,j}$, $b = (\sum_{x_i>0} d_i)/(\sum_{x_i<0} d_i)$, and $\mathbf{y} = (\mathbf{1}+\mathbf{x})/2 - b(\mathbf{1}-\mathbf{x})/2$, where $\mathbf{x}$ is an n-dimensional indicator vector where $x_i = 1$ if vertex v_i is in $\mathbf{A}$ and $x_i = -1$ otherwise. It is then shown that if (2.5) is relaxed such that all components of $\mathbf{y} \in \mathbb{R}$, then the solution is equivalent to the generalized eigenvector corresponding to the smallest nontrivial eigenvalue of

$$(\mathbf{D}-\mathbf{W})\mathbf{y} = \lambda \mathbf{D}\mathbf{y} \ . \tag{2.6}$$

The components of the resulting eigenvector can then be clustered using an algorithm, such as k-means, and then assigned a discrete label.

Note that this formulation of the relaxed minimization problem is mathematically equivalent to that of Laplacian Eigenmaps (LE) [4], where it is assumed that the data being analyzed lie on a manifold that is embedded in a high-dimensional space. LE attempts to reduce the dimensionality of the data by seeking a mapping such that data points with similar attributes in the original space retain small distances in the new feature space. If the new feature space has dimension one, then LE can be expressed as the following constrained minimization problem:

$$\begin{aligned} \min_{\mathbf{y}} \quad & \sum_{i,j} W_{i,j} \|y_i - y_j\|_2^2 \\ \text{subject to} \quad & \mathbf{y^T}\mathbf{D}\mathbf{y} = \mathbf{I} \\ & \mathbf{y^T}\mathbf{D}\mathbf{1} = \mathbf{0} \ . \end{aligned} \tag{2.7}$$

The orthogonality constraint $\mathbf{y^T}\mathbf{D}\mathbf{y} = \mathbf{I}$ ensures that the final data embedding is non-trivial, and the balance constraint $\mathbf{y^T}\mathbf{D}\mathbf{1} = \mathbf{0}$ is necessary to avoid the trivial eigenvalue and to ensure that partitions in the final data embedding are *balanced*. A difficulty with using LE, however, is that while pixels with small local distances in the original feature space retain this relationship in the final embedding, pixels in the final data embedding may still be far enough apart that assignment of discrete labels is often ambiguous.

An ideal embedding of data in a new feature space would distribute pixels of the same region tightly around a single point, while pushing pixels of different classes apart to eliminate the ambiguity of cluster boundaries in the new feature space. The Piecewise Flat Embedding (PFE) [27] was proposed to achieve this goal and does so by modifying the LE objective function

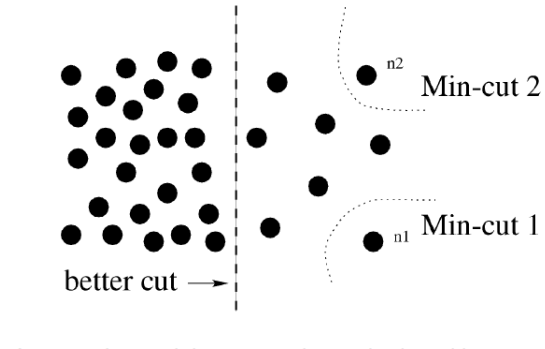


Figure 1: An example of NCuts yielding a better partition of the data than the Minimum Cut [20].

to use an ℓ_1-norm. The use of an ℓ_1-norm in minimization promotes sparse solutions to the segmentation problem, which makes the assignment of discrete labels to pixels a trivial process [27]. The PFE problem can be expressed as the constrained minimization problem:

$$\begin{aligned} \min_{\mathbf{y}} \quad & \sum_{i,j} W_{i,j} \|y_i - y_j\|_1 \\ \text{subject to} \quad & \mathbf{y^T D y} = \mathbf{I} \ . \end{aligned} \tag{2.8}$$

Incorporating the ℓ_1-norm promotes sparse solutions such that pixels in the new feature space that belong to the same class have distances close to zero, while pixels belonging to different classes have much larger distances. This makes segmentation in the new feature space a straightforward process.

In addition to their introduction of the PFE problem [27], Yu et al. also introduced a two-stage numerical approach to solving (2.8), which we will further outline in Chapter 4. Although this two-stage approach yielded promising results on a publicly available dataset, the method does not scale well to large images, requires an initial estimate of the data embedding, and, due to its two-stage nature, has two different sets of parameters that must be tuned at each stage.

2.3 Research Aims

The main goals of this thesis are (1) introducing a new cut cost that, when relaxed, yields the PFE problem, (2) proposing computational improvements made to the two-stage numerical approach introduced in [27], (3) introducing a new algorithm for computing the PFE, and (4) illustrating how image segmentation performed with our new cut cost outperforms the SOC algorithm proposed in [27].

In Chapter 3, we introduce our new cost that seeks to strike a balance between the minimum

cut [23] and the normalized cut [19, 20]. While our new cut cost allows for singleton partitions as in [23], the new cut cost still maintains a notion of normalization, albeit to a lesser extent than the original normalized cut, that significantly reduces the number of singleton partitions in the final pixel labeling.

In Chapter 4, we demonstrate two computational improvements made to the algorithm introduced in [27] to allow the method to scale to larger images. We use these computational improvements to generate our own MATLAB implementation of the algorithm to reproduce the results generated in [27] and to utilize in our own experiments.

In Chapter 5, we introduce our new algorithm for solving the PFE problem. The inspiration for our algorithm came from the Iteratively Reweighted Least Squares (IRLS) algorithms typically used for ℓ_1-minimization. The advantages of the IRLS algorithms are that they do not require an initial estimate of the data embedding and, in special cases [9], have provable linear convergence guarantees. Our algorithm seeks to overcome the shortcomings of the two-stage numerical approach presented in [27] by only requiring the tuning of a single hyperparameter and ensuring convergence to the global minimum independent of embedding initialization.

In Chapter 6, we illustrate the use of CCNCuts for image segmentation on a publicly available database of images. We then compare the results from the SOC algorithm to our proposed algorithm and discuss the benefits and limitations of each algorithm.

3 Compassionately Conservative Normalized Cuts (CCNCuts)

In this Chapter, we introduce the Compassionately Conservative Normalized Cut (CCNCut), which provides an alternative normalization to NCut. When minimized, CCNCut yields graph partitions in such a way that few singleton partitions are permitted, but the notion of *balanced* partitions still exists. Consider the example presented in Figure (2), where we wish to partition the toy graph into three subgraphs. This toy graph (2a) contains 15 vertices and 19 edges; all but two of the edges have unit weight, and the two indicated edges have weight $\alpha \in (0, 1)$. For each cost function, the three-way partitioning of minimum cost depends on whether α falls above or below some critical value α^*. As expected, when the three-way Cut cost is minimized, the resulting partitions are heavily unbalanced, as shown in (2b).

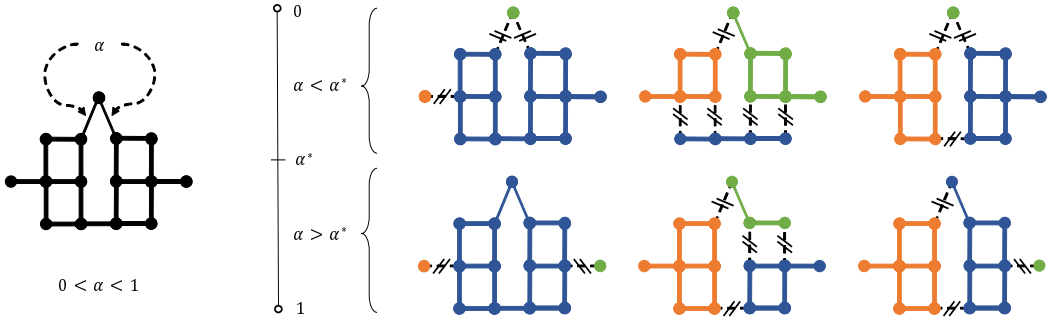
While the three-way NCut cost (the k-way NCut cost is defined in [25]) yields more balanced partitions shown in (2c), the Cut costs of these partitionings are relatively high and these partitions do not necessarily look "natural" from a gestalt sense. We argue that the partitions demonstrated in (2d) have a more "natural" look than those in (2b), striking a compromise between the partitionings resulting from the Minimum Cut (2b) and the Normalized Cut (2c).

The partition depicted in (2d) is our proposed Compassionately Conservative Normalized Cut, which differs from NCuts in its normalization by the *square root* of the total degrees of each partition. This normalization is more conservative than the original NCuts formulation, allowing us the flexibility to obtain singleton partitions, while also maintaining a notion of balanced partitions. For natural imagery, this flexibility allows more realistic partitions of the data to be obtained.

3.1 Definition of the CCNCut

Consider an undirected weighted graph $\mathcal{G} = (V, \mathcal{E})$ that we we wish to partition into k disjoint subgraphs, $\mathcal{G}_i = (V_i, \mathcal{E}_i)$, $i = 1, 2, \ldots, k$, where $\bigcup_{i=1}^{k} V_i = V$. $\mathcal{G}$ can be partitioned by removing edges connecting each of the subgraphs to every other subgraph. To find an optimal partitioning of $\mathcal{G}$, we must define and optimize a partitioning cost. A standard partitioning cost that is analogous to the minimum cut cost is the *multiway cut cost*, defined as the total weight of the edges that have been removed:

$$\text{Cut}(V_1, \ldots, V_k) = \tfrac{1}{2} \sum_{\ell=1}^{k} \text{Cut}(V_\ell, V \backslash V_\ell) \ , \tag{3.1}$$



(a) Toy Graph (b) arg min Cut (c) arg min NCut (d) arg min CCN-Cut

Figure 2: (a) A graph we wish to partition into three subgraphs; all edges have unit weight except for the two edges with weight $\alpha \in (0, 1)$. Partitioning solutions differ based on whether α falls above or below a critical value α^*. (b) Minimizing the 3-way cut (Cut) yields configurations in which two single vertices are removed. (c) Minimizing the 3-way Normalized Cut (NCut) avoids singleton subgraphs, but forces cuts between strongly-connected vertices in order to yield "balanced" solutions. (d) Minimizing the 3-way *Compassionately Conservative Normalized Cut* (CCNCut) enables singleton partitions where vertices are weakly connected to the rest of the graph, but retains balance between the remaining partitions.

where the pairwise cut cost is defined by:

$$\text{Cut}(A, B) = \sum_{v_i \in A, v_j \in B} w_{i,j} \ , \tag{3.2}$$

the vertex set $V = \{v_1, v_2, \cdots, v_n\}$, and $\mathbf{W}$ is the weighted adjacency matrix (or *affinity* matrix) of $\mathcal{G}$ containing only positive weights. Since the graph $\mathcal{G}$ is undirected, $\mathbf{W}$ will be symmetric.

Minimizing this Cut cost is undesirable, however, as it can yield partitionings in which one or more of the subgraphs contain a single vertex [23]. More balanced partitions emerge if the pairwise cut costs are normalized by the total degrees (volumes) of the subgraphs. Yu and Shi [25] define a multiway generalization of the NCut cost [19, 20] by:

$$\text{NCut}(V_1, \ldots, V_k) = \tfrac{1}{2} \sum_{\ell=1}^{k} \frac{\text{Cut}(V_\ell, V \backslash V_\ell)}{\text{Vol}(V_\ell)} \ , \tag{3.3}$$

where $\text{Vol}(V_\ell) = \sum_{v_j \in V_\ell} d_j$, and $d_j = \sum_m w_{j,m}$ is the degree of vertex v_j.

Instead of normalizing the pairwise cut costs by the volumes of each subgraph, we propose normalizing by the square roots of the volumes:

$$\text{CCNCut}(V_1, \ldots, V_k) = \tfrac{1}{2} \sum_{\ell=1}^{k} \frac{\text{Cut}(V_\ell, V \backslash V_\ell)}{\sqrt{\text{Vol}(V_\ell)}} \ . \tag{3.4}$$

In simple terms, minimizing the CCNCut cost (3.4) should still yield partitions that are more balanced than when minimizing (3.1), while better preserving strongly-connected subgraphs than when minimizing (3.3).

By defining an $n \times k$ indicator matrix $\mathbf{X}$ such that $X_{i,j} = 1$ if $v_i \in V_j$ and $X_{i,j} = 0$ otherwise, it is straightforward to see how (3.3)–(3.4) can be reformulated. If $\mathbf{x}_i$ is the i^{th} column of $\mathbf{X}$, then $\text{Vol}(V_i)$ can be written in terms of the degree matrix $\mathbf{D} = \text{diag}(\mathbf{d})$ as $\mathbf{x}_i^{\mathbf{T}}\mathbf{D}\mathbf{x}_i$, and the pairwise cut cost between V_i and $V \backslash V_i$ can be written as $\text{Cut}(V_i, V \backslash V_i) = \mathbf{x}_i^{\mathbf{T}}\mathbf{W}\left(\mathbf{1} - \mathbf{x}_i\right) = \mathbf{x}_i^{\mathbf{T}}\mathbf{d} - \mathbf{x}_i^{\mathbf{T}}\mathbf{W}\mathbf{x}_i = \mathbf{x}_i^{\mathbf{T}}\mathbf{D}\mathbf{x}_i - \mathbf{x}_i^{\mathbf{T}}\mathbf{W}\mathbf{x}_i = \mathbf{x}_i^{\mathbf{T}}\mathbf{L}\mathbf{x}_i$, where $\mathbf{L} = \mathbf{D} - \mathbf{W}$ is the graph Laplacian matrix. This allows us to express (3.3)–(3.4) as:

$$\text{NCut}(V_1, \ldots, V_k) = \tfrac{1}{2}\sum_{\ell=1}^{k} \frac{\mathbf{x}_\ell^{\mathbf{T}}(\mathbf{D} - \mathbf{W})\,\mathbf{x}_\ell}{\mathbf{x}_\ell^{\mathbf{T}}\mathbf{D}\mathbf{x}_\ell} \ , \tag{3.5}$$

$$\text{CCNCut}(V_1, \ldots, V_k) = \tfrac{1}{2}\sum_{\ell=1}^{k} \frac{\mathbf{x}_\ell^{\mathbf{T}}(\mathbf{D} - \mathbf{W})\,\mathbf{x}_\ell}{\sqrt{\mathbf{x}_\ell^{\mathbf{T}}\mathbf{D}\mathbf{x}_\ell}} \ . \tag{3.6}$$

Minimizing (3.3)–(3.4) is equivalent to minimizing (3.5)–(3.6) subject to the constraint that $\mathbf{X}^{\mathbf{T}}\mathbf{X}$ is positive diagonal, which ensures that none of the V_i's will collapse to the empty set.

3.2 Relaxation of the CCNCut

Minimizing (3.6) is NP-hard, as is minimizing (3.5). However, relaxing (3.6) yields a problem whose solution can be efficiently approximated. To relax the CCNCut, we use an argument similar to Yu and Shi [25] in their relaxation of the NCut. From [25], we see that if $\mathbf{Y} = \mathbf{X}\left(\mathbf{X}^{\mathbf{T}}\mathbf{D}\mathbf{X}\right)^{-1/2}$, then $\mathbf{Y}^{\mathbf{T}}\mathbf{D}\mathbf{Y} = \mathbf{I}$ and (3.5) is equivalent to $\text{tr}\left(\mathbf{Y}^{\mathbf{T}}\mathbf{L}\mathbf{Y}\right)$. Hence, the solution to minimizing a relaxed version of (3.5) is $\tilde{\mathbf{Y}} = \mathbf{U}\mathbf{Q}$, where $\mathbf{U}$ is the $n \times k$ matrix whose columns are the orthonormal eigenvectors $\mathbf{u}_2$, …, $\mathbf{u}_{k+1}$ corresponding to the smallest nontrivial eigenvalues of $\mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2}$, and $\mathbf{Q}$ is an arbitrary $k \times k$ orthogonal matrix. The optimal solution $\tilde{\mathbf{X}}$ to (3.5) can then be approximated by k-means clustering [17], nonmaximal suppression [25] or Procrustean rounding [28] on $\tilde{\mathbf{Y}}$.

Now define $\hat{\mathbf{y}}_i$ to be the i^{th} row of $\mathbf{Y}$. We then have that the minimization of the relaxed version of (3.5) is equivalent to solving the constrained minimization problem:

$$\begin{aligned} \min_{\mathbf{Y} \in \mathbb{R}^{n \times k}} \quad & \mathcal{J}_2(\mathbf{Y}) := \sum_{i=1}^{n}\sum_{j=1}^{n} w_{i,j}\left\|\hat{\mathbf{y}}_i - \hat{\mathbf{y}}_j\right\|_2^2 \\ \text{subject to:} \quad & \mathbf{Y}^{\mathbf{T}}\mathbf{D}\mathbf{Y} = \mathbf{I} \ , \quad \mathbf{Y}^{\mathbf{T}}\mathbf{D}\mathbf{1} = \mathbf{0} \ , \end{aligned} \tag{3.7}$$

which is identical to the Laplacian Eigenmaps (LE) problem [4] for computing embeddings of data that are assumed to lie on a manifold. Recall that the *balance constraint* $\mathbf{Y}^{\mathbf{T}}\mathbf{D}\mathbf{1} = \mathbf{0}$ is

necessary to avoid eigenvectors of $\mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2}$ corresponding to the trivial eigenvalue.

Turning our attention now to (3.6), if we define $\alpha_\ell = \left(\sum_i d_i x_{i,\ell}^2\right)^{-1/2}$ for $\ell = 1, \ldots, k$, and we note that $\left(\mathbf{X}^\mathbf{T}\mathbf{D}\mathbf{X}\right)^{-1/2} = \mathrm{diag}(\boldsymbol{\alpha})$, we can write:

$$\begin{aligned}
\mathrm{CCNCut}(V_1, \ldots, V_k) &= \sum_{\ell=1}^{k} \frac{\sum_{i,j} w_{i,j} \left(x_{i,\ell} - x_{j,\ell}\right)^2}{\left(\sum_i d_i x_{i,\ell}^2\right)^{1/2}} \\
&= \sum_{\ell=1}^{k} \alpha_\ell \sum_{i,j} w_{i,j} \left|x_{i,\ell} - x_{j,\ell}\right| \\
&= \sum_{i,j} w_{i,j} \left\| \left(\mathbf{X}^\mathbf{T}\mathbf{D}\mathbf{X}\right)^{-1/2} (\hat{\mathbf{x}}_i - \hat{\mathbf{x}}_j) \right\|_1 \\
&= \sum_{i,j} w_{i,j} \left\|\hat{\mathbf{y}}_i - \hat{\mathbf{y}}_j\right\|_1 \ . \qquad (3.8)
\end{aligned}$$

Hence, the relaxation of (3.6) is obtained by dropping the condition that $\mathbf{Y} = \mathbf{X}\left(\mathbf{X}^\mathbf{T}\mathbf{D}\mathbf{X}\right)^{-1/2}$ and solving the constrained minimization problem:

$$\begin{aligned}
\min_{\mathbf{Y} \in \mathbb{R}^{n \times k}} \quad & \mathcal{J}_1(\mathbf{Y}) := \sum_{i=1}^{n} \sum_{j=1}^{n} w_{i,j} \left\|\hat{\mathbf{y}}_i - \hat{\mathbf{y}}_j\right\|_1 \qquad (3.9) \\
\text{subject to:} \quad & \mathbf{Y}^\mathbf{T}\mathbf{D}\mathbf{Y} = \mathbf{I} \ .
\end{aligned}$$

Interestingly enough, (3.9) is exactly the Piecewise Flat Embedding problem [27], which, in contrast with Laplacian Eigenmaps, yields embeddings in which the data are naturally clustered since it promotes sparsity in the differences between rows of $\mathbf{Y}$.

4 Two-Stage Numerical Approach to Solving the Piecewise Flat Embedding (PFE) Problem

In this Chapter, we present the two-stage numerical approach outlined in [27] for approximating a solution to the PFE problem. We then explore the limitations of this two-stage approach and present two improvements to allow the method to be more computationally feasible on larger images*.

4.1 Overview of the Two-Stage Numerical Approach

Recall the PFE problem outlined in (3.9) where, provided n data points $\mathcal{X} = \{x_1, \ldots, x_n\}$ in $\mathbb{R}^d$, we wish to transform the data to a new k-dimensional space where $\mathbf{Y}$ is our new $n \times k$ dimensional embedding representing the transformed data. Minimizing (3.9) is more difficult than minimizing (3.7) since (3.9) includes both an orthogonality constraint, as well as the minimization of a term involving the ℓ_1-norm. Due to the orthogonality constraint, (3.9) cannot be solved analytically; hence, Yu et al. [27] introduce a method that utilizes two nested algorithms to effectively minimize (3.9) subject to an orthogonality constraint. Since (3.9) is convex, they apply the Splitting Orthogonality Constraint (SOC) algorithm proposed in [13] to handle the orthogonality constraint, while also applying the Split Bregman algorithm proposed in [11] to handle the ℓ_1 minimization.

To handle the orthogonality constraint, the SOC algorithm defines $\mathbf{P} = \mathbf{D}^{1/2}\mathbf{Y}$ and rewrites (3.9) as

$$\begin{aligned} \min_{\mathbf{Y} \in \mathbb{R}^{n \times k}} \quad & \mathcal{J}_1(\mathbf{Y}) := \sum_{i=1}^{n} \sum_{j=1}^{n} w_{i,j} \left\| \hat{\mathbf{y}}_i - \hat{\mathbf{y}}_j \right\|_1 \\ \text{subject to:} \quad & \mathbf{D^{1/2}Y} = \mathbf{P} \ , \quad \mathbf{P^T P} = \mathbf{I} \ . \end{aligned} \tag{4.1}$$

The SOC algorithm (Algorithm (1)) then approximates a solution to (4.1) by using a succession of Bregman iterations [5].

While the update to $\mathbf{P}^{(m+1)}$ in step (b) of the SOC algorithm has a closed form solution described in [27], we note that the update to $\mathbf{Y}^{(m+1)}$ in step (a) still includes a term involving the ℓ_1-norm. To obtain a solution to this ℓ_1-norm minimization problem, Yu et al. [27] utilize the Split Bregman algorithm [11] to transform the original problem into a series of differentiable unconstrained convex optimization problems.

To update $\mathbf{Y}^{(m+1)}$ using the Split Bregman algorithm, Yu. et al. [27] concatenate the columns of

*The computational improvements described in this chapter were published in [16].

Algorithm 1 SOC Algorithm for Approximating (4.1)

procedure SOC($\mathbf{W}$, $\mathbf{Y}^{(0)}$)

$\mathbf{D} := \text{diag}(\mathbf{W}1)$, $m := 0$, $\mathbf{P}^{(0)} := \mathbf{D}^{1/2}\mathbf{Y}^{(0)}$, $\mathbf{B}^{(0)} := \mathbf{0}_{n\times k}$

repeat

(a) $\mathbf{Y}^{(m+1)} := \arg\min_{\mathbf{Y}} \left(\sum_{i,j} w_{i,j} \left\| \hat{\mathbf{y}}_i - \hat{\mathbf{y}}_j \right\|_1 + \frac{r}{2} \left\| \mathbf{D}^{1/2}\mathbf{Y} - \mathbf{P}^{(m)} + \mathbf{B}^{(m)} \right\|_2^2 \right)$

(b) $\mathbf{P}^{(m+1)} := \arg\min_{\mathbf{P}} \left\| \mathbf{P} - \left(\mathbf{D}^{1/2}\mathbf{Y}^{(m+1)} + \mathbf{B}^{(m)} \right) \right\|_2^2$
s.t. $\mathbf{P}^{\mathbf{T}}\mathbf{P} = \mathbf{I}$

(c) $\mathbf{B}^{(m+1)} := \mathbf{B}^{(m)} + \mathbf{D}^{1/2}\mathbf{Y}^{(m+1)} - \mathbf{P}^{(m+1)}$

(d) $m \leftarrow m + 1$

until convergence

return $\mathbf{Y}^{(m)}$

end procedure

the matrices $\mathbf{Y}^{(m)}$, $\mathbf{P}^{(m)}$, and $\mathbf{B}^{(m)}$ into the vectors $\mathbf{Y}_v^{(m)}$, $\mathbf{P}_v^{(m)}$, and $\mathbf{B}_v^{(m)}$ respectively. They next define $\mathbf{M} = \{M_{i,j}\}$ to be an $n(n-1)/2 \times n$ dimensional sparse matrix with $M_{m,i} = w_{i,j}$ and $M_{m,j} = -w_{i,j}$ for two points x_i and x_j which form the m-th pair. Finally, a $(kn(n-1)/2) \times (kn)$ matrix $\mathbf{L}$ and a $(kn) \times (kn)$ matrix $\tilde{\mathbf{D}}$ are defined as follows:

$$\mathbf{L} := \mathbf{I}_{k\times k} \otimes \mathbf{M} \ , \tag{4.2}$$

$$\tilde{\mathbf{D}} := \mathbf{I}_{k\times k} \otimes \mathbf{D} \ , \tag{4.3}$$

where $\otimes$ denotes the Kronecker product. Using these new definitions, step (a) of the SOC algorithm is rewritten as

$$\mathbf{Y}_v^{(m+1)} := \arg\min_{\mathbf{Y}_v} \|\mathbf{L}\mathbf{Y}_v\|_1 + \frac{r}{2} \left\| \tilde{\mathbf{D}}^{1/2}\mathbf{Y}_v - \mathbf{P}_v^{(m)} + \mathbf{B}_v^{(m)} \right\|_2^2 \ , \tag{4.4}$$

which can be solved using the Split Bregman algorithm [11] outlined in Algorithm (2). Note that a least-squares problem is formulated in step (a) of Algorithm (2), which we will minimize using its normal equations.

As in [27], we propose to perform these two algorithms in a two-stage approach. In the first stage, we implement the full numerical solution by using nested Bregman iterations and the SOC algorithm. This solves the ℓ_1-minimization problem while strictly enforcing the orthogonality constraint in the outer loop. In the second stage, we relax the orthogonality constraint and only execute the Bregman iterations to minimize the objective function involving the term with the ℓ_1-norm. This stage utilizes the Split Bregman algorithm and allows the ℓ_1-term to reach lower energy levels. A flowchart outlining this two-stage approach is presented in Figure (3).

Algorithm 2 Split Bregman Algorithm for Approximating (4.4)

procedure SPLITBREGMAN($\mathbf{M}$, $\mathbf{D}$, $\mathbf{P}^{(m)}$, $\mathbf{B}^{(m)}$)

Construct $\mathbf{L}$ and $\tilde{\mathbf{D}}$ from (4.2)–(4.3).

$\ell := 0$, $\mathbf{b}^{\ell} := \mathbf{0}_{(kn(n-1)/2)\times 1}$, $\mathbf{d}^{\ell} := \mathbf{0}_{(kn(n-1)/2)\times 1}$

while $\left\|\mathbf{Y}_v^{(m,\ell+1)} - \mathbf{Y}_v^{(m,\ell)}\right\| \leq \epsilon$ **do**

(a) $\mathbf{Y}_v^{(m,\ell+1)} := \arg\min_{\mathbf{Y}_v} \left(\frac{\lambda}{2} \left\| \mathbf{L}\mathbf{Y}_v + \mathbf{b}^{\ell} - \mathbf{d}^{\ell} \right\|_2^2 + \frac{r}{2} \left\| \tilde{\mathbf{D}}^{1/2}\mathbf{Y}_v - \mathbf{P}_v^{(m)} + \mathbf{B}_v^{(m)} \right\|_2^2 \right)$

(b) $\mathbf{d}^{\ell+1} := \text{Shrink}\left(\mathbf{L}\mathbf{Y}_v^{(m,\ell+1)} + \mathbf{b}^{\ell}, 1/\lambda \right)$

(c) $\mathbf{b}^{\ell+1} := \mathbf{b}^{\ell} + \mathbf{L}\mathbf{Y}_v^{(m,\ell+1)} - \mathbf{d}^{\ell+1}$

(d) $\ell \leftarrow \ell + 1$

end while

return $\mathbf{Y}^{(m,\ell)}$

end procedure

procedure SHRINK($\mathbf{y}$, δ)

$z_i = \text{sign}(y_i) \cdot \max(|y_i| - \delta, 0), i = 1, \ldots, \text{numel}(\mathbf{z})$

return $\mathbf{z}$

end procedure

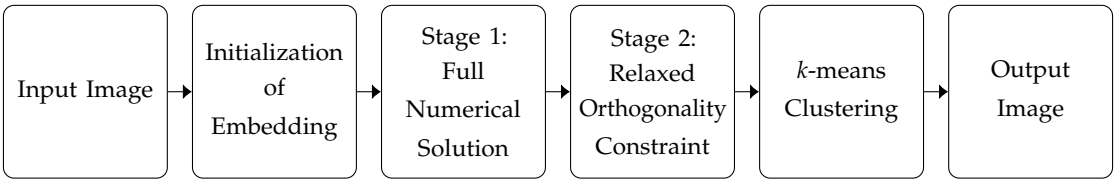


Figure 3: The two-stage approach using the SOC algorithm [13] paired with the Split Bregman algorithm [11] to solve the ℓ_1-minimization problem in (3.9) subject to an orthogonality constraint. The PFE is a result of the combination of stages 1 and 2.

4.2 Efficient Computation of the PFE Problem

Consider a 400×600 pixel image that we wish to partition into 20 clusters using the two-stage approach outlined in [27]. We construct a graph of the image with each node corresponding to a single pixel and use the 4-nearest neighbors algorithm to assign weights to edges between nodes that are within the 4-pixel neighborhood of one another. Based on this graph construction and the use of double-precision floating-point values for our computations, our sparse weighted adjacency matrix $\mathbf{W}$ and our data embedding $\mathbf{Y}$ would require approximately 7.7MB and 38.4MB of storage respectively. To apply the Split Bregman algorithm, we must compute the sparse matrices $\mathbf{M}$, $\mathbf{L}$, and $\tilde{\mathbf{D}}$, where $\mathbf{M}$ will be $((400 \cdot 600)(400 \cdot 600 - 1)/2) \times (400 \cdot 600) = 28,799,880,000 \times 240,000$ with $1,920,000$ non-zero entries (15.36MB), $\mathbf{L}$ will

be $(20 \cdot (400 \cdot 600)\,(400 \cdot 600 - 1)\,/2) \times (20 \cdot (400 \cdot 600)) = 575,997,600,000 \times 4,800,000$ with $38,400,000$ non-zero entries (307.2MB), and $\tilde{\mathbf{D}}$ will be $(20 \cdot (400 \cdot 600)) \times (20 \cdot (400 \cdot 600)) = 4,800,000 \times 4,800,000$ with $4,800,000$ non-zero entries located on the main diagonal (38.4MB).

While the image in our example is relatively small and we use a modest neighborhood structure to construct our graph, the storage space required for these matrices is still quite large. As such, the task of multiplying $\mathbf{L}$ by $\mathbf{Y}_v$ in step (a) of the Split Bregman algorithm will be computationally intensive for solving the PFE problem. Recall that the solution to step (a) of Algorithm (2) requires the solution to the following normal equations:

$$\left[\tfrac{\lambda}{2}\mathbf{L}^{\mathbf{T}}\mathbf{L} + \tfrac{r}{2}\tilde{\mathbf{D}}\right]\mathbf{Y}^{(m,\ell+1)} = \tfrac{\lambda}{2}\mathbf{L}^{\mathbf{T}}\mathbf{q_1} + \tfrac{r}{2}\tilde{\mathbf{D}}^{1/2}\mathbf{q}_2 \ , \tag{4.5}$$

where $\mathbf{q_1} = \mathbf{d}^{\ell} - \mathbf{b}^{\ell}$ and $\mathbf{q_2} = \mathbf{B}_v^{(m)} - \mathbf{P}_v^{(m)}$. Solving (4.5) by inverting $\frac{\lambda}{2}\mathbf{L}^{\mathbf{T}}\mathbf{L} + \frac{r}{2}\tilde{\mathbf{D}}$ would be unwise as it would require the formation of a large, dense matrix ($4,800,000 \times 4,800,000$ matrix (184.32TB) in our example). Attempting to use a direct solver for (4.5) would be infeasible due to the large amount of memory required since $\frac{\lambda}{2}\mathbf{L}^{\mathbf{T}}\mathbf{L} + \frac{r}{2}\tilde{\mathbf{D}}$ is sparse and banded, but its bands are far from the main diagonal.

While downsampling the image would preserve storage space, we would sacrifice clarity in the image, possibly resulting in a poor partitioning of the data. Thus, we present the following modifications to the two-stage approach outlined in [27] for approximating a solution to the PFE problem.

First, we define the function $\texttt{vec}\colon \mathbb{R}^{s\times t} \to \mathbb{R}^{st}$ that "unwraps" a matrix $\mathbf{Z} = \{\mathbf{z_1}, \ldots, \mathbf{z_t}\}$ into the vector $\texttt{vec}(\mathbf{Z}) = \left[\mathbf{z_1^T}, \ldots, \mathbf{z_t^T}\right]^{\mathbf{T}}$. This allows us to write $\mathbf{Y}_v^{(m)} = \texttt{vec}\left(\mathbf{Y}^{(m)}\right)$, $\mathbf{P}_v^{(m)} = \texttt{vec}\left(\mathbf{P}^{(m)}\right)$, and $\mathbf{B}_v^{(m)} = \texttt{vec}\left(\mathbf{B}^{(m)}\right)$ and thus we have:

$$\mathbf{L}\mathbf{Y}_v = \texttt{vec}\left(\mathbf{M}\mathbf{Y}\right) \ , \tag{4.6}$$

$$\tilde{\mathbf{D}}^{1/2}\mathbf{Y}_v = \texttt{vec}\left(\mathbf{D}^{1/2}\mathbf{Y}\right) \ . \tag{4.7}$$

By using the vec notation, equation (4.6) allows us to rewrite steps (b)–(c) of the Split Bregman algorithm as:

$$\mathbf{d}^{\ell+1} = \text{Shrink}\left(\texttt{vec}\left(\mathbf{M}\mathbf{Y}^{(m,\ell+1)}\right) + \mathbf{b}^{\ell}, 1/\lambda\right) \ , \tag{4.8}$$

where Shrink is as defined in Algorithm (2)

$$\mathbf{b}^{\ell+1} = \mathbf{b}^{\ell} + \texttt{vec}\left(\mathbf{M}\mathbf{Y}^{(m,\ell+1)}\right) - \mathbf{d}^{\ell+1} \ , \tag{4.9}$$

which reduces the computation time for steps (b)–(c) by a factor of k. Moreover, we can write:

$$\mathbf{L}^{\mathbf{T}}\mathbf{L}\mathbf{Y}_v = \texttt{vec}\left(\mathbf{M}^{\mathbf{T}}\mathbf{M}\mathbf{Y}\right) \ , \tag{4.10}$$

$$\tilde{\mathbf{D}}\mathbf{Y}_v = \texttt{vec}\left(\mathbf{D}\mathbf{Y}\right) \ , \tag{4.11}$$

and thus, we rewrite (4.5) as:

$$\left[\tfrac{\lambda}{2}\mathbf{M}^{\mathbf{T}}\mathbf{M} + \tfrac{r}{2}\mathbf{D}\right]\mathbf{Y}^{(m,\ell+1)} = \tfrac{\lambda}{2}\mathbf{M}^{\mathbf{T}}\mathbf{Q_1} + \tfrac{r}{2}\mathbf{D}^{1/2}\mathbf{Q}_2 \ , \tag{4.12}$$

where $\texttt{vec}(\mathbf{Q_1}) = \mathbf{d}^{\ell} - \mathbf{b}^{\ell}$ and $\mathbf{Q_2} = \mathbf{B}^{(m)} - \mathbf{P}^{(m)}$. By replacing steps (a), (b), and (c) in the Split Bregman algorithm with (4.12), (4.8), and (4.9) respectively, we arrive at our first computational efficiency improvement to the two-stage approach outlined in [27]. We now *simultaneously* solve the k different $n \times n$ systems of equations in (4.12) instead of solving the single $(kn) \times (kn)$ system of equations in (4.5), which eliminates the necessity of computing the large matrices $\mathbf{L}$ and $\tilde{\mathbf{D}}$.

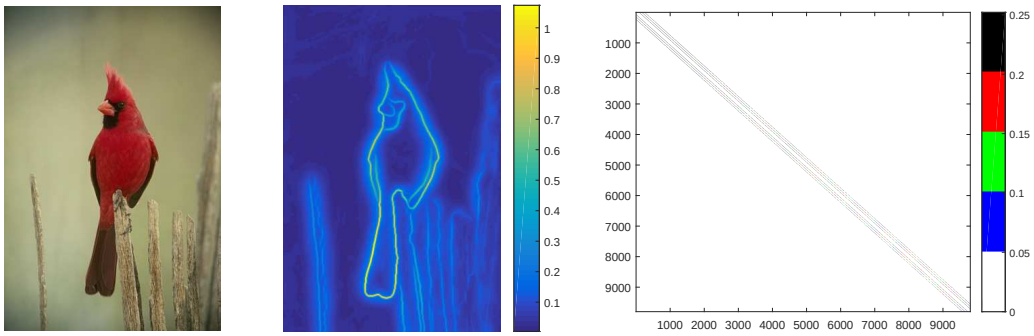
Although the formulation in (4.12) preserves memory by a factor of k when compared to (4.5), we found experimentally that (4.12) is still infeasible to solve by matrix inversion or by a direct solver. Thus, our second computational improvement utilizes the preconditioned conjugate gradient (PCG) method [3] with an incomplete Cholesky preconditioner to approximate a solution to (4.12). This solution completes the approximation to step (a) of the Split Bregman algorithm.

4.3 Two-Stage Approach for Segmentation

To investigate the consistency of our efficient PFE implementation on an image segmentation task, we replicate two of the experiments in [27] that utilize the 200 test images in the BSDS500 dataset [1], each of which has a variety of manually-labeled segmentations with different numbers of segments that can be used as ground truth. We follow the two-stage approach outlined in [27]. A first stage is run with ten outer iterations, five inner iterations, and hyperparameters $\lambda = 10000$ and $r = 100$, and a second stage with a maximum of 100 iterations, $\lambda = 10000$, and $r = 10$, where λ and r are used in Algorithms (1) and (2). (These are the same parameter choices as in [27]). To approximate the PFE, we use a Gaussian Mixture Model (GMM) initialization and k-means clustering on the final embedding described in [27].

We test our efficient PFE implementation using two different methods of graph construction as in [27]. The first is based on the nearest-neighbor construction proposed in [4]. Given two pixels, x_i and x_j, that are within the 4-nearest neighbors of one another, we place an edge between them with a weight defined by the heat kernel:

$$w_{i,j} = \exp\left(-\frac{\|x_i - x_j\|^2}{\sigma}\right) \ , \tag{4.13}$$



(a) Original Image (b) gPb Signal (c) Affinity Matrix

Figure 4: The gPb signal and resulting affinity matrix from an image in the BSDS500 dataset at one-quarter the original resolution. The resized image resolution is 121×81 pixels in this example.

where σ is a user-defined parameter. If x_i and x_j are far from one another, the exponential function will be raised to a large, negative value, forcing the weight to be small. If x_i and x_j are close to one another, the exponential function will be raised to a small, negative value, forcing the weight to be large. We refer to this graph construction as the "intensity graph construction" for the remainder of this thesis, since the weights between neighboring pixels are based on their differences in pixel intensity.

The second graph construction is based on the globalized probability of boundary (gPb) [1]. The gPb measures the differences in features between two halves of a disc that is divided at a specific angle. These measured differences allow us to predict the posterior probability of a boundary at every pixel in an image. To compute the gPb graph construction, we utilize the code provided with the BSDS500 dataset [1]. The code first places a vertex at each pixel in the original image and then computes the gPb at each pixel. The maximum gPb at eight different angles is determined and an edge is placed between pixel x_i and x_j with weight

$$w_{i,j} = \exp\left(\frac{-\max\left\{\text{gPb}\left(\mathbf{p}_{i,j}\right)\right\}}{\rho}\right) , \tag{4.14}$$

when $\|\mathbf{p}_{i,j}\| \leq r$ and $w_{i,j} = 0$ otherwise, where $\mathbf{p}_{i,j}$ is the line segment connecting pixel x_i to x_j, ρ is a constant and r is a user-defined threshold. We refer to this graph construction as the "gPb graph construction" for the remainder of this thesis. Figure (4) demonstrates an example of the gPb signal from an image and the resulting affinity matrix.

4.4 Performance Comparison Between Algorithms Against Ground-Truth

To quantitatively compare the PFE implementations, we evaluate the segmentations with respect to ground-truth using the three criteria described in [1] and used in [27]: Segmentation covering, Probabilistic Rand Index (PRI), and Variation of Information (VI). Segmentation covering is the measure of overlap between two regions, PRI compares the compatibility of two regions, and VI measures the distance between two regions based on their average conditional entropy [1]. That is, for a ground truth image S', the covering by a machine segmentation S is defined in [1] as:

$$\mathcal{C}(S' \to S) = \frac{1}{N} \sum_{R \in S} |R| \cdot \max_{R' \in S'} \mathcal{O}(R, R') \ , \tag{4.15}$$

where N is the total number of pixels in the image and the overlap between regions R and R' is defined as

$$\mathcal{O}(R, R') = \frac{|R \cap R'|}{|R \cup R'|} \ . \tag{4.16}$$

The Probabilistic Rand Index for a set $\{G_i\}$ of ground truth segmentations is defined in [1] as

$$\text{PRI}(S, \{G_i\}) = \frac{1}{T} \sum_{i<j} \left[c_{ij} p_{ij} + (1 - c_{ij}) (1 - p_{ij}) \right] \ , \tag{4.17}$$

where T is the total number of pixel pairs in the image, c_{ij} is the event that pixel i and pixel j have the same label, and p_{ij} is the corresponding probability for c_{ij}. Variation of Information is defined in [1] as:

$$\text{VI}(S, S') = H(S) + H(S') - 2I(S, S') \ , \tag{4.18}$$

where H and I represent the entropies and mutual information between two segementations respectively. As a segmentation becomes closer to ground-truth, covering and PRI will increase, while VI will decrease.

Following the strategy outlined in [27], we report results for both a *fixed* scheme, where we run the algorithm repeatedly with k corresponding to each number of segments in the ground-truth and average the performance across multiple runs, and a *dynamic* scheme, where we choose the value of k from $k = 5, 10, 15, 20, 25$ that yields the best performance for a particular image. The results pertaining to both of these schemes using the intensity graph construction are presented in Table (1) and the results using the gPb graph construction are presented in Table (2).

From these tables, we observe that our efficient implementation of PFE yields similar covering and PRI values when compared to the implementation in Yu et al. [27] for both methods of graph construction. We note, however, that for both methods of graph construction, our VI measures were slightly higher than those reported in [27]. If we consider only the gPb graph construction, we found the standard deviation in VI values to be approximately 0.44

Method	Covering		PRI		VI	
	fixed	dynamic	fixed	dynamic	fixed	dynamic
Yu-PFE	**0.46**	**0.52**	**0.77**	**0.79**	**2.21**	**1.91**
Ours	0.42	**0.52**	0.75	**0.79**	2.59	2.20

Table 1: Comparison of Yu et al. implementation of PFE (Yu-PFE) and our efficient PFE method (Ours) on the BSDS5000 dataset using the intensity-based affinity construction. All results were averaged across images and the best results for each performance measure have been highlighted in bold.

Method	Covering		PRI		VI	
	fixed	dynamic	fixed	dynamic	fixed	dynamic
Yu-PFE	**0.45**	**0.56**	**0.78**	**0.81**	**2.26**	**1.77**
Ours	0.43	0.51	0.74	0.78	2.30	2.04

Table 2: Comparison of Yu et al. implementation of PFE (Yu-PFE) and our efficient PFE method (Ours) on the BSDS5000 dataset using the gPb-based affinity construction [1]. All results were averaged across images and the best results for each performance measure have been highlighted in bold.

for both the fixed and dynamic schemes, with similar high deviation values for the intensity graph construction as well. If the implementation in Yu et al. [27] had high standard deviation values as well (which were not reported in their paper), it is likely that the difference in VI values between methods is not statistically significant. We will further explore the statistical significance of each of the three quantitative measures outlined here in Chapter 6.

In Figure (5), we show a plot of the computation time (in seconds), as a function of cluster number, required to compute the PFE for the 200 test images. From this figure we can see that using the intensity-based graph construction, our implementation takes anywhere between 10 seconds and 15 minutes to compute the PFE, while the gPb-based graph construction takes anywhere between 1.5 and 110 minutes to compute the PFE, depending on how many clusters are desired. The large amount of time required for the gPb-based graph construction could possibly be a result of the structure of the adjacency matrix, and could be investigated in future work. While Yu et al. [27] report that their implementation requires approximately 15 minutes to compute the PFE per image, they have not yet released code, prohibiting us from performing a direct comparison.

In this Chapter, we presented the two-stage approach outlined in [27] for approximating a solution to the PFE problem. We then outlined the computational improvements that we made to the approach and demonstrated the results of our efficient implementation based on two experiments evaluated on the BSDS500 dataset [1]. We note that at this time, code for the Yu et

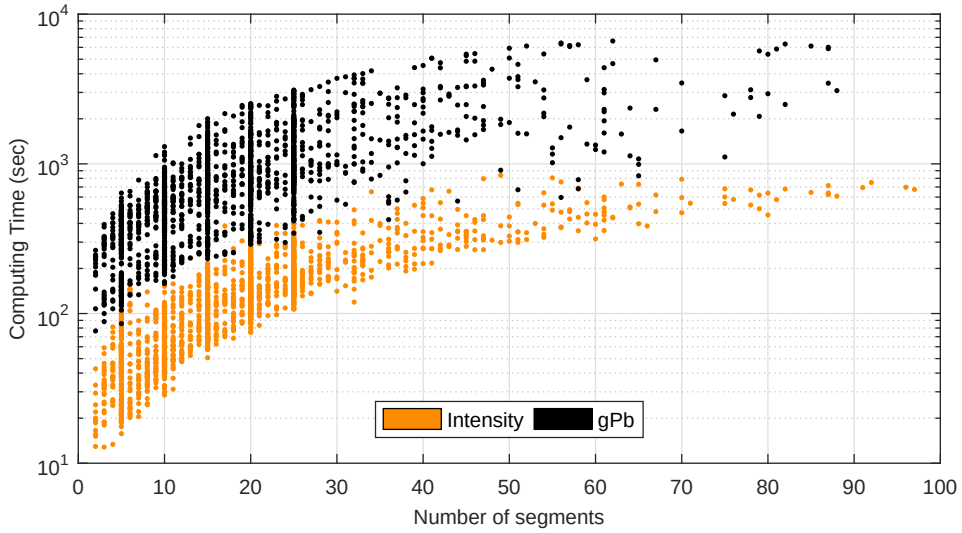


Figure 5: Computation times for the efficient implementation of PFE based on the intensity and gPb graph constructions.

al. [27] implementation of PFE has not yet been publicly released. For this reason, our efficient implementation of PFE will be used for all experiments pertaining to the SOC algorithm for PFE approximation for the remainder of this thesis.

5 Piecewise Flat Embeddings (PFE) with Iteratively Reweighted Rayleigh Quotients (IRRQ)

In this Chapter, we present an alternative algorithm for solving the PFE problem motivated by the Iteratively Reweighted Least Squares (IRLS) algorithms commonly used to solve ℓ_1-minimization problems [9]. In IRLS, ℓ_1-minimization is performed by iteratively solving a succession of weighted least-squares (ℓ_2-minimization) problems, with weights updated at each iteration to decrease the impact of large residual errors. Our new algorithm is termed Iteratively Reweighted Rayleigh Quotients (IRRQ), and, in contrast with the two-stage approach outlined in [27], it does not require an initial estimate of the data embedding, only requires the tuning of a single hyperparameter, and does not rely on a nested iterative structure.

5.1 Iteratively Reweighted Rayleigh Quotients Minimization Algorithm

To solve (3.9) subject to an orthogonality constraint, we will show that the final data embedding can be computed by iteratively solving a series of constrained weighted ℓ_2-minimization problems, with weights updated similarly to IRLS. Each constrained weighted ℓ_2-minimization problem has the form

$$\begin{aligned} \min_{\mathbf{Y}\in\mathbb{R}^{n\times k}} \quad & \mathcal{J}_2^{\mathbf{\Gamma}}(\mathbf{Y}) := \sum_{i=1}^{n}\sum_{j=1}^{n} w_{i,j}\gamma_{i,j}\left\|\hat{\mathbf{y}}_i - \hat{\mathbf{y}}_j\right\|_2^2 \\ \text{subject to:} \quad & \mathbf{Y}^{\mathbf{T}}\mathbf{D}\mathbf{Y} = \mathbf{I} \ , \quad \mathbf{Y}^{\mathbf{T}}\mathbf{D}\mathbf{1} = \mathbf{0} \ , \end{aligned} \tag{5.1}$$

where $\mathbf{\Gamma}$ is the $n \times n$ matrix of weights (with entries $\gamma_{i,j}$) that is updated at each iteration.

To establish a connection between (3.9) and (5.1), we must first eliminate the balance constraint from (5.1) using the result of the following Lemma, which is proved in Appendix 8.1.

Lemma 5.1. *Suppose* $\mathbf{Y} = \mathbf{D}^{-1/2}\mathbf{B}\mathbf{G}$, *where* $\mathbf{B} \in \mathbb{R}^{n\times(n-1)}$ *and* $\mathbf{G} \in \mathbb{R}^{(n-1)\times k}$, *with* $\mathbf{B} = \mathbf{C}\left(\mathbf{C}^{\mathbf{T}}\mathbf{C}\right)^{-1/2}$, *and where* $\mathbf{C}^{\mathbf{T}} \in \mathbb{R}^{(n-1)\times n}$ *projects vectors onto the subspace orthogonal to* $\mathbf{q} = \mathbf{D}^{1/2}\mathbf{1}/\left\|\mathbf{D}^{1/2}\mathbf{1}\right\|$. *Then* $\mathbf{Y}^{\mathbf{T}}\mathbf{D}\mathbf{1} = \mathbf{0}$ *and* $\mathbf{Y}^{\mathbf{T}}\mathbf{D}\mathbf{Y} = \mathbf{G}^{\mathbf{T}}\mathbf{G}$.

By using this Lemma, we can solve (5.1) by first solving:

$$\hat{\mathbf{G}} := \arg\min_{\mathbf{G}^{\mathbf{T}}\mathbf{G}=\mathbf{I}} \mathcal{J}_2^{\mathbf{\Gamma}}\left(\mathbf{D}^{-1/2}\mathbf{B}\mathbf{G}\right) \ , \tag{5.2}$$

and then computing $\mathbf{Y} = \mathbf{D}^{-1/2}\mathbf{B}\hat{\mathbf{G}}$.

If we assume the restrictive assumption that our embedding must be one-dimensional and that none of the differences in embedding components vanish, then we can show that the

solution to (5.1) coincides with the solution to (3.9) when we choose weights according to $\gamma_{i,j} = \left| y_i^* - y_j^* \right|^{-1}$ for $i \neq j$. This proof is shown in Appendix 8.2. In practice, however, many of the component-wise differences will vanish and hence we regularize the weights as in [9]:

$$\gamma_{i,j} := \left[w_{i,j} \left\| \hat{\mathbf{y}}_i - \hat{\mathbf{y}}_j \right\|_2^2 + \varepsilon^2 \right]^{-1/2} \quad , \tag{5.3}$$

and we update ε according to the schedule prescribed by [9], which suggests:

$$\varepsilon \leftarrow \min\left(\varepsilon, n^{-1} r(\mathbf{Y})_{\kappa+1} \right) \quad , \tag{5.4}$$

where $r(\mathbf{Y})_\kappa$ is the κ^{th} largest element of $\left\{ w_{i,j} \left\| \hat{\mathbf{y}}_i - \hat{\mathbf{y}}_j \right\|_2, \forall i, j = 1, \ldots, n \right\}$.

Combining the steps of solving (5.1) and updating (5.3)–(5.4) into a sequence of iterations yields Algorithm (3) for computing the PFE. Note that (5.2) can be transformed into an unconstrained minimization of the following Rayleigh quotient:

$$\begin{aligned} \hat{\mathbf{G}} := \arg\min \operatorname{tr}\Big(& \mathbf{G}^\mathbf{T} \mathbf{B}^\mathbf{T} \mathbf{D}^{-1/2} \hat{\mathbf{L}} \mathbf{D}^{-1/2} \mathbf{B} \mathbf{G} \\ & \cdot \left(\mathbf{G}^\mathbf{T} \mathbf{G} \right)^{-1} \Big) \quad , \end{aligned} \tag{5.5}$$

where $\hat{\mathbf{L}}$ is the Laplacian of the graph having weight matrix $\mathbf{W} \odot \mathbf{\Gamma}$ and $\odot$ denotes Hadamard product. Since (5.1) is equivalent to (5.2) and (5.2) can be transformed into an unconstrained minimization of a Rayleigh Quotient, we term this algorithm *Iteratively Reweighted Rayleigh Quotient* (IRRQ) Minimization.

In contrast with the SOC algorithm, IRRQ requires the tuning of only a single hyperparamter, κ, and it guarantees a solution in which the orthogonality constraint is strictly enforced. Furthermore, IRRQ does not require an initial estimate of the final data embedding. With an initial set of unit weights, IRRQ can be thought of as being implicitly initialized with the solution to the relaxed NCut problem, since $\mathcal{J}_2^{\mathbf{1}\mathbf{1}^\mathbf{T}}(\mathbf{Y})$ is equivalent to the LE objective function $\mathcal{J}_2(\mathbf{Y})$ in (3.7). If different initializations are desired, for instance, by computing an initial embedding $\mathbf{Y}^{(0)}$ using a Gaussian Mixture Model as in [27], these can be incorporated by setting the initial weights $\gamma_{i,j}^{(0)} = \left\| \hat{\mathbf{y}}_i^{(0)} - \hat{\mathbf{y}}_j^{(0)} \right\|_2^{-1}$ or $\gamma_{i,j}^{(0)} = \left[w_{i,j} \left\| \hat{\mathbf{y}}_i^{(0)} - \hat{\mathbf{y}}_j^{(0)} \right\|_2^2 + \varepsilon_0^2 \right]^{-1/2}$.

5.2 Solving Step (a) of the IRRQ Algorithm

The computation to be done in step (a) of Algorithm (3) is non-trivial. From the relationship between (5.1)–(5.2), we can see that computing $\mathbf{Y}^{(m+1)}$ is equivalent to solving

$$\hat{\mathbf{G}}^{(m+1)} := \arg \min_{\mathbf{G}^\mathbf{T}\mathbf{G}=\mathbf{I}} \mathcal{J}_2^{\mathbf{\Gamma}^{(m)}} \left(\mathbf{D}^{-\mathbf{1/2}} \mathbf{B} \mathbf{G} \right) \tag{5.6}$$

Algorithm 3 IRRQ Minimization Algorithm for PFE

procedure IRRQ($\mathbf{W}$, k, κ)

$\gamma_{i,j}^{(0)} := 1$, $\varepsilon_0 := 1$, $m := 0$, $n := \text{size}(\mathbf{W}, 1)$

while $\varepsilon_m > 0$ **do**

(a) $\mathbf{Y}^{(m+1)} := \arg \min_{\substack{\mathbf{Y}^\mathbf{T}\mathbf{D}\mathbf{Y}=\mathbf{I} \\ \mathbf{Y}^\mathbf{T}\mathbf{D}\mathbf{1}=\mathbf{0} \\ \mathbf{Y}\in\mathbb{R}^{n\times k}}} \mathcal{J}_2^{\mathbf{\Gamma}^{(m)}}(\mathbf{Y})$

(b) $\varepsilon_{m+1} := \min\left(\varepsilon_m, n^{-1} r\left(\mathbf{Y}^{(m+1)}\right)_{\kappa+1}\right)$

(c) $\gamma_{i,j}^{(m+1)} := \left[w_{i,j}\left\|\hat{\mathbf{y}}_i - \hat{\mathbf{y}}_j\right\|_2^2 + \varepsilon_{m+1}^2\right]^{-1/2}$

end while

return $\mathbf{Y}^{(m)}$

end procedure

and then computing $\mathbf{Y}^{(m+1)} = \mathbf{D}^{-1/2}\mathbf{B}\hat{\mathbf{G}}^{(m+1)}$. The problem posed in (5.6) can be expressed as the following Rayleigh quotient minimization:

$$\hat{\mathbf{G}}^{(m+1)} := \arg\min \text{tr}\Big(\mathbf{G}^\mathbf{T}\mathbf{B}^\mathbf{T}\mathbf{D}^{-1/2}\hat{\mathbf{L}}^{(m)}\mathbf{D}^{-1/2}\mathbf{B}\mathbf{G} \cdot \left(\mathbf{G}^\mathbf{T}\mathbf{G}\right)^{-1}\Big) \quad , \tag{5.7}$$

where $\hat{\mathbf{L}}^{(m)}$ is the Laplacian of the graph having weight matrix $\mathbf{W} \odot \mathbf{\Gamma}^{(m)}$ and $\odot$ denotes Hadamard product. The solution to (5.7) is given by $\hat{\mathbf{G}}^{(m+1)} = \mathbf{U}\mathbf{H}$, where $\mathbf{U} \in \mathbb{R}^{(n-1)\times k}$ is the matrix whose columns are the orthonormal eigenvectors $\mathbf{u}_1, \ldots, \mathbf{u}_k$ corresponding to the smallest eigenvalues of $\mathbf{B}^\mathbf{T}\mathbf{D}^{-1/2}\hat{\mathbf{L}}^{(m)}\mathbf{D}^{-1/2}\mathbf{B}$, and $\mathbf{H} \in \mathbb{R}^{k\times k}$ is an arbitrary orthogonal matrix. (Note that by eliminating the balance constraint, we also eliminate the possibility of a trivial eigenvalue of $\mathbf{B}^\mathbf{T}\mathbf{D}^{-1/2}\hat{\mathbf{L}}^{(m)}\mathbf{D}^{-1/2}\mathbf{B}$. Such an eigenvalue would have eigenvector $\mathbf{p}$ for which $\mathbf{B}\mathbf{p}$ is in the direction of $\mathbf{q}$; however, this is contradicted by the fact that $\text{range}(\mathbf{B}) = \text{range}(\mathbf{C}^\mathbf{T})$.)

In order for this solution to scale to large n, we must consider the structure of $\mathbf{C}$, and whether or not the matrix $\mathbf{B}$ must explicitly be constructed. Since we have the ability to choose any $\mathbf{C}$ such that $\mathbf{C}^\mathbf{T}$ projects vectors onto the subspace orthogonal to $\mathbf{q}$, we choose $\mathbf{C}^\mathbf{T} = [\hat{\mathbf{q}} \,|\, -q_1\mathbf{I}_{n-1}]$, where $\hat{\mathbf{q}} = [q_2, q_3, \ldots, q_n]^\mathbf{T}$ and $\mathbf{I}_{n-1}$ is the $(n-1)\times(n-1)$ identity matrix. This particular choice of $\mathbf{C}$ is sparse, and therefore, as shown in Appendix 8.3, the multiplication of an arbitrary vector by $\mathbf{B}^\mathbf{T}\mathbf{D}^{-1/2}\hat{\mathbf{L}}^{(m)}\mathbf{D}^{-1/2}\mathbf{B}$ can be performed efficiently without explicitly constructing $\mathbf{B}$.

Finally, we note that the solution to step (a) is not unique: postmultiplying $\mathbf{Y}^{(m+1)}$ by $\mathbf{H}^\mathbf{T}$, where $\mathbf{H}$ is orthogonal, still produces a valid solution. This does not pose a problem for steps (b) and (c) of Algorithm (3) since r and $\gamma_{i,j}$ are invariant to such transformations of $\mathbf{Y}^{(m+1)}$. As a

consequence, IRRQ minimization could yield an entire family of solutions to the PFE problem, which could be problematic since the ℓ_1-norm is not invariant under orthogonal transformations. In practice, however, we have found that the ℓ_1-norm is minimized for the choice $\mathbf{H} = \mathbf{I}$ and, as such, we suggest this choice. Proof that this is the best choice remains an open problem.

5.3 Choosing κ for Rapid Convergence

Linear convergence in IRLS algorithms for ℓ_1-minimization can typically be achieved if κ is chosen large enough so that if the solution is θ-sparse, then $\kappa > \theta$. We note that there are more sophisticated convergence results provided in [9], but this is a good rule-of-thumb. While proving convergence results for the IRRQ algorithm remains an open problem, we use a similar strategy to [9] in choosing κ. In practice, the main difficulty in choosing κ is that θ is not known exactly until the problem is solved. To approximate θ, we use an estimate $\hat{\theta}$ equal to twice the number of graph edges that connect different clusters from a k-means clustering performed on the initial embedding $\mathbf{Y}^{(0)}$.

To provide "scale-free" behavior, we introduce the hyperparameter $\tilde{\kappa}$ that can be selected in $(0, 1)$. $\tilde{\kappa}$ can then be mapped to κ by $\kappa = \hat{\theta} + \tilde{\kappa}\left(2|E| - \hat{\theta}\right)$, where $|E|$ is the total number of edges in the graph.

6 Segmentation Experiments and Results

In this Chapter, we provide an overview of the experiments we conducted to test the relative performance of both CCNCut and NCut minimization and to compare the SOC and IRRQ algorithms on the task of image segmentation. We then provide results from these experiments and explore the benefits and limitations of the two algorithms.

6.1 Segmentation Experiments

To compare the performance of CCNCut and NCut minimization, as well as to compare the relative performance of the SOC and IRRQ algorithms for CCNCut minimization, we use these algorithms to segment the 200 test images in the BSDS500 dataset [1]. This dataset contains a variety of manually-labeled segmentations with varying numbers of segments that can be used as ground truth. Similar to the experiments we conducted in Chapter 4, we construct two types of graphs for each image. We first downsample the image by a factor of four and then compute either the intensity differences across pixels within a four-pixel neighborhood of one another based on the 4-nearest neighbor algorithm, or compute the globalized probability of boundary (gPb) at each pixel using the code provided with the BSDS500 dataset.

For each of the algorithms, we segment each image multiple times, once for each k (number of segments) value reflected in the ground truth and once for each $k = 5, 10, 15, 20$, and 25 not reflected in the ground truth. Each segmentation algorithm proceeds by computing the desired data embedding (LE or PFE) and then performing k-means clustering on the final embedding to assign discrete labels to each pixel.

For CCNCut minimization using the SOC algorithm, we test segmentation performance based on two different initializations. The first is based on the Laplacian Eigenmaps (LE) algorithm and the second is based on the Gaussian Mixture Model (GMM) as in [27]. These two initialization schemes are labeled as C-SOC-L and C-SOC-G respectively in our experimental results. For both initialization methods, we perform the two-stage approach outlined in [27]. Unless otherwise stated, in stage one, we set the maximum number of outer iterations to 10 and the maximum number of inner iterations to 5 with hyperparameters $\lambda = 10,000$ and $r = 100$. Unless otherwise stated, in stage two, we set the maximum number of inner iterations to 100 with hyperparameters $\lambda = 10,000$ and $r = 10$. All of these parameter choices are the same as in [27].

For CCNCut minimization using the IRRQ algorithm, we again test segmentation performance

based on two different weight initialization schemes. The first is our default scheme where all weights are initialized to unity (implicit LE initialization of PFE) and the second is a scheme where weights are initialized according to embeddings formed from a GMM. These two initialization schemes are labeled as C-IRRQ and C-IRRQ-G respectively in our experimental results. In both initialization schemes, unless otherwise stated, the maximum number of iterations is 20, and the hyperparameter $\tilde{\kappa}$ is set to 0.02. The seed for the random number generator is reset before every initialization, ensuring that the same GMM's are used to initialize the C-SOC-G and C-IRRQ-G algorithms.

6.2 State-of-the-Art on BSDS500

Although Yu et al. [27] held state-of-the-art results on the BSDS500 dataset at the time of their publication in 2015, we must note a few key points. First, the state-of-the-art results published in [27] were based on the contour-driven hierarchical segmentation schemes presented in [1] and [2]. While the hierarchical schemes were not used in this thesis for discrete labeling, we believe these clustering strategies would be interesting to explore in future work for both the SOC and IRRQ algorithms. Second, in 2016, [7] developed a scheme to better align hierarchical data partitions based on depth and scale that produced comparable results to those presented in [27].

In 2016, Zhao and Griffin [29] utilized multi-level cues and semantic predictions from a Fully Convolutional Neural Network (FCNN) [14] to build an image segmentation scheme based on a bottom-up approach. At the time of writing this thesis, the results presented in [29] hold state-of-the-art on the BSDS500 dataset. Table (3) shows the results from [7, 27, 29] to give the reader an understanding of current state-of-the-art segmentation covering, PRI, and VI values on the BSDS500 dataset. In the table, there are two results reported for each quantitative measure: Optimal Dataset Scale (ODS) and Optimal Image Scale (OIS) [1]. ODS and OIS provide two different approaches for generating a single segmentation of an image, based on the use of hierarchical clustering techniques.

6.3 Results

Figures (6) and (7) demonstrate segmentation results for various images from the BSDS500 dataset, each chosen for specific values of k based on the gestalt sense of the image provided by the segmentation. Figure (8) demonstrates C-IRRQ segmentation results for varying cluster numbers k.

Method	Covering		PRI		VI	
	ODS	OIS	ODS	OIS	ODS	OIS
MCG [2]	0.61	0.66	0.83	0.86	1.57	1.39
PFE + mPb [27]	0.62	0.67	0.84	0.86	1.61	1.43
PFE + MCG [27]	0.62	0.68	0.84	0.87	1.56	1.36
SAA [7]	0.63	0.68	0.83	0.86	1.53	1.38
FCNN [29]	0.64	0.70	0.84	**0.88**	1.42	1.23
FCNN + HED [29]	**0.66**	**0.71**	**0.86**	**0.88**	**1.36**	**1.20**

Table 3: Comparison of various segmentation methods on the BSDS500 dataset. 'MCG' denotes results produced by the Multiscale Combinatorial Grouping method [2]. 'PFE + mPb' and 'PFE + MCG' denote results produced by the Piecewise Flat Embedding technique using global contour information [27]. 'SAA' denotes results produced by the Scale-Aware Alignment technique [7]. 'FCNN' and 'FCNN + HED' denote results produced by the FCNN strategy both without and with the holistically-nested edge detection scheme respectively [24]. Best results for each performance measure highlighted in bold.

Although the results presented in Figures (6), (7), and (8) provide qualitative segmentation results for comparison of the NCut and CCNCut minimization algorithms, we use the three quantitative measures outlined in [1] to determine which algorithm yields the *best* segmentation results. These are the same three quantitative measures we used in Chapter 4, and include segmentation covering, Probabilistic Rand Index (PRI), and Variation of Information (VI). Segmentation covering and PRI will increase as a segmentation becomes closer to ground truth, while VI will decrease.

As in [27] and in Chapter 4, we report results using both a *fixed* and a *dynamic* scheme. In the fixed scheme, we run the algorithm once for each k value reflected in the ground truth and average the segmentation results. In the dynamic scheme, we run the algorithm once for each $k = 5, 10, 15, 20$, and 25 and choose the segmentation corresponding to the k value that yielded the best performance with respect to the three quantitative performance measures.

Tables (4) and (5) demonstrate the performance results of each algorithm based on the intensity and gPb graph constructions respectively. From these tables, we might infer the following: CCNCut-based segmentation performs better than NCut-based segmentation in general, CCNCut minimization performed by the SOC algorithm performs better than CCNCut minimization performed by IRRQ, the SOC algorithm performs better when initialized with the GMM over LE, and IRRQ minimization yields equivalent results independent of initialization. We note that results from the two different methods of graph construction yield nearly identical results, with the exception of the SOC algorithm initialized by LE and the majority of the VI measures. In the case of C-SOC-L, the gPb construction yields slightly better results.

Method	Covering		PRI		VI	
	fixed	dynamic	fixed	dynamic	fixed	dynamic
NCut	0.27	0.38	0.74	0.75	3.04	2.56
C-SOC-L	0.30	0.40	0.74	0.76	2.87	2.43
C-SOC-G	**0.42**	**0.52**	**0.75**	**0.79**	**2.59**	**2.20**
C-IRRQ	0.31	0.41	0.73	0.75	2.90	2.42
C-IRRQ-G	0.31	0.43	0.71	0.75	2.86	2.35

Table 4: Comparison of various segmentation methods on BSDS500 test set using intensity graph construction, averaged across images. Best results for each performance measure highlighted in bold.

Method	Covering		PRI		VI	
	fixed	dynamic	fixed	dynamic	fixed	dynamic
NCut	0.27	0.39	0.74	0.75	3.04	2.50
C-SOC-L	0.39	0.47	**0.75**	**0.78**	2.40	2.13
C-SOC-G	**0.43**	**0.51**	0.74	**0.78**	**2.30**	**2.04**
C-IRRQ	0.29	0.42	0.73	0.76	2.95	2.37
C-IRRQ-G	0.30	0.42	0.73	0.76	2.95	2.37

Table 5: Comparison of various segmentation methods on BSDS500 test set using gPb graph construction, averaged across images. Best results for each performance measure highlighted in bold.

Although Tables (4) and (5) provide us with some notable inferences about NCut and CCNCut minimization, we further provide the results from a Wilcoxon rank sum test to determine the statistical significance of the Covering, PRI, and VI measures we obtained numerically. The Wilcoxon rank sum test tests the null hypothesis that two sets of performance results are from continuous distributions with equal medians. Tables (6) and (7) report the p-values for which we would reject the null hypothesis based on the intensity and gPb graph constructions respectively.

From Table (6), we can infer that most of the Covering results are statistically significantly different from one another with the exception of C-SOC-L from C-IRRQ, and C-IRRQ from C-IRRQ-G. We note that almost all of the PRI values from Table (6) are not significantly different from one another, indicating that the PRI metric may not be discriminative in assessing differences between different algorithms. For the VI metric, Table (6) indicates that there is no statistical significance between C-SOC-L and C-IRRQ or C-IRRQ-G, or between C-IRRQ and C-IRRQ-G. It is interesting to note that in all three metrics, C-SOC-L and C-IRRQ, and C-IRRQ and C-IRRQ-G do not yield significantly different results.

Similarly, Table (7) demonstrates that all methods yield significantly different Covering results

	B	C	D	E	B	C	D	E
A	**2e-05**	**2e-46**	**3e-05**	**7e-07**	**3e-03**	**1e-33**	**2e-04**	**1e-09**
B		**4e-34**	9e-01	3e-01		**9e-25**	3e-01	**1e-03**
C			**2e-29**	**3e-25**			**7e-21**	**7e-15**
D	Fixed: Covering			4e-01	Dynamic: Covering			3e-02
A	8e-01	4e-01	3e-01	2e-02	4e-01	**3e-04**	9e-01	1e-00
B		6e-01	2e-01	1e-02		6e-03	5e-01	4e-01
C			5e-02	**7e-04**			**5e-04**	**1e-04**
D	Fixed: PRI			2e-01	Dynamic: PRI			8e-01
A	**6e-05**	**2e-15**	**4e-03**	**4e-04**	**4e-05**	**6e-13**	**3e-05**	**4e-08**
B		**8e-07**	3e-01	8e-01		**1e-06**	9e-01	8e-02
C			**3e-08**	**3e-06**			**2e-06**	**7e-04**
D	Fixed: VI			4e-01	Dynamic: VI			9e-02

Table 6: *p*-values of two-sided Wilcoxon rank sum tests for the intensity graph construction at which we would reject the null hypothesis that performance measures computed from methods *i* and *j* have the same medians. Methods are: (A) NCut, (B) C-SOC-L, (C) C-SOC-G, (D) C-IRRQ, (E) C-IRRQ-G. *p*-values less than 0.05 are highlighted in bold.

	B	C	D	E	B	C	D	E
A	**2e-41**	**2e-49**	**3e-03**	**1e-03**	**4e-18**	**3e-27**	**1e-04**	**3e-04**
B		**8e-05**	**3e-31**	**3e-30**		**2e-03**	**9e-08**	**5e-08**
C			**3e-40**	**3e-39**			**3e-15**	**2e-15**
D	Fixed: Covering			8e-01	Dynamic: Covering			8e-01
A	3e-01	8e-01	7e-01	8e-01	**3e-02**	**3e-02**	9e-01	9e-01
B		4e-01	2e-01	2e-01		9e-01	4e-01	3e-01
C			6e-01	6e-01			4e-01	4e-01
D	Fixed: PRI			9e-01	Dynamic: PRI			9e-01
A	**1e-34**	**1e-38**	9e-02	6e-02	**5e-20**	**3e-24**	**3e-04**	**1e-04**
B		**4e-02**	**2e-27**	**5e-27**		6e-02	**3e-10**	**6e-10**
C			**5e-32**	**2e-31**			**3e-14**	**6e-14**
D	Fixed: VI			8e-01	Dynamic: VI			8e-01

Table 7: *p*-values of two-sided Wilcoxon rank sum tests for the gPb graph construction at which we would reject the null hypothesis that performance measures computed from methods *i* and *j* have the same medians. Methods are: (A) NCut, (B) C-SOC-L, (C) C-SOC-G, (D) C-IRRQ, (E) C-IRRQ-G. *p*-values less than 0.05 are highlighted in bold.

with the exception of C-IRRQ and C-IRRQ-G. Again, for the PRI metric in Table (7), almost none of the methods yield significantly different results, indicating that PRI may not be discriminative in assessing differences between methods. For VI in Table (7), depending on whether the fixed or dynamic scheme is chosen, there may or may not be significant differences between the NCut and C-IRRQ methods, and between C-SOC-L and C-SOC-G. For all three metrics, and for both types of graph construction, however, C-IRRQ and C-IRRQ-G do not yield significantly different results.

6.4 Comparison of Algorithms for CCNCut Minimization

It may be tempting to infer that SOC-based CCNCut minimization is preferable to the IRRQ algorithm based solely on segmentation performance measures. However, it is evident that the SOC-based methods do not yield similar results provided different initializations, at least when they are restricted to the maximum ten outer loop iterations suggested in [27].

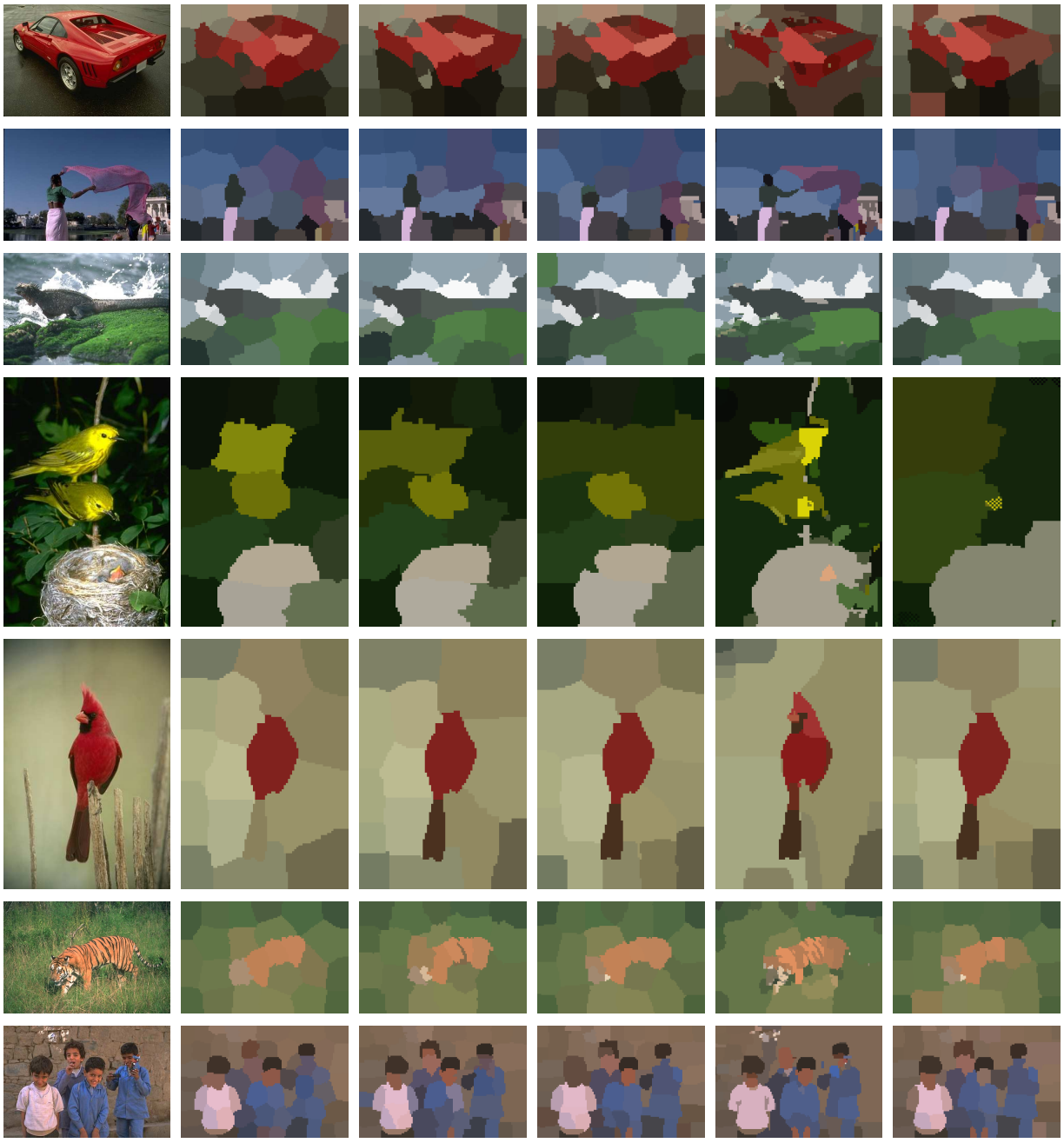
To determine whether this result is a consequence of early stopping in the first stage of the SOC method, we re-ran the SOC-based segmentation algorithm with a maximum of 200 outer loop iterations. For the gPb graph construction, this increase in maximum iterations yielded performance measures that were not significantly different when SOC was intialized with LE versus GMM. However, for the intensity graph construction, increasing the maximum number of outer iterations did not significantly change the performance metrics, which still remained in favor of GMM initialization.

While increasing the maximum number of iterations yielded more consistent SOC-based segmentation results for the gPb graph construction, we now turn our concern towards computation time. Figure (9) demonstrates the computation time needed to compute the PFE (for each value of k) via IRRQ, SOC with 10 outer loop iterations, and SOC with 200 outer loop iterations, all initialized with LE and based on gPb graph construction. (Timing for GMM-initialized methods is similar.) As we can see, SOC with 200 outer iterations requires nearly two orders of magnitude more computation time than IRRQ.

In Figure (10), we compare segmentation results from different initializations of the same algorithms based on both the intensity and gPb graph constructions. We see that, based on both graph constructions, the Covering and PRI measures between segmentations from IRRQ initialized with LE versus GMM are higher, and VI lower, than between segmentations from SOC using both initializations.

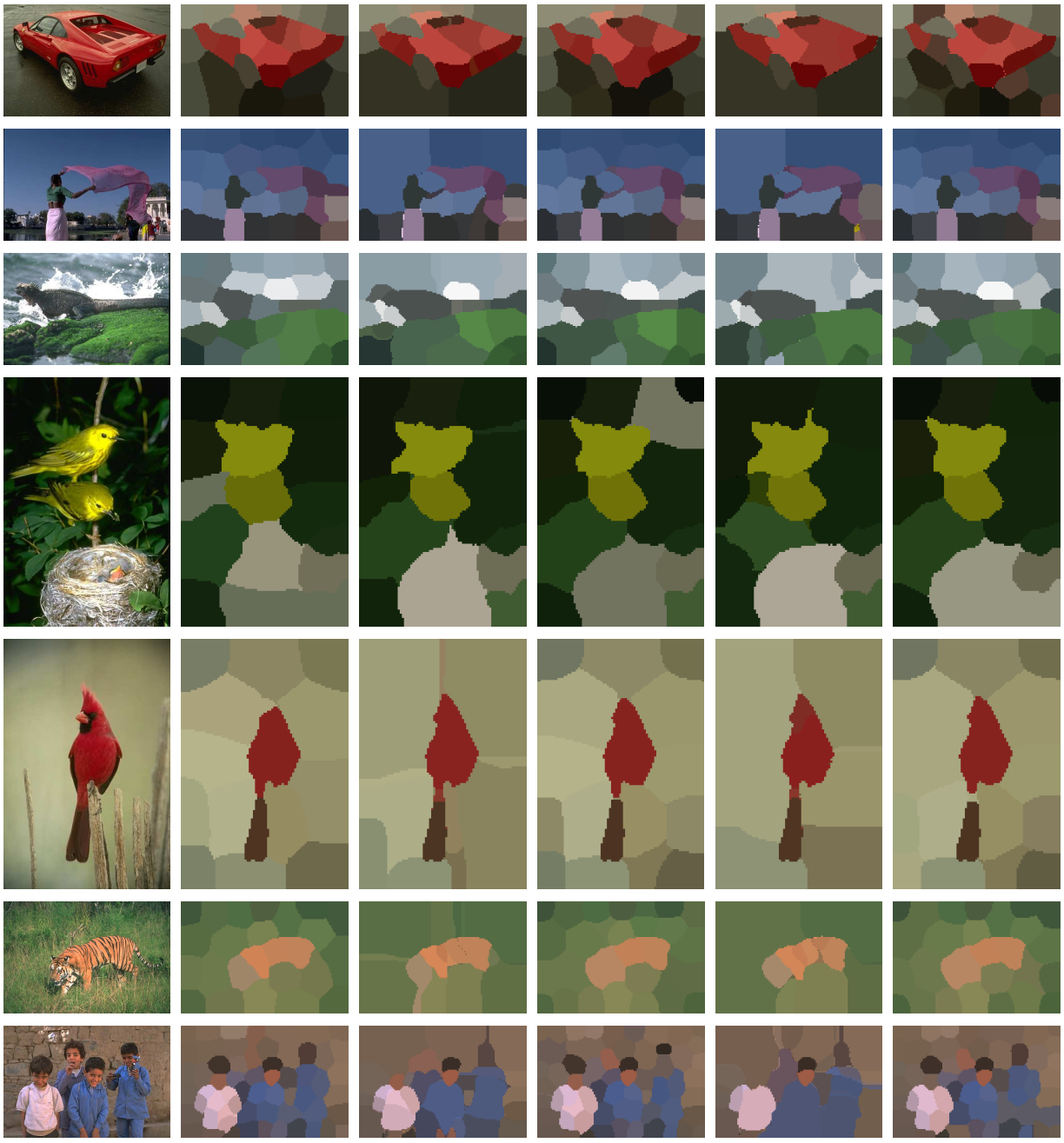
These results are a strong indication that the IRRQ minimization algorithm provides more

consistent segmentation results, with more efficient computation times. Recall, that in order to reach lower cut costs, the SOC algorithm relaxes the orthogonality constraint in its second stage. Figure (11) demonstrates the cut cost of segmentation using SOC versus IRRQ when initialized from LE and GMM on a particular image from the BSDS500 dataset [1]. While the SOC method attains a much lower cost than IRRQ after just 10 iterations, this lower energy comes with the penalty of disobeying the orthogonality constraint. Due to its consistency, efficient computation time, and strict enforcement of the orthogonality constraint, we have ample evidence to suggest that IRRQ is a more suitable algorithm for approximate CCNCut minimization than SOC.



(a) original (b) NCut (c) C-SOC-L (d) C-IRRQ (e) C-SOC-G (f) C-IRRQ-G

Figure 6: BSDS500 test images and segmentation results with intensity graph construction: (a) original, (b) NCut, (c) CCNCut by SOC initialized with Laplacian Eigenmaps, (d) CCNCut by IRRQ, (e) CCNCut by SOC Initialized with Gaussian mixture model, (f) CCNCut by IRRQ initialized with Gaussian mixture model.



(a) original (b) NCut (c) C-SOC-L (d) C-IRRQ (e) C-SOC-G (f) C-IRRQ-G

Figure 7: BSDS500 test images and segmentation results with gPb graph construction: (a) original, (b) NCut, (c) CCNCut by SOC initialized with Laplacian Eigenmaps, (d) CCNCut by IRRQ, (e) CCNCut by SOC Initialized with Gaussian mixture model, (f) CCNCut by IRRQ initialized with Gaussian mixture model.

Figure 8: BSDS500 test image with CCNCut-based segmentation using the IRRQ algorithm with the gPb graph construction for various values of k.

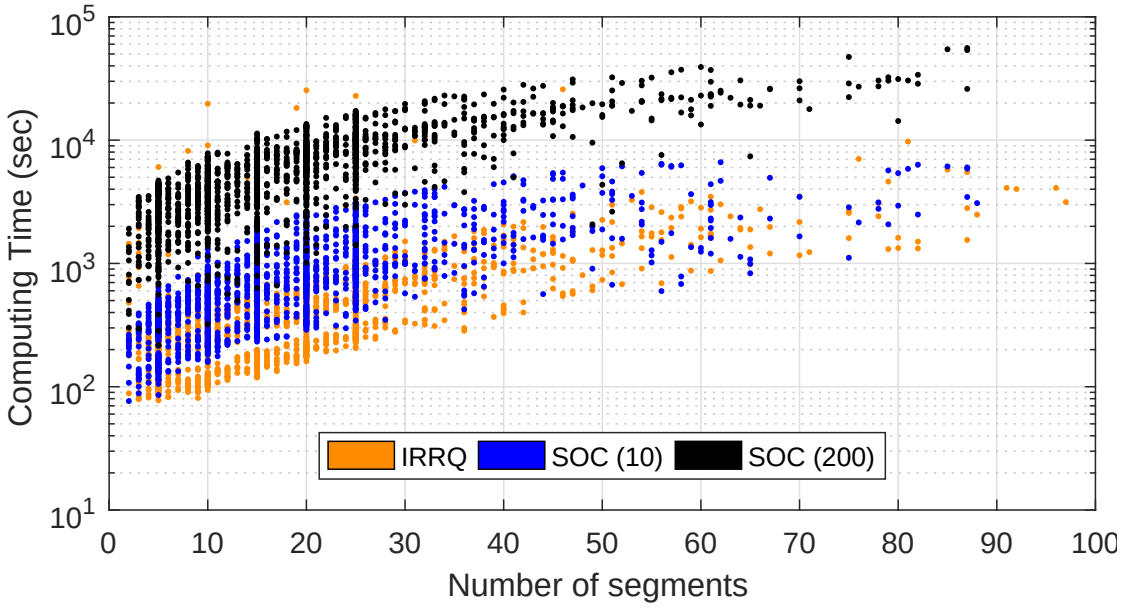
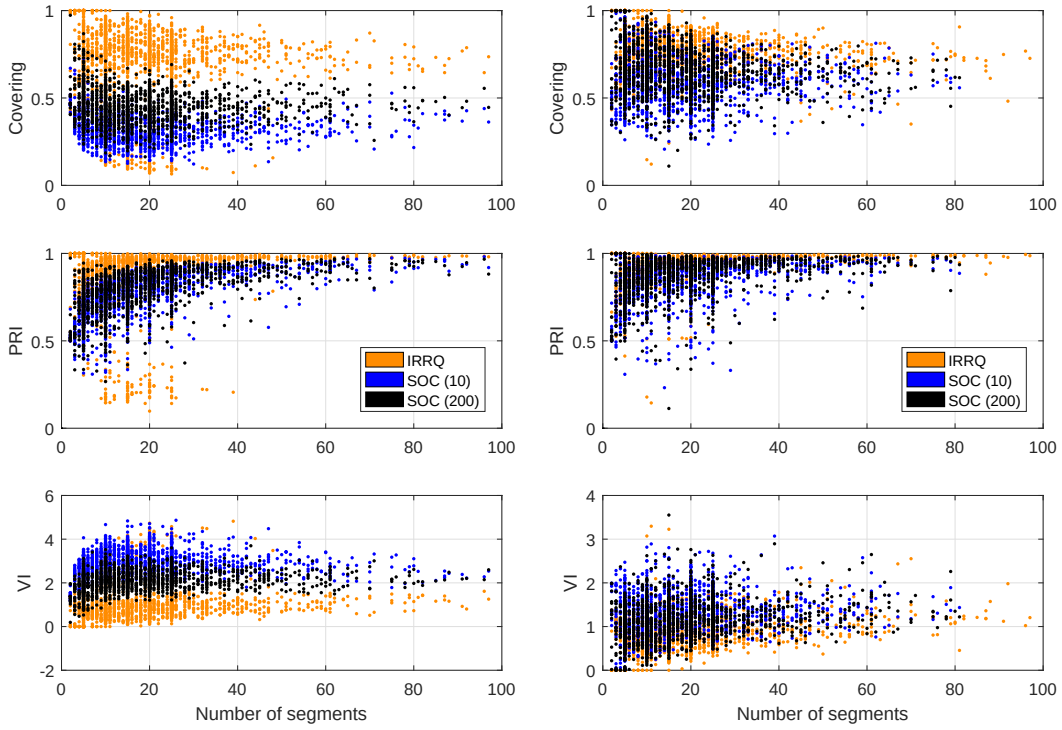


Figure 9: Computation times for CCNCut-based segmentation of BSDS images for various numbers of segments based on gPb graph construction. Algorithms are all initialized with LE.



(a) Intensity Graph Construction

(b) gPb Graph Construction

Figure 10: Covering, PRI, and VI between segmentations generated from GMM and LE initializations of each CCNCut minimization method based on (a) intensity graph construction and (b) gPb graph construction.

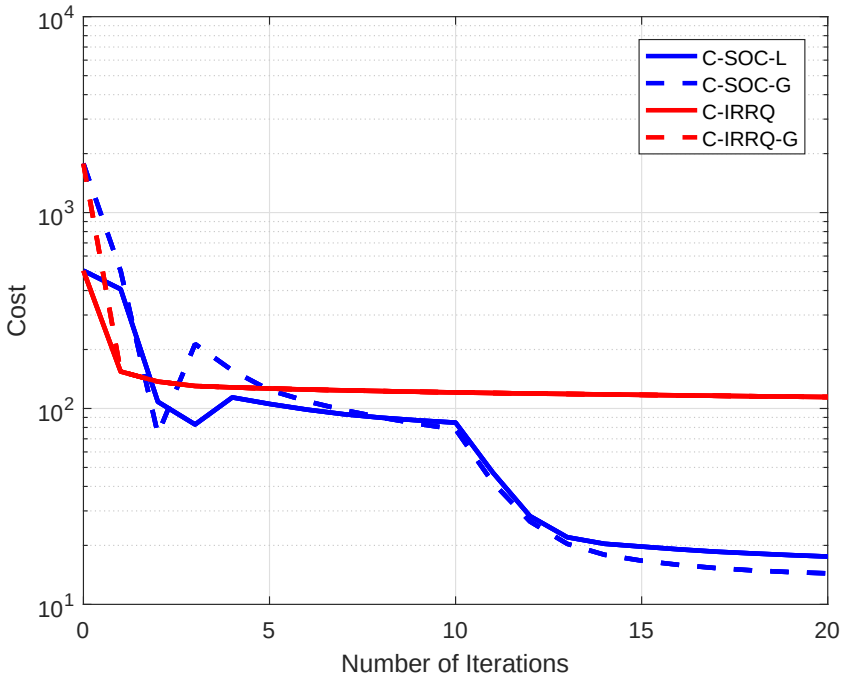


Figure 11: Minimization of the cost function for a particular image as a function of number of iterations.

7 Conclusions and Future Work

7.1 Conclusions

Graph-based partitioning algorithms have proven useful in their ability to help automate the task of image segmentation in the computer vision and machine learning communities. By defining a new cut cost for partitioning, with a more modest normalization scheme than previous methods, we showed that minimization of the CCNCut is advantageous in avoiding too many singleton partitions while also maintaining a notion of *balanced* partitions. Experiments on a publicly available dataset demonstrated that CCNCut-based image segmentation, with simple clustering on the final data embedding, provides more accurate results with respect to ground truth than NCut-based image segmentation.

In Chapter 3, we defined the CCNCut cost, which is NP-hard to minimize. To approximate a solution to the CCNCut minimization problem, we demonstrated how a relaxation of the CCNCut could be obtained that is mathematically equivalent to the PFE introduced in [27].

In Chapter 4, we outlined the two-stage numerical approach for approximating a solution to the PFE problem introduced in [27]. To allow the algorithm to be more computationally efficient on larger images, we proposed two improvements to the two-stage approach. These improvements included reformulating the original PFE problem to allow multiple linear algebra computations to be performed in parallel and utilizing an iterative linear solver to quickly solve a succession of least-squares problems.

In Chapter 5, we highlighted the limitations of the two-stage approach from [27] including the requirement of an initial estimate of the data embedding , the use of nested iterations of two separate algorithms, each with their own set of tunable hyperparameters, and the loose enforcement of the orthogonality constraint. Due to these limitations, we introduced a new algorithm for approximating a solution to the PFE problem, inspired by the well-known IRLS algorithms for ℓ_1-minimization. We demonstrated how our new algorithm, IRRQ, involves the minimization of a Rayleigh quotient and provided helpful implementation details pertaining to the algorithm.

Finally, in Chapter 6, we provided a comparison of NCut-based image segmentation with CCNCut-based image segmentation on the BSDS500 dataset [1]. We showed that the IRRQ algorithm is less sensitive to initialization than the SOC algorithm for PFE approximation, and CCNCut-based image segmentation is more accurate with respect to ground truth than NCut-based image segmentation.

7.2 Future Work

There are a few immediate possibilities for future work. The first opportunity involves testing CCNCut-based image segmentation on more publicly available RGB-image datasets. Similarly, we would like to test CCNCut-based clustering on other image modalities, such as hyperspectral image data, as well as other clustering datasets that may not necessarily contain images.

A more interesting extension of this work involves the use of semi-supervised segmentation. The formulations of NCuts and CCNCuts presented in this thesis are both forms of unsupervised segmentation schemes in that they do not require any labeled training data for segmentation. Over the years, the incorporation of expert a priori knowledge about how pixels or regions should be grouped together has improved segmentation results of the NCuts algorithm and has resulted in a semi-supervised segmentation approach.

This a priori information can be provided to graph-based segmentation algorithms in the form of must-link or cannot-link constraints. These constraints specify groups of two or more pixels that must be grouped in the same or different partitions respectively. Formulations of NCuts that incorporate must-link and/or cannot-link constraints were introduced in [8, 10, 15, 26] and resulted in more accurate segmentation results. Based on these improved results, we would be interested in exploring the use of constraints in a semi-supervised segmentation approach to PFE.

8 Appendix

8.1 Proof of Lemma 5.1

First, note that:

$$\begin{aligned}\mathbf{Y^T D 1} &= \mathbf{G^T B^T D}^{1/2}\mathbf{1} \\ &= \mathbf{G^T}\Big(\mathbf{C^T C}\Big)^{-1/2}\mathbf{C^T D}^{1/2}\mathbf{1}\ ,\end{aligned} \tag{8.1}$$

which vanishes because $\mathbf{C^T}$ annihilates vectors in the direction of $\mathbf{q}$. Secondly, note that:

$$\begin{aligned}\mathbf{Y^T D Y} &= \mathbf{G^T B^T B G} \\ &= \mathbf{G^T}\Big(\mathbf{C^T C}\Big)^{-1/2}\mathbf{C^T C}\Big(\mathbf{C^T C}\Big)^{-1/2}\mathbf{G} \\ &= \mathbf{G^T G}\ .\end{aligned} \tag{8.2}$$

8.2 Special Case of (3.9)–(5.1) Equivalence

In this section, we prove the following lemma:

Lemma 8.1. *Suppose* $\mathbf{g}^* \in \mathbb{R}^{n-1}$ *such that* $\mathbf{y}^* = \mathbf{D}^{-1/2}\mathbf{B}\mathbf{g}^*$ *solves (3.9),* $\mathbf{y}^*$ *satisfies* $y_i^* \neq y_j^*$ *for all* $i \neq j$*, and* $\gamma_{i,j} = \left|y_i^* - y_j^*\right|^{-1}$ *for* $i \neq j$*. Then the solution to (5.1) coincides with* $\mathbf{y}^*$*.*

To prove Lemma 8.1, we first introduce and prove this lemma:

Lemma 8.2. *Let* $\mathbf{g}$ *be a unit vector in* $\mathbb{R}^{n-1}$*, and let* $\mathbf{y} = \mathbf{D}^{-1/2}\mathbf{B}\mathbf{g}$*. Then* $\mathbf{g}$ *solves (5.2) if and only if*

$$\begin{aligned}&\left|\sum_{y_i \neq y_j} w_{i,j} s_{i,j}(\xi_i - \xi_j) - \mathcal{J}_1(\mathbf{y}) \cdot \Big(\mathbf{g^T}\boldsymbol{\eta}\Big)\right| \\ &\qquad\qquad \leq \sum_{y_i = y_j} w_{i,j}|\xi_i - \xi_j|\ ,\end{aligned} \tag{8.3}$$

where $\boldsymbol{\xi} = \mathbf{D}^{-1/2}\mathbf{B}\boldsymbol{\eta}$*, for all* $\boldsymbol{\eta} \in \mathbb{R}^{n-1}$ *such that* $\mathbf{g} + 2\boldsymbol{\eta}$ *is a unit vector.*

The proofs of these lemmas use the simplified notation $\mathcal{J}_{1'}(\mathbf{z}) := \mathcal{J}_1\Big(\mathbf{D}^{-1/2}\mathbf{B}\mathbf{z}\Big)$ and $\mathcal{J}_{2'}^{\boldsymbol{\Gamma}}(\mathbf{z}) := \mathcal{J}_2^{\boldsymbol{\Gamma}}\Big(\mathbf{D}^{-1/2}\mathbf{B}\mathbf{z}\Big)$.

Proof of Lemma 8.2. This proof follows similarly to the proof of Lemma 2.1 in [9]. Suppose $\mathbf{g}$ solves (5.2). Then for all $\boldsymbol{\eta} \in \mathbb{R}^{n-1}$, we have:

$$\mathcal{J}_{1'}\left(\frac{\mathbf{g} + t\boldsymbol{\eta}}{\|\mathbf{g} + t\boldsymbol{\eta}\|_2}\right) \geq \mathcal{J}_{1'}(\mathbf{g}) \tag{8.4}$$

for all $t \in \mathbb{R}$ except when $t\boldsymbol{\eta} = -\mathbf{g}$. Noting that $\mathcal{J}_{1'}(\beta\mathbf{x}) = |\beta|\,\mathcal{J}_{1'}(\mathbf{x})$ for all $\beta \in \mathbb{R}$, and using the relationship between $\mathcal{J}_{1'}$ and $\mathcal{J}_1$, we can write (8.4) as:

$$\frac{\mathcal{J}_1(\mathbf{y} + t\boldsymbol{\xi})}{\|\mathbf{g} + t\boldsymbol{\eta}\|_2} \geq \mathcal{J}_1(\mathbf{y}) \quad . \tag{8.5}$$

Expanding both sides of (8.5) yields:

$$\frac{\sum_{i,j} w_{i,j}\left|y_i - y_j + t\left(\xi_i - \xi_j\right)\right|}{\sqrt{1 + 2t\mathbf{g}^{\mathbf{T}}\boldsymbol{\eta} + t^2\boldsymbol{\eta}^T\boldsymbol{\eta}}} \geq \sum_{i,j} w_{i,j}\left|y_i - y_j\right| \quad . \tag{8.6}$$

Taking t sufficiently small so that $\left(y_i - y_j + t\left(\xi_i - \xi_j\right)\right)$ has the same sign as $s_{i,j} := \operatorname{sign}\left(y_i - y_j\right)$ for all (i,j) such that $y_i \neq y_j$, we can write (8.6) as:

$$\begin{aligned} &\sum_{y_i \neq y_j} w_{i,j} s_{i,j}\left(y_i - y_j\right) + t \sum_{y_i \neq y_j} w_{i,j} s_{i,j}\left(\xi_i - \xi_j\right) \\ &\qquad + |t| \sum_{y_i = y_j} w_{i,j}\left|\xi_i - \xi_j\right| \\ &\geq \left(1 + 2t\mathbf{g}^{\mathbf{T}}\boldsymbol{\eta} + t^2\boldsymbol{\eta}^T\boldsymbol{\eta}\right)^{1/2} \sum_{y_i \neq y_j} w_{i,j} s_{i,j}\left(y_i - y_j\right) \quad . \end{aligned} \tag{8.7}$$

If we write (8.7) using the shorthand notation:

$$c_1 + tc_2 + |t|\,c_3 \geq \left(1 + tc_4 + t^2 c_5\right)^{1/2} c_1 \quad , \tag{8.8}$$

we find after algebraic manipulation that the following inequality must hold:

$$\left|2c_1c_2 - c_1^2c_4 + tc_2^2 + tc_3^2 - tc_1^2c_5\right| \leq 2c_1c_3 + 2tc_2c_3 \quad , \tag{8.9}$$

whence (8.3) arises by considering the case $t = 0$, completing the proof of the forward direction. To prove the backward direction, first note that if $\|\mathbf{g} + 2\boldsymbol{\eta}\| = 1$, then $\mathbf{g}^{\mathbf{T}}\boldsymbol{\eta} = 2\mathbf{g}^{\mathbf{T}}\boldsymbol{\eta} + \boldsymbol{\eta}^T\boldsymbol{\eta}$, and so (8.3) becomes:

$$\left|\sum_{y_i \neq y_j} w_{i,j} s_{i,j}\left(\xi_i - \xi_j\right) - \mathcal{J}_1(\mathbf{y}) \cdot \left(2\mathbf{g}^{\mathbf{T}}\boldsymbol{\eta} + \boldsymbol{\eta}^T\boldsymbol{\eta}\right)\right| \leq \sum_{y_i = y_j} w_{i,j}\left|\xi_i - \xi_j\right| \quad . \tag{8.10}$$

Now, we have:

$$\begin{aligned}
\mathcal{J}_{1'}\left(\frac{\mathbf{g}+\boldsymbol{\eta}}{\|\mathbf{g}+\boldsymbol{\eta}\|_2}\right) &= \frac{\mathcal{J}_1(\mathbf{y}+\boldsymbol{\xi})}{\|\mathbf{g}+\boldsymbol{\eta}\|_2} = \frac{\sum_{i,j} w_{i,j}\left|y_i - y_j + \xi_i - \xi_j\right|}{\|\mathbf{g}+\boldsymbol{\eta}\|_2} \\
&\geq \frac{\sum_{y_i \neq y_j} w_{i,j} s_{i,j}\left(y_i - y_j + \xi_i - \xi_j\right)}{\|\mathbf{g}+\boldsymbol{\eta}\|_2} + \frac{\sum_{y_i = y_j} w_{i,j}\left|\xi_i - \xi_j\right|}{\|\mathbf{g}+\boldsymbol{\eta}\|_2} \\
&\geq \frac{\sum_{y_i \neq y_j} w_{i,j} s_{i,j}\left(y_i - y_j + \xi_i - \xi_j\right)}{\|\mathbf{g}+\boldsymbol{\eta}\|_2} \\
&\quad + \frac{\left|\sum_{y_i \neq y_j} w_{i,j} s_{i,j}\left(\xi_i - \xi_j\right) - \mathcal{J}_1(\mathbf{y}) \cdot \left(2\mathbf{g}^{\mathbf{T}}\boldsymbol{\eta} + \boldsymbol{\eta}^T\boldsymbol{\eta}\right)\right|}{\|\mathbf{g}+\boldsymbol{\eta}\|_2} \\
&\geq \frac{\sum_{y_i \neq y_j} w_{i,j} s_{i,j}\left(y_i - y_j + \xi_i - \xi_j\right)}{\|\mathbf{g}+\boldsymbol{\eta}\|_2} \\
&\quad + \frac{\mathcal{J}_1(\mathbf{y}) \cdot \left(2\mathbf{g}^{\mathbf{T}}\boldsymbol{\eta} + \boldsymbol{\eta}^T\boldsymbol{\eta}\right) - \sum_{y_i \neq y_j} w_{i,j} s_{i,j}\left(\xi_i - \xi_j\right)}{\|\mathbf{g}+\boldsymbol{\eta}\|_2} \\
&= \frac{\mathcal{J}_1(\mathbf{y}) \cdot \left(2\mathbf{g}^{\mathbf{T}}\boldsymbol{\eta} + \boldsymbol{\eta}^T\boldsymbol{\eta}\right) + \mathcal{J}_1(\mathbf{y})}{\|\mathbf{g}+\boldsymbol{\eta}\|_2} = \mathcal{J}_{1'}(\mathbf{g}) \quad . \qquad (8.11)
\end{aligned}$$

□

Proof of Lemma 8.1. This proof follows similarly to the proof in the footnote of [9]. Suppose $\mathbf{y}^* \in \mathbb{R}^n$ satisfies $\mathbf{y}^*\mathbf{D}\mathbf{y}^* = 1$ and $\mathbf{y}^*\mathbf{D}\mathbf{1}^* = 0$. Then there exists a $\mathbf{g}^* \in \mathbb{R}^{n-1}$ such that $\mathbf{y}^* = \mathbf{D}^{-1/2}\mathbf{B}\mathbf{g}^*$, and Lemma 5.1 ensures that $\mathbf{g}^*$ is a unit vector. Now suppose that $\mathbf{y}^*$ does *not* solve (5.1). Then there exists a nontrivial vector $\boldsymbol{\eta} \in \mathbb{R}^{n-1}$ with $\|\mathbf{g}^* + 2\boldsymbol{\eta}\| = 1$ such that

$$\mathcal{J}_{2'}^{\boldsymbol{\Gamma}}\left(\frac{\mathbf{g}^*+\boldsymbol{\eta}}{\|\mathbf{g}^*+\boldsymbol{\eta}\|_2}\right) < \mathcal{J}_{2'}^{\boldsymbol{\Gamma}}(\mathbf{g}^*) \quad . \qquad (8.12)$$

Using the property that $\mathcal{J}_{2'}^{\boldsymbol{\Gamma}}(\beta\mathbf{x}) = \beta^2\mathcal{J}_{2'}^{\boldsymbol{\Gamma}}(\mathbf{x})$, simplifying the norm of $\mathbf{g}^* + \boldsymbol{\eta}$, and defining $\boldsymbol{\xi} = \mathbf{D}^{-1/2}\mathbf{B}\boldsymbol{\eta}$ allows us to write (8.12) as:

$$\frac{\mathcal{J}_2^{\boldsymbol{\Gamma}}(\mathbf{y}^*+\boldsymbol{\xi})}{1 + 2\boldsymbol{\eta}^T\mathbf{g}^* + \boldsymbol{\eta}^T\boldsymbol{\eta}} < \mathcal{J}_2^{\boldsymbol{\Gamma}}(\mathbf{y}^*) \quad . \qquad (8.13)$$

This inequality simplifies to:

$$\begin{aligned}
&2\sum_{i,j} w_{i,j} s_{i,j}\left(\xi_i - \xi_j\right) + \mathcal{J}_{2'}^{\boldsymbol{\Gamma}}(\boldsymbol{\eta}) \\
&\qquad < \left(2\boldsymbol{\eta}^T\mathbf{g}^* + \boldsymbol{\eta}^T\boldsymbol{\eta}\right) \cdot \mathcal{J}_{1'}(\mathbf{g}^*) \quad , \qquad (8.14)
\end{aligned}$$

which, after noting that $2\boldsymbol{\eta}^T\mathbf{g}^* + \boldsymbol{\eta}^T\boldsymbol{\eta} = \boldsymbol{\eta}^T\mathbf{g}^*$ and employing Lemma 8.2, reduces to:

$$\mathcal{J}_{2'}^{\boldsymbol{\Gamma}}\left(\frac{\boldsymbol{\eta}}{\|\boldsymbol{\eta}\|}\right) < 0 \quad , \qquad (8.15)$$

a contradiction. Hence, $\mathbf{y}^*$ solves (5.1). □

8.3 Computing $\mathbf{B}^{\mathbf{T}}\mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2}\mathbf{B}\mathbf{x}$

Define the $n \times (n-1)$ matrix:

$$\mathbf{C} = \left[\hat{\mathbf{q}} \,|\, -q_1 \mathbf{I}_{n-1}\right]^{\mathbf{T}} \ , \tag{8.16}$$

where $\mathbf{q} \in \mathbb{R}^n$ is the unit vector in the direction of $\mathbf{D}^{1/2}\mathbf{1}$, $\hat{\mathbf{q}} \in \mathbb{R}^{n-1}$ is defined by $\hat{\mathbf{q}} = [q_2, q_3, \ldots, q_n]^{\mathbf{T}}$, and $\mathbf{I}_{n-1}$ is the $(n-1) \times (n-1)$ identity matrix. We point out a few things related to $\mathbf{C}$. First, note that

$$\mathbf{C}^{\mathbf{T}}\mathbf{C} = q_1^2 \mathbf{I}_{n-1} + \hat{\mathbf{q}}\hat{\mathbf{q}}^{\mathbf{T}} \ , \tag{8.17}$$

and so

$$\left(\mathbf{C}^{\mathbf{T}}\mathbf{C}\right)^{1/2} = q_1 \mathbf{I}_{n-1} + \frac{\hat{\mathbf{q}}\hat{\mathbf{q}}^{\mathbf{T}}}{1+q_1} \ . \tag{8.18}$$

Hence, by the Sherman-Morrison formula, we can write:

$$\left(\mathbf{C}^{\mathbf{T}}\mathbf{C}\right)^{-1/2} = q_1^{-1} \mathbf{I}_{n-1} - \frac{\hat{\mathbf{q}}\hat{\mathbf{q}}^{\mathbf{T}}}{q_1(1+q_1)} \ . \tag{8.19}$$

Since we have now established by construction that $\left(\mathbf{C}^{\mathbf{T}}\mathbf{C}\right)^{-1/2}$ exists, we have that:

$$\left(\mathbf{C}^{\mathbf{T}}\mathbf{C}\right)^{-1/2}\mathbf{C}^{\mathbf{T}} = \left[\hat{\mathbf{q}} \,|\, -\mathbf{I}_{n-1} + \hat{\mathbf{q}}\hat{\mathbf{q}}^{\mathbf{T}}/(1+q_1)\right] \ . \tag{8.20}$$

Hence, $\mathbf{z}_3 = \mathbf{B}^{\mathbf{T}}\mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2}\mathbf{B}\mathbf{x}$ can be computed without constructing any dense matrix by performing the following steps:

1. $\mathbf{z}_1 = q_1^{-1}\mathbf{z} - \left(\dfrac{\hat{\mathbf{q}}^{\mathbf{T}}\mathbf{z}}{q_1\,(1+q_1)}\right)\hat{\mathbf{q}}$
2. $\mathbf{z}_2 = \mathbf{C}^{\mathbf{T}}\left(\mathbf{D}^{-1/2}\left(\mathbf{L}\left(\mathbf{D}^{-1/2}\left(\mathbf{C}\mathbf{z}_1\right)\right)\right)\right)$
3. $\mathbf{z}_3 = q_1^{-1}\mathbf{z}_2 - \left(\dfrac{\hat{\mathbf{q}}^{\mathbf{T}}\mathbf{z}_2}{q_1\,(1+q_1)}\right)\hat{\mathbf{q}}$

References

[1] P. Arbeláez, M. Maire, C. Fowlkes, and J. Malik. Contour detection and hierarchical image segmentation. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 33(5):898–916, May 2011.

[2] P. Arbelaez, J. Pont-Tuset, J. T. Barron, F. Marques, and J. Malik. Multiscale combinatorial grouping. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.

[3] R. Barrett, M. W. Berry, T. F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. Van der Vorst. *Templates for the solution of linear systems: building blocks for iterative methods*, volume 43. Siam, 1994.

[4] M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural computation*, 15(6):1373–1396, 2003.

[5] L. M. Bregman. The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming. *USSR computational mathematics and mathematical physics*, 7(3):200–217, 1967.

[6] P. K. Chan, M. D. F. Schlag, and J. Y. Zien. Spectral k-way ratio-cut partitioning and clustering. *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, 13(9):1088–1096, 1994.

[7] Y. Chen, D. Dai, J. Pont-Tuset, and L. Van Gool. Scale-aware alignment of hierarchical image segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 364–372, 2016.

[8] S. E. Chew and N. D. Cahill. Semi-supervised normalized cuts for image segmentation. In *Proc. International Conference on Computer Vision*, pages 1716–1723, December 2015.

[9] I. Daubechies, R. DeVore, M. Fornasier, and C. S. Güntürk. Iteratively reweighted least squares minimization for sparse recovery. *Communications on Pure and Applied Mathematics*, 63(1):1–38, 2010.

[10] A. Eriksson, C. Olsson, and F. Kahl. Normalized cuts revisited: A reformulation for segmentation with linear grouping constraints. *Journal of Mathematical Imaging and Vision*, 39(1):45–61, 2011.

[11] T. Goldstein and S. Osher. The split bregman method for l1-regularized problems. *SIAM Journal on Imaging Sciences*, 2(2):323–343, 2009.

[12] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *International journal of computer vision*, 1(4):321–331, 1988.

[13] R. Lai and S. Osher. A splitting method for orthogonality constrained problems. *Journal of Scientific Computing*, 58(2):431–449, 2014.

[14] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3431–3440, 2015.

[15] S. Maji, N. K. Vishnoi, and J. Malik. Biased normalized cuts. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, CVPR, pages 2057–2064, 2011.

[16] R. Meinhold, T. Hayes, and N. Cahill. Efficiently computing piecewise flat embeddings for data clustering and image segmentation. In *Proc. IEEE MIT Undergraduate Research and Technology Conference*, pages 1–4, 2016.

[17] A. Y. Ng, M. I. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. *Advances in Neural Information Processing Systems*, 2:849–856, 2002.

[18] J. A. Sethian. *Level set methods and fast marching methods: evolving interfaces in computational geometry, fluid mechanics, computer vision, and materials science*, volume 3. Cambridge university press, 1999.

[19] J. Shi and J. Malik. Normalized cuts and image segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, CVPR, pages 731–737, Jun 1997.

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

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

[22] L. Vincent and P. Soille. Watersheds in digital spaces: an efficient algorithm based on immersion simulations. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 13(6):583–598, 1991.

[23] Z. Wu and R. Leahy. An optimal graph theoretic approach to data clustering: theory and its application to image segmentation. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 15(11):1101–1113, Nov 1993.

[24] S. Xie and Z. Tu. Holistically-nested edge detection. In *Proc. International Conference on Computer Vision*, pages 1395–1403, 2015.

[25] S. X. Yu and J. Shi. Multiclass spectral clustering. In *Proc. International Conference on Computer Vision*, ICCV, pages 313–319. IEEE, 2003.

[26] S. X. Yu and J. Shi. Segmentation given partial grouping constraints. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 26(2):173–183, Feb 2004.

[27] Y. Yu, C. Fang, and Z. Liao. Piecewise flat embedding for image segmentation. In *Proc. International Conference on Computer Vision*, pages 1368–1376, 2015.

[28] Z. Zhang and M. I. Jordan. Multiway spectral clustering: A margin-based perspective. *Statistical Science*, 23(3):383–403, 2008.

[29] Q. Zhao and L. D. Griffin. Better image segmentation by exploiting dense semantic predictions. *arXiv preprint arXiv:1606.01481*, 2016.