Adaptive Superpixel Cuts for Hyperspectral Images

Aleksandar Popovic

February 2024

Abstract

Blind segmentation in hyperspectral images is a challenging problem. Many traditional methods suffer from poor identification of materials and expensive computational costs, which can be partially eased by trading the accuracy with efficiency.

In this paper, we propose a novel graph-based algorithm for segmentation in hyperspectral images. Utilizing the fact that pixels in a local region are likely to have similar spectral features, a pre-clustering algorithm is used to extract the homogenous regions, called superpixels. After extracting the superpixels, a weighted graph is constructed with the weights representing both the spectral similarity and spatial distance between each superpixel and its neighbors. The normalized graph cuts algorithm is then used to perform an initial segmentation of the image. To effectively extract the material information in the superpixels, the mean spectra in each segment is used to estimate the abundance of each endmember in each superpixel using a graph regularized hyperspectral unmixing algorithm. The resulting abundance information is used as a supportive feature, which when combined with the spectral features, form a new spectral feature vector for each superpixel. Using this new feature vector, the weighted graph is once again constructed and the normalized cuts algorithm is applied, resulting in a final segmentation of the image.

Experiments on a real hyperspectral datasets illustrate great potential of the proposed method in terms of accuracy and efficiency.

Contents

1 Introduction

1.1 Motivations

1.2 Applications to Biomedical Imaging

1.3 Hyperspectral Data

2 Background

Analysis in hyperspectral images has traditionally been a computationally expensive and difficult task due to algorithms scaling in both the spatial and spectral resolution of the images. This section will focus on building a relevant background for common preclustering, abundance estimation, and segmentation techniques from the perspective of imaging that will be later be adapted for use in the proposed algorithm.

2.1 The Hyperspectral Cube

In traditional, RGB based imaging systems, an image can be represented by a 3-dimensional tensor of shape $(n_x, n_y, 3)$, where the last dimension corresponds to the color channel the image was captured in. From a mathematical point of view, a hyperspectral image, denoted by $\mathbf{X}$, is a tensor of shape (n_x, n_y, n_λ) with nonnegative entries. Each pixel in the tensor is represented using a vector $\mathbf{x} \in \mathbb{R}_+^{n_\lambda}$. From a physical point of view, the first two dimensions in $\mathbf{X}$ represent the spatial coordinates of the pixels, while the last dimension represents the specific wavelength band the spectral intensity, reflectance or transmittance measurements were taken at.

2.2 Superpixel Generation

As mentioned in Section (2.1), high spectral resolution is a common constraint in hyperspectral image analysis, with time complexity of the algorithms commonly scaling polynomially with the number of pixels in the image.

Before the introduction of neural network based solutions in computer vision applications, there was interest in preclustering images into locally homogeneous regions called superpixels [superpixel ref]. Addressing the main motivation of reducing the overall granularity of the data, superpixels are also shown to preserve spectral information, adhere to spatial features in the image, and introduce robustness against noise in subsequent analysis tasks.

2.2.1 Simple Linear Iterative Clustering

In this section, we will introduce the Simple Linear Iterative Clustering (SLIC) algorithm. The algorithm is a special case of the k-means algorithm adapted to the task generating superpixels in a 5-dimensional space, where the first 3 dimensions correspond to the the pixel color vector in the CIELAB colorpsace, and last 2 dimensions correspond to the spatial coordinates (i, j) of the pixel in the image.. Formally, each pixel $\mathbf{x}_{(i,j)} = [x_l, x_a, x_b]$ is restructured into the form $\tilde{\mathbf{x}}_{(i,j)} = [x_l, x_a, x_b, i, j]$. With this modified feature vector $\tilde{\mathbf{x}}$, we incorporate both spectral and spatial information into the clustering, however, while the spectral information has bounds on it's values, the spatial information depends on the size of the image.

Taking as an input the desired number of superpixels n_s, for an image with $n_p = n_x n_y$ pixels, each superpixel would be composed of approximately n_s/n_p pixels. Assuming the superpixels lie on a grid, a superpixel centroid would occur at every grid interval $S = \sqrt{n_s/n_p}$. At the onset of the algorithm, a grid of n_s superpixel centers $\mathbf{C}_n = [c_l, c_a, c_b, i, j]$ where $n = 1, \cdots, n_s$ are sampled across the image with regular grid intervals S. To avoid sampling noisy pixels, clusters are moved to the lowest gradient position in a 3×3 neighborhood where the image gradient is calcuated, using the original spectral vector x in the CIELAB color space as:

$$\mathbb{G}(i, j) = \|\mathbf{x}_{(i+1,j)} - \mathbf{x}_{(i-1,j)}\|^2 + \|\mathbf{x}_{(i,j+1)} - \mathbf{x}_{(i,j-1)}\|^2 \tag{1}$$

After initialization, a modified distance measure is proposed to enforce color similarity and spatial extent within the superpixels. Since the approximate area of each superpixel is S^2, it is assumed that pixels associated with a superpixel lie within a $2S \times 2S$ neighborhood of the superpixel centroid. Introducing the parameter m to control the compactness and shape of the superpixels, the modified distance is then calculated as

$$\mathbb{D}(x, y) = \|\mathbf{x}_{lab} - \mathbf{y}_{lab}\|^2 + \frac{m}{S}\|\mathbf{x}_{ij} - \mathbf{y}_{ij}\|^2 \tag{2}$$

Each pixel is the image is associated with the nearest cluster whose search area overlaps this pixel. After all pixels are associated with a cluster, a new center is computed as the average feature vector of all the pixels belonging to the cluster. This is repeated for a set number of iterations. After exhausting all iterations, a final step is performed by relabelling disjoint segments with the labels of the largest neighboring cluster. This step is optional as disjoint segments tend to not occur for larger inputs of n_s and m.

Algorithm 1: SLIC Superpixel Algorithm

Input: $m > 0$, $n_s > 0$, $n_{\text{iters}} > 0$, CIELAB Image $\mathbf{X}$.
Initialize: $\mathbf{c}_n = [c_l, c_a, c_b, i, j]$ where $n = 1, \cdots, n_s$ by sampling pixels at regular grid intervals S. Perturb cluster centers to lowest gradient position in a 3×3 neighborhood according to (1)
for $k = 1$ **to** n_{iters} **do**
  Assign best matching pixels from a $2S \times 2S$ neighborhood around clusters C_k according to distance measure (2).
  Compute new cluster centers according to average vector of all pixels belonging to cluster.
end
Optional: Relabel disjoint segments.
Output: Superpixeled Image $\mathbf{X}_s = [\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_{n_s}]$

The SLIC algorithm is shown to produce meaningful and noise-robust segments in traditional computer vision applications. This algorithm proves useful in Section 3.2 when adapted as a pre-clustering step in the hyperspectral domain.

2.3 Hyperspectral Unmixing

In hyperspectral imaging applications, there is emphasis on estimating the relative abundance of a given representantive materical, called an endmember, within each pixel.

Unmixing results often give more detailed information about the overall composition of a hyperspectral scene with respect to simple segmentation. This section will introduce the foundational knowledge behind unmixing and abundance estimation and derive an ADMM based approach to abundance estimation.

2.3.1 Linear Mixing Model

In reality, most pixels in a hyperspectral image capture a mixture of spectra reflected from various materials present within the spatial area, due to constraints with how large a spatial resolution can be acheived.

Figure 1: Image of LMM goes here!

The foundational model behind hyperspectral unmixing is the linear mixing model, which dictates that spectra of every pixel $\mathbf{x} \in \mathbb{R}_+^{n_b}$ in a hyperspectral image is a linear combination of a set of n_e spectra, $\mathbf{m}_1, \mathbf{m}_2, \cdots, \mathbf{m}_{n_e} \in \mathbb{R}_+^{n_b}$, from pure representative materials, called endmembers, with weights $a_1, a_2, \cdots, a_{n_e} \in \mathbb{R}$. Denoting $\mathbf{M} = [\ \mathbf{m}_1 \mid \mathbf{m}_2 \mid \cdots \mid \mathbf{m}_{n_e}\ ] \in \mathbb{R}_+^{n_b \times n_e}$ and $\mathbf{a} = [a_1, a_2, \cdots, a_{n_e}]^T \in \mathbb{R}^{n_e}$, the linear mixing model is formulated as follows:

$$\mathbf{x} = \mathbf{M}\mathbf{a} + \epsilon. \tag{3}$$

While this model serves useful, there is no direct physical interpretation to the weights in $\mathbf{a}$, instead, we aim to estimate the physical proportion, called the abundance, of each endmember within each pixel by imposing two constraints on the entries in $\mathbf{a}$. The abundance nonnegativity constraint (ANC) requires that the entries in $\mathbf{a}$ must be greater than or equal to zero, while the abundance sum-to-one constraint (ASC) requires that the entries in $\mathbf{a}$ sum to 1. Combining the two constraints, we have an extension of the linear mixing model

$$\mathbf{x} = \mathbf{M}\mathbf{a} + \epsilon \quad \text{s.t } \mathbf{a} \in \mathbb{R}_+^{n_e} \text{ and } \|\mathbf{a}\|_1 = 1. \tag{4}$$

The linear mixing model can be additionally be extended from a per pixel basis onto a collection of n_p pixels $\mathbf{X} = [\ \mathbf{x}_1 \mid \mathbf{x}_2 \mid \cdots \mid \mathbf{x}_{n_p}\ ] \in \mathbb{R}_+^{n_b \times n_p}$, with each pixel $\mathbf{x}_i$ having a corresponding abundance vector $\mathbf{a}_i$. Arranging the abundance vectors into an abudance matrix $\mathbf{A} = [\ \mathbf{a}_1 \mid \mathbf{a}_2 \mid \cdots \mid \mathbf{a}_{n_p}\ ] \in \mathbb{R}^{n_e \times n_p}$, we denote the ANC-ASC constraint using the set $\Delta = \{\mathbf{A} \in \mathbb{R}_+^{n_e \times n_p} \mid \mathbf{1}_{n_e}^T \mathbf{A} = \mathbf{1}_{n_p}\}$. This new extension of the linear mixing model to a collection of pixels that will be used for the following sections

$$\mathbf{X} = \mathbf{M}\mathbf{A} + \epsilon \quad \text{s.t } \mathbf{A} \in \Delta \tag{5}$$

The linear mixing model is an objectively simple model, which enforces a linear relationship between the spatial mixing of endmembers through assuming that pixels lie on a flat plane. This is almost never the case, however the model remains a efficient and powerful tool for extracting spectral information from a scene.

2.3.2 Abundance Estimation

Given the linear mixing model as formulated in Section 2.3.1, in traditional hyperspectral imaging tasks, both $\mathbf{M}$ and $\mathbf{A}$ are unknown. Often, researchers aim to estimate $\mathbf{M}$ first, as spectral signatures collected from endmembers in same scene under the same conditions will be almost identical. Notably, in the field of remote sensing, effort has been made to create a library of spectral signatures derived from common vegetation and minerals in land cover images, allowing focus to be made solely in estimating $\mathbf{A}$ [REF]. This section will cover the scenario where $\mathbf{M}$ is known and $\mathbf{A}$ is to be estimated.

The task is referred to as abundance estimation and continues to be an active area of research, where the aim is to find $\mathbf{A}$ such that an error function $\mathcal{L}$ is minimized with respect to the reconstructed collections of pixels $\tilde{\mathbf{X}} = \mathbf{M}\mathbf{A}$ and the original collection of pixels $\mathbf{X}$. Traditionally, we aim to minimize the least-square reconstruction error between the entries in $\tilde{\mathbf{X}}$ and $\mathbf{X}$

$$\mathcal{L}(\mathbf{X}, \tilde{\mathbf{X}}) = \sum_{i=1}^{n_b} \sum_{j=1}^{n_p} \left(\mathbf{x}_{(i,j)} - \tilde{\mathbf{x}}_{(i,j)}\right)^2 = \|\tilde{\mathbf{X}} - \mathbf{X}\|_F^2. \tag{6}$$

The least-squares reconstruction error can alternatively be written as the squared Frobenius norm, denoted as $\|\cdot\|_F^2$, of the difference between $\tilde{\mathbf{X}}$ and $\mathbf{X}$. This choice of $\mathcal{L}$ is the straightforward approach as $\mathcal{L}$ is both convex and differentiable, with the additional properties that $\mathcal{L}(\tilde{\mathbf{X}}, \mathbf{X}) = \mathcal{L}(\mathbf{X}, \tilde{\mathbf{X}})$ and $\mathcal{L}(\mathbf{X}, \tilde{\mathbf{X}}) = \mathcal{L}(\mathbf{X}^T, \tilde{\mathbf{X}^T})$ [REF].

To incorporate the ANC-ASC constraint into the overall formulation of $\mathcal{L}$, the set Δ from Section 2.3.1 proves useful. It is important to note that Δ is a convex set, meaning that for matrices $A, B \in \Delta$, for all $0 \leq \alpha \leq 1$, the matrix $C = \alpha A + (1-\alpha)B$ is also an element of Δ. The inclusion of the constraints on $\mathbf{A}$ is facilitated using the piecewise function χ_S defined as follows

$$\chi_S(x) = \begin{cases} 0 & \text{if } x \in S \\ \infty & \text{if } x \notin S. \end{cases} \tag{7}$$

Adding χ_Δ in the formulation of $\mathcal{L}$ restricts the values $\mathbf{A}$ can take on to the set Δ while ensuring that the overall formulation still has a global minimum within Δ. Additionally, a regularization term J can be added to impose other constraints on the values in $\mathbf{A}$. Later sections operate on the assumption that J is also convex. Formally, abudance estimation can be formualated as a convex optimization problem of the form

$$\mathbf{A} = \underset{\mathbf{A} \in \mathbb{R}^{n_e \times n_p}}{\arg\min} \frac{1}{2}\|\mathbf{M}\mathbf{A} - \mathbf{X}\|_F^2 + \chi_\Delta(\mathbf{A}) + J(\mathbf{A}). \tag{8}$$

This problem has no closed form solution, relying on iterative methods or applying solvers like GUROBI [REF] or CVXOPT [REF] to solve the problem for individual pixels given the problem can be split pixelwise. The formulation of the abundance estimation allows for flexibility in choice of $\mathcal{L}$. There is particular interest in choosing $\mathcal{L}$ such that it is convex and differentiable, in order to apply gradient-based methods to estimate the abundance matrix, however there exist approaches that choose rely on divergence-based or non differentiable metrics [REF].

2.4 Alternating Direction Method of Multipliers

Alternating Direction Method of Multipliers, or ADMM, introduced by Boyd et al. [REF] is a framework for solving convex optimization problems of the form:

$$\begin{aligned} &\text{minimize} \quad f(x) + g(z) \\ &\text{subject to} \quad Ax + Bz = c \end{aligned} \tag{9}$$

with variables $x \in \mathbb{R}^n$ and $z \in \mathbb{R}^m$, where $A \in \mathbb{R}^{p \times n}$, $B \in \mathbb{R}^{p \times m}$ and $c \in \mathbb{R}^p$. f and g are assumed to be convex. The aim of ADMM is to incorporate the decomposability of the dual ascent method into the superior convergence properties of method of multipliers. To allow for this, ADMM introduces the corresponding augmented Lagrangian $\mathcal{L}_\mu$ defined as:

$$\mathcal{L}_\mu(x, z, y) = f(x) + g(z) + y^T(Ax + Bz - c) + \frac{\mu}{2}\|Ax + Bz - c\|_2^2 \tag{10}$$

where $\mu > 0$ is augmented lagrangian convergence parameter and $y \in \mathbb{R}^p$ is the corresponding dual variable. Scaling with $u = \frac{1}{\mu}y$ gives the following equivalent definition:

$$\mathcal{L}_\mu(x, z, u) = f(x) + g(z) + \frac{\mu}{2}\|Ax + Bz - c + u\|_2^2. \tag{11}$$

ADMM aims to minimize scaled form of $\mathcal{L}_\mu$ by alternating minimizations with respect to x, y, and u by performing the following updates:

$$\begin{aligned} x^{(k+1)} &= \arg\min_x \ \mathcal{L}_\mu(x, z^{(k)}, u^{(k)}) \\ z^{(k+1)} &= \arg\min_z \ \mathcal{L}_\mu(x^{(k+1)}, z, u^{(k)}) \\ u^{(k+1)} &= u^{(k)} + Ax^{(k+1)} + Bz^{(k+1)} - c. \end{aligned} \tag{12}$$

Under mild conditions on f and g, ADMM can be shown to provide guarenteed objective and residual convergence, independent on choice of μ. For lax choices of μ, the algorithm provides modest accuracy solutions in a relatively low number of iterations, favorable to tasks in statistical learning where parameter estimation often yields little improvement to results. The algorithm allows practioners to put focus on efficient implementations to the minimization problems for x and z, which proves useful in the following section.

2.5 Spectral Clustering

Clustering aims to partition unlabelled data into a set of groupings called clusters such that a predefined similarity metric is minimized within the data points in the cluster and maximized between clusters. Traditional clustering methods such as k-means, and tree based methods suffer in situations where both spatial and spectral information must be taken into account for computing clusters and often are sensitive to initialization and outliers in data. The focus of this section is to introduce the concept of spectral clustering, which aims to partition a set of data points into cluster by storing similarity between data points in a graph structure then using spectral analysis techniques to calculate globally optimal partitions.

The particular focus will be in the context of imaging, and particularly hyperspectral imaging, the final product of a clustering algorithm should be perceptually meaningful groupings that respect both the spectral and spatial features in the image. Considering a collection of pixels $\mathbf{X} = [\ \mathbf{x}_1 \mid \mathbf{x}_2 \mid \cdots \mid \mathbf{x}_{n_p}\ ] \in \mathbb{R}_+^{n_b \times n_p}$ and a symmetric similarity measure d, the affinity matrix $\mathbf{W} \in \mathbb{R}_+^{n_p \times n_p}$ is constructed as

$$\mathbf{W}_{(i,j)} = d(\mathbf{x}_i, \mathbf{x}_j). \tag{13}$$

Typical choices for d in imaging applications include calculating the euclidean norm and the cosine angle between the spectral features of $\mathbf{x}_i$ and $\mathbf{x}_j$. The euclidean distance calculates the difference in magnitude between the two pixels, leading to sensitivity under different lighting conditions. Cosine angle calculates the relative angle between the two pixel vectors, with 0 indicating that the pixels are exactly identical or one of them is a scaled version of the other. Cosine angle is often the metric of choice due to its scale invariance property, allowing for better distinction of materials in different lighting conditions.

$$\begin{aligned} d_{L_2}(\mathbf{x}_i, \mathbf{x}_j) &= \|\mathbf{x}_i - \mathbf{x}_j\|_2 \\ d_\theta(\mathbf{x}_i, \mathbf{x}_j) &= \arccos\left(\frac{\mathbf{x}_i \mathbf{x}_j^T}{\|\mathbf{x}_i\|_2 \|\mathbf{x}_j\|_2}\right) \end{aligned} \tag{14}$$

As $d_{L_2} \in [0, \infty)$ and $d_\theta \in [0, \pi]$, the affinity matrix $\mathbf{W}$ can alternatively be constructed using the heat kernel matrix with $0 < \sigma < 1$, this pushes similar pixels to have $\mathbf{W}(i,j) = 0$ and similar pixels to have $\mathbf{W}(i,j) = 1$.

$$\mathbf{W}_{(i,j)} = \exp\left(-\frac{d(\mathbf{x}_i, \mathbf{x}_j)^2}{\sigma^2}\right). \tag{15}$$

A graph $G = (V, E)$ is a set of vertices V and edges E that connect them. Considering the set of pixels $\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_{n_p}\}$ as the set of vertices and $d(\mathbf{x}_i, \mathbf{x}_j)$ as the edges between them, $\mathbf{W}$ is the matrix representation of a graph G_W. Spectral Analysis is then the study of $\mathbf{W}$ using linear algebra techniques to determine insights on the structure of G_W using the eigenvalues and eigenvectors of $\mathbf{W}$. A fundamental matrix in spectral analysis is the graph Laplacian matrix $\mathbf{L}$, calculated as the difference between the diagonal matrix $\mathbf{D}$ where

$$\mathbf{D}_{(i,j)} = \begin{cases} \sum_j \mathbf{W}_{(i,j)} & \text{if } i = j, \\ 0 & \text{if } i \neq j \end{cases} \tag{16}$$

and $\mathbf{W}$. Formally,

$$\mathbf{L} = \mathbf{D} - \mathbf{W}. \tag{17}$$

As $\mathbf{L}$ is known to be postive semidefinite, the eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_n$ are strictly non-negative and provide key insights into the structure of G.

2.5.1 Normalized Cuts

A graph $G = (V, E)$ with affinity matrix $\mathbf{W}$ can be partitioned into two subgraphs $G_A = (V_A, E_A)$ and $G_B = (V_B, E_B)$ such that $V_A \cup V_B = V$ and $V_A \cap V_B = \emptyset$ by removing the edges between the vertices in G_A and G_B. The dissimilarity between these two graphs can be calculated as the sum of the edges cut to form the partitions

$$\text{cut}(G_A, G_B) = \sum_{i \in V_A,\ j \in V_B} \mathbf{W}_{(i,j)} \tag{18}$$

The optimal bipartitioning of G_W is given as the graphs G_A and G_B that minimize (18). However, in the case of image segmentation, this criteria will heavily prioritize partitioning single pixels from the image. Instead, the

normalized cuts criteria [REF] is proposed, focusing on balancing the ratio between the edges cut and the sum of the internal edge nodes within G_A and G_B defined as

$$\text{ncut}(G_A, G_B) = \frac{\text{cut}(G_A, G_B)}{\text{assoc}(G_A, G)} + \frac{\text{cut}(G_A, G_B)}{\text{assoc}(G_B, G)} \tag{19}$$

where

$$\text{assoc}(G_A, G) = \sum_{i \in V_A,\, j \in V} \mathbf{W}_{(i,j)}. \tag{20}$$

The normalized cuts criteria, in general terms, aims to minimize the disassociation between the subgraphs and maximize the association within them. While (19) is NP-Complete, Shi et. al [REF] show that solving for the eigenvector $\mathbf{u}_2$ corresponding to the second smallest eigenvalue λ_2 in the system

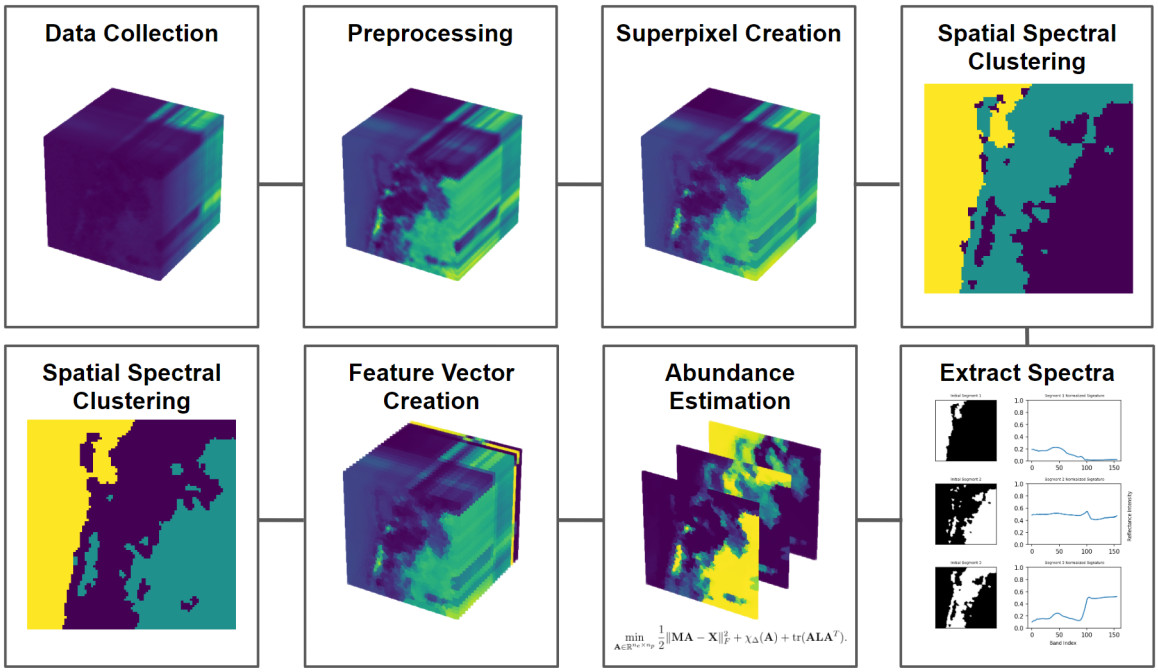
$$\mathbf{D}^{-\frac{1}{2}}(\mathbf{D} - \mathbf{W})\mathbf{D}^{-\frac{1}{2}}\mathbf{z} = \lambda \mathbf{z} \tag{21}$$

provides a approximate real valued solution to (19) through assigning subgraph membership of the vertices according to the sign of the entries in $\mathbf{u}_2$.

Spectral graph techniques like the normalized cuts algorithm provide the advantage of flexibile and deterministic results based on the initialization of the adjacency matrix $\mathbf{W}$, however begin to fall behind iterative methods when the graph is large, due to the time complexity of solving (21) scaling cubically with the number of vertices.

3 Adaptive Superpixel Cuts

This section focuses on introducing the proposed unsupervised hyperspectral image segmentation algorithm Adaptive Superpixel Cuts for Hyperspectral Images (ASC-HSI). This algorithm leverages the algorithms proposed in Section 2 to perform image segmentation on a superpixel-basis rather than a pixel-basis, reducing computational and memory costs while allowing for the use of a graph based approach to segmentation and unmixing. The steps of the algorithm can depicted as follows:



3.1 Dataset Preprocessing

As mentioned in Section 2.1, the input to the algorithm is a hyperspectral image represented by a nonnegative tensor $\mathbf{X}$ of shape (n_x, n_y, n_λ). Raw hyperspectral images are susceptible to outliers and scale invariance across wavelengths. As such, to ensure algorithms perform reliably along a wide range of domains, preprocessing is a crucial step. Typically, the first main goal of practioners is to deal with noise across the spectral dimension, then work to deal with spatial artifacts in the image. This section will cover the Singular Value Decomposition and Layer Normalization methods that form the basis of the spectral preprocessing methods for the algorithm.

3.1.1 Singular Value Decomposition

do i even need this to be honest?

3.1.2 Layer Normalization

Hyperspectral images often have bands with significantly higher intensity values compared to others. It is important to consider the full range of wavelengths rather than allow the overprioritization of higher intensity wavelengths, especially in situations where direct scale comparison is made between pixels. By normalizing each spectral band to a similar scale, all spectral bands contribute more equally to the segmentation process. This ensures that results capture the underlying spectral information more effectively.

A common approach to normalization in hyperspectral images is band normalization, where intesity values for the hyperspectral image at spectral band k, denoted $\mathbf{X}_{(\cdot,\cdot,k)}$ are independently scaled to $[0, 1]$ according to the minimum and maximum values at that band. Formally,

$$\hat{\mathbf{X}}_{(\cdot,\cdot,k)} = \frac{\mathbf{X}_{(\cdot,\cdot,k)} - \min \mathbf{X}_{(\cdot,\cdot,k)}}{\max \mathbf{X}_{(\cdot,\cdot,k)} - \min \mathbf{X}_{(\cdot,\cdot,k)}}. \tag{22}$$

3.2 Hyperspectral Superpixel Generation

In 2.2.1, the SLIC algorithm was introduced, aiming to perceptually group pixels into locally homogeneous groups called superpixels. While the SLIC algorithm was originally developed for use in the CIELAB colorspace, a similar methodology can be applied towards the hyperspectral space. The main change when trasitioning to the hyperspectral domain requires the consideration of all the spectral features.

With the preprocessed hyperspectral image $\hat{\mathbf{X}}$, each pixel is given by the vector $\hat{\mathbf{x}}_{(i,j)} = [\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_{n_\lambda}]$. Then, taking as an input the number of superpixels n_s, centers $\mathbf{c}_n = [c_1, c_2, \ldots, c_{n_\lambda}]$ where $n = 1, 2 \ldots, n_s$ are created at regular grid intervals $S = \sqrt{n_s/n_p}$ across the image. The initial clusters are moved to the lowest gradient position in a 3×3 spatial neighborhood where the image gradient is now calculated using the original hyperspectral features instead of the CIELAB features:

$$\mathbb{G}(i,j) = \|\hat{\mathbf{x}}_{(i+1,j)} - \hat{\mathbf{x}}_{(i-1,j)}\|_2^2 + \|\hat{\mathbf{x}}_{(i,j+1)} - \hat{\mathbf{x}}_{(i,j-1)}\|_2^2 \tag{23}$$

Following the original formulation of the SLIC algorithm, a modified distance measure is proposed to enforce color similarity and spatial extent within the superpixels. Using the same parameter m to control the compactness and shape of the superpixels, the modified distance between a pixel $\hat{\mathbf{x}}$ and cluster $\mathbf{c}_n$ is now calculated as L_2 difference between the spectral features plus a scaled version of the spatial euclidean distance between the pixel and the cluster center:

$$\mathbb{D}(\hat{\mathbf{x}}, \mathbf{c}_n) = \|\hat{\mathbf{x}} - \mathbf{c}_n\|_2^2 + \frac{m}{S} d_{\text{spatial}}(\hat{\mathbf{x}}, \mathbf{c}_n)^2 \tag{24}$$

Each pixel is associated with the nearest cluster $\mathbf{c}_n$ whose search area overlaps the pixel. After all pixels are associated with a cluster, a new cluster center is computed as the average vector of all the pixels belonging to the cluster. This is repeated for a set number of iterations. In the hyperspectral version of the SLIC algorithm, the option of relabelling disjoint segments is not performed, instead opting for higher selection of the m and n_s parameters to avoid disjoint segments all together. Once the algorithm is completed, the final superpixeled image is given by arranging the feature vectors into columns of the matrix $\mathbf{C} = [\mathbf{c}_1 | \mathbf{c}_2 | \ldots | \mathbf{c}_{n_s}]$.

Algorithm 2: Hyperspectral SLIC Algorithm

Input: $m > 0$, $n_s > 0$, $n_{\text{iters}} > 0$, Preprocessed Hyperspectral Image $\hat{\mathbf{X}}$.
Initialize: $\mathbf{c}_n = [c_l, c_a, c_b]$ where $n = 1, \cdots, n_s$ by sampling pixels at regular grid intervals S. Perturb cluster centers to lowest gradient position in a 3×3 neighborhood according to (23).
for $k = 1$ **to** n_{iters} **do**
  Assign best matching pixels from a $2S \times 2S$ neighborhood around clusters $\mathbf{c}_k$ according to (24).
  Compute new cluster centers according to average of all pixels belonging to cluster.
end
Output: Superpixeled Image Matrix $\mathbf{C}$

In the hyperspectral domain, superpixels are slightly less adept at creating visually meaningful partitionings due to use of the raw hyperspectral spectra rather than a perceptual colorspace like CIELAB. To alleviate this, higher values of m and n_s are used to form more spatially compact regions akin to the ones formed in the original algorithm. Nonetheless, superpixels are valuable in allowing practioners to avoid having to consider the variation between individual pixels and instead consider the variation between these new perceptual groupings of pixels. Superpixels are exceptionally useful for dealing with artifacts or imaging deficits in simple instances.

SHOW RESULTS OF SUPERPIXELING RESULTS FOR DIFFERENT VALUES OF NS AND NP

3.3 Spatial Spectral Clustering

In Section 2.5, the Normalized Cuts algorithm was introduced for the task of bipartitioning a group of pixels through creating an affinity matrix and solving the relaxed eigensystem in (21). Considering a matrix of superpixels $\mathbf{C}$ from the results of the SLIC algorithm in Section 3.2, two matrices are formed. The first matrix is the spectral affinity matrix given by calculating the cosine similarity between the spectral features of the superpixels:

$$\mathbf{W}_{\text{spectral}_{(i,j)}} = \arccos\left(\frac{\mathbf{c}_i\mathbf{c}_j^T}{\|\mathbf{c}_i\|_2\|\mathbf{c}_j\|_2}\right). \tag{25}$$

The second matrix is the spatial distance matrix given by calculating the spatial euclidean distance between the superpixels:

$$\mathbf{W}_{\text{spatial}_{(i,j)}} = d_{\text{spatial}}(\mathbf{c}_i, \mathbf{c}_j). \tag{26}$$

To combine the spatial and spectral information within the image, a spectral similarity parameter $\sigma > 0$ and spatial limit parameter $\kappa > 0$ are introduced and the spatial-spectral affinity matrix $\mathbf{W}$ is then constrcuted as follows

$$\mathbf{W}_{(i,j)} = \begin{cases} \exp\left(-\frac{\mathbf{W}_{\text{spectral}}{}^2_{(i,j)}}{\sigma^2}\right) & \text{if } \mathbf{W}_{\text{spatial}_{(i,j)}} \leq \kappa \\ 0 & \text{if } \mathbf{W}_{\text{spatial}_{(i,j)}} > \kappa. \end{cases} \tag{27}$$

The intuiton behind constructing the spatial-spectral affinity matrix is to calculate spectral similarity between two superpixels if and only if the centroids of the superpixels are within a spatial range κ of each other. This introduces spatial compactness within the partitioning.

After constructing the spatial-spectral affinity matrix, the goal is to to then utilize the normalized cuts algorithm to recursively bipartition the graph $G_{\mathbf{W}}$ represented by $\mathbf{W}$ into n_e subgraphs. Doing so provides a segmentation of the columns of $\mathbf{C}$ representing the superpixels into n_e clusters. Using the spatial-spectral affinity matrix $\mathbf{W}$, the diagonal matrix $\mathbf{D}$ is calculated according (16) and the initial bipartitioning of $G_{\mathbf{W}}$ can be determined by solving for the second smallest eigenvalue λ_2 and the corresponding eigenvector $\mathbf{u}_2$ in the system given in (21). Bipartitioning the graph $G_{\mathbf{W}}$ according the sign of the entries in $\mathbf{u}_2$, the next partition is given by the one that minimizes (19) within the two subgraphs. This process is continued until the graph $G_{\mathbf{W}}$ is partitioned into n_e subgraphs. Cluster membership of the superpixels in $\mathbf{C}$ are assigned to the corresponding subgraph they belong to. Additionally, the mean spectral signatures of the superpixels within each cluster are calculated and arranged into the matrix $\mathbf{M}$.

Algorithm 3: Spatial Spectral Clustering

Input: Superpixel Matrix $\mathbf{C}$, $\kappa > 0$, $\sigma > 0$, $n_e \geq 2$.
Initialize: Construct the spatial-spectral affinity matrix $\mathbf{W}$ and diagonal matrix $\mathbf{D}$ according to (27) and (16).
for $k = 2$ **to** n_e **do**
  For each subgraph, solve the system (21). Bipartition the subgraph according to the cut that minimizes (19).
end
Output: Assign superpixel cluster memberships to a vector $\mathbf{v}_i \in \{1, 2, \ldots, n_e\}$ according to the subgraph each node belongs to. Form a endmember spectra matrix $\mathbf{M} = [\mathbf{m}_1|\mathbf{m}_2, |\ldots|\mathbf{m}_{n_e}]$ where $\mathbf{m}_i$ is the average spectral feature vector for all superpixels within the cluster i.

The algorithm allows for a flexible and efficient framework for segmentation tasks. The initial construction of the affinity matrix according to (27) needs to only be done once, with subsequent subsegmentations being done using selected columns and rows corresponding the the subgraphs each node belongs to. The most obvious bottleneck in the algorithm is the step in which the eigensystem is solved, scalling cubically with the number of superpixels n_s. The lower the number of superpixels, the faster the algorithm performs. On the other end, the higher the number of superpixels, the slower the algorithm performs. The final result is determined by tuning σ and κ. The lower σ is, the more of an emphasis the spectral features have on the final result, while, the lower κ is, the more of an emphasis the spatial information has on the final result. Careful selection of the two parameters allows for meaningful results.

3.4 Graph Regularized Abundance Estimation

In Section 2.3.2, the abundance estimation problem for a collection of pixels $\mathbf{X}$ given the endmember spectra matrix $\mathbf{M}$ was known was stated as follows:

$$\mathbf{A} = \underset{\mathbf{A}\in\mathbb{R}^{n_e\times n_p}}{\arg\min}\ \frac{1}{2}\|\mathbf{MA}-\mathbf{X}\|_F^2 + \chi_\Delta(\mathbf{A}) + J(\mathbf{A}).$$

The goal of this section is to apply a similar framework to the collection of superpixels $\mathbf{C}$ and determine estimates on the fractional abundances given an cluster spectra matrix $\mathbf{M}$ from the output of the clustering in Section 3.3. The abundance estimation problem in terms of superpixels can now be restated as

$$\mathbf{A} = \underset{\mathbf{A}\in\mathbb{R}^{n_e\times n_p}}{\arg\min}\ \frac{1}{2}\|\mathbf{MA}-\mathbf{C}\|_F^2 + \chi_\Delta(\mathbf{A}) + J(\mathbf{A}).$$

In previous sections, a regularization term J was introduced to provide further control on the final values of $\mathbf{A}$. In imaging applications, a common assumption is that color values should typically not vary greatly for pixels next to eachother. In a similar fashion, abundance values should not vary greatly for superpixels spatially close to eachother. To accomodate this assumption, the matrix $\mathbf{W}_{\text{spatial}}$ given in (26) can be exploited by considering

$$\mathbf{W}_{\kappa_{(i,j)}} = \begin{cases} 1 & \text{if } \mathbf{W}_{\text{spatial}_{(i,j)}} \le \kappa \\ 0 & \text{if } \mathbf{W}_{\text{spatial}_{(i,j)}} > \kappa. \end{cases} \tag{28}$$

The regularization term

$$J(\mathbf{A}) = \frac{1}{2}\sum_{i=1}^{n_s}\sum_{j=1}^{n_s}\mathbf{W}_{\kappa_{(i,j)}}\|\mathbf{a}_i-\mathbf{a}_j\|_2^2$$

is minimized under the assumption that $\mathbf{a}_i$ and $\mathbf{a}_j$ should be similar if $\mathbf{c}_i$ and $\mathbf{c}_j$ are spatially within a distance κ of eachother. Importantly, J is convex and differentiable and can alternatively be represented using the corresponding Laplacian matrix $\mathbf{L}$ described in (17) for the matrix $\mathbf{W}_\kappa$:

$$J(\mathbf{A}) = \frac{1}{2}\sum_{i=1}^{n_s}\sum_{j=1}^{n_s}\mathbf{W}_{\kappa_{(i,j)}}\|\mathbf{a}_i-\mathbf{a}_j\|_2^2 = \text{tr}(\mathbf{ALA}^T). \tag{29}$$

The graph regularized abundance estimation problem with the known cluster spectra matrix $\mathbf{M}$ and regularization weight parameter $\beta > 0$ is represented as:

$$\mathbf{A} = \underset{\mathbf{A}\in\mathbb{R}^{n_e\times n_p}}{\arg\min}\ \frac{1}{2}\|\mathbf{MA}-\mathbf{C}\|_F^2 + \chi_\Delta(\mathbf{A}) + \frac{\beta}{2}\text{tr}(\mathbf{ALA}^T). \tag{30}$$

The goal of this rest of the section is to demonstrate how this problem can be equivalently represented in a form where the alternating direction method of multipliers technique can be applied. The abundance estimation problem when one or more than one regularization terms are added belongs to a class of problems called global consensus optimization problems shown in Boyd et al. [REF]. The approach to transforming (30) is to introduce matrices $\mathbf{U}\in\mathbb{R}^{n_e\times n_p}$, $\mathbf{V}_1\in\mathbb{R}^{n_b\times n_p}$, $\mathbf{V}_2\in\mathbb{R}^{n_e\times n_p}$, and $\mathbf{V}_3\in\mathbb{R}^{n_e\times n_p}$ and rewrite the problem as:

$$\begin{aligned} \underset{\mathbf{U},\mathbf{V}_1,\mathbf{V}_2,\mathbf{V}_3}{\text{minimize}} \quad & \frac{1}{2}\|\mathbf{V}_1-\mathbf{C}\|_F^2 + \chi_\Delta(\mathbf{V}_2) + \frac{\beta}{2}\text{tr}(\mathbf{V}_3\mathbf{L}\mathbf{V}_3^T) \\ \text{subject to} \quad & \mathbf{V}_1 = \mathbf{MU} \\ & \mathbf{V}_2 = \mathbf{U} \\ & \mathbf{V}_3 = \mathbf{U} \end{aligned} \tag{31}$$

Further manipulation shows that by letting $g(\mathbf{V}) = \frac{1}{2}\|\mathbf{V}_1-\mathbf{C}\|_F^2 + \chi_\Delta(\mathbf{V}_2) + \frac{\beta}{2}\text{tr}(\mathbf{V}_3\mathbf{L}\mathbf{V}_3^T)$,

$$\mathbf{V} = \begin{bmatrix}\mathbf{V}_1 & & \\ & \mathbf{V}_2 & \\ & & \mathbf{V}_3\end{bmatrix}, \quad \mathbf{G} = \begin{bmatrix}\mathbf{M}\\ \mathbf{I}\\ \mathbf{I}\end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix}-\mathbf{I} & & \\ & -\mathbf{I} & \\ & & -\mathbf{I}\end{bmatrix}$$

The problem in (31) can then be rewritten in an equivalent form as

$$\begin{aligned} \underset{\mathbf{U},\mathbf{V}}{\text{minimize}} \quad & g(\mathbf{V}) \\ \text{subject to} \quad & \mathbf{G}\mathbf{U} + \mathbf{B}\mathbf{V} = \mathbf{0}. \end{aligned} \tag{32}$$

The scaled augmented lagrangian $\mathcal{L}_\mu$ with parameter $\mu > 0$ and scaled dual variable $\mathbf{D}$ is then given as:

$$\mathcal{L}_\mu(\mathbf{U}, \mathbf{V}, \mathbf{D}) = g(\mathbf{V}) + \frac{\mu}{2}\|\mathbf{G}\mathbf{U} + \mathbf{B}\mathbf{V} - \mathbf{D}\|_F^2 \tag{33}$$

where

$$\mathbf{D} = \begin{bmatrix} \mathbf{D}_1 & & \\ & \mathbf{D}_2 & \\ & & \mathbf{D}_3 \end{bmatrix}.$$

ADMM aims to minimize scaled form of $\mathcal{L}_\mu$ by alternating minimizations with respect to $\mathbf{U}$, $\mathbf{V}$, and $\mathbf{D}$ by performing the following updates:

$$\begin{aligned} \mathbf{U}^{(k+1)} &= \arg\min_{\mathbf{U}} \ \frac{\mu}{2}\|\mathbf{G}\mathbf{U} + \mathbf{B}\mathbf{V}^{(k)} - \mathbf{D}^{(k)}\|_F^2 \\ \mathbf{V}^{(k+1)} &= \arg\min_{\mathbf{V}} \ g(\mathbf{V}) + \frac{\mu}{2}\|\mathbf{G}\mathbf{U}^{(k+1)} + \mathbf{B}\mathbf{V} - \mathbf{D}^{(k)}\|_F^2 \\ \mathbf{D}^{(k+1)} &= \mathbf{D}^{(k)} - \mathbf{G}\mathbf{U}^{(k+1)} - \mathbf{B}\mathbf{V}^{(k+1)}. \end{aligned} \tag{34}$$

While the updates are in a simpler format, further work needs to be done to derive updates for $\mathbf{V}$. Looking at the $\|\mathbf{G}\mathbf{U} + \mathbf{B}\mathbf{V}^{(k)} - \mathbf{D}^{(k)}\|_F^2$ term in (33), the structure of it's components give leeway to splitting the term into individual components, notably

$$\begin{aligned} \|\mathbf{G}\mathbf{U} + \mathbf{B}\mathbf{V} - \mathbf{D}\|_F^2 &= \left\| \begin{bmatrix} \mathbf{M}\mathbf{U} - \mathbf{V}_1 - \mathbf{D}_1 & & \\ & \mathbf{U} - \mathbf{V}_2 - \mathbf{D}_2 & \\ & & \mathbf{U} - \mathbf{V}_3 - \mathbf{D}_3 \end{bmatrix} \right\|_F^2 \\ &= \|\mathbf{M}\mathbf{U} - \mathbf{V}_1 - \mathbf{D}_1\|_F^2 + \|\mathbf{U} - \mathbf{V}_2 - \mathbf{D}_2\|_F^2 + \|\mathbf{U} - \mathbf{V}_3 - \mathbf{D}_3\|_F^2. \end{aligned}$$

Applying this expansion, the updates in (34) can be rewritten. The $\mathbf{U}$ update becomes

$$\begin{aligned} \mathbf{U}^{(k+1)} = \arg\min_{\mathbf{U}} \ & \frac{\mu}{2}\|\mathbf{M}\mathbf{U} - \mathbf{V}_1^{(k)} - \mathbf{D}_1^{(k)}\|_F^2 \\ & + \frac{\mu}{2}\|\mathbf{U} - \mathbf{V}_2^{(k)} - \mathbf{D}_2^{(k)}\|_F^2 \\ & + \frac{\mu}{2}\|\mathbf{U} - \mathbf{V}_3^{(k)} - \mathbf{D}_3^{(k)}\|_F^2. \end{aligned} \tag{35}$$

Under the same expansion, the $\mathbf{V}$ update becomes

$$\begin{aligned} \mathbf{V}^{(k+1)} = \arg\min_{\mathbf{V}} \ & \frac{1}{2}\|\mathbf{V}_1 - \mathbf{C}\|_F^2 + \chi_\Delta(\mathbf{V}_2) + \frac{\beta}{2}\mathrm{tr}(\mathbf{V}_3\mathbf{L}\mathbf{V}_3^T) \\ & + \frac{\mu}{2}\|\mathbf{M}\mathbf{U}^{(k+1)} - \mathbf{V}_1 - \mathbf{D}_1^{(k)}\|_F^2 \\ & + \frac{\mu}{2}\|\mathbf{U}^{(k+1)} - \mathbf{V}_2 - \mathbf{D}_2^{(k)}\|_F^2 \\ & + \frac{\mu}{2}\|\mathbf{U}^{(k+1)} - \mathbf{V}_3 - \mathbf{D}_3^{(k)}\|_F^2. \end{aligned}$$

Furthermore, each component of the update for $\mathbf{V}$ can be split into individual updates for $\mathbf{V}_1$, $\mathbf{V}_2$ and $\mathbf{V}_3$,

$$\begin{aligned} \mathbf{V}_1^{(k+1)} &= \arg\min_{\mathbf{V}_1} \ \frac{1}{2}\|\mathbf{V}_1 - \mathbf{C}\|_F^2 + \frac{\mu}{2}\|\mathbf{M}\mathbf{U}^{(k+1)} - \mathbf{V}_1 - \mathbf{D}_1^{(k)}\|_F^2 \\ \mathbf{V}_2^{(k+1)} &= \arg\min_{\mathbf{V}_2} \ \chi_\Delta(\mathbf{V}_2) + \frac{\mu}{2}\|\mathbf{U}^{(k+1)} - \mathbf{V}_2 - \mathbf{D}_2^{(k)}\|_F^2 \\ \mathbf{V}_3^{(k+1)} &= \arg\min_{\mathbf{V}_3} \ \frac{\beta}{2}\mathrm{tr}(\mathbf{V}_3\mathbf{L}\mathbf{V}_3^T) + \frac{\mu}{2}\|\mathbf{U}^{(k+1)} - \mathbf{V}_3 - \mathbf{D}_3^{(k)}\|_F^2 \end{aligned} \tag{36}$$

Lastly, in similar fashion to $\mathbf{V}$, the $\mathbf{D}$ update in (34) can also be split component wise:

$$\begin{aligned}\mathbf{D}_1^{(k+1)} &= \mathbf{D}_1^{(k)} - \mathbf{M}\mathbf{U}^{(k+1)} + \mathbf{V}_1^{(k+1)} \\ \mathbf{D}_2^{(k+1)} &= \mathbf{D}_2^{(k)} - \mathbf{U}^{(k+1)} + \mathbf{V}_2^{(k+1)} \\ \mathbf{D}_3^{(k+1)} &= \mathbf{D}_3^{(k)} - \mathbf{U}^{(k+1)} + \mathbf{V}_3^{(k+1)}.\end{aligned} \tag{37}$$

Taking into account (35), (36), (37), the updates in (34) can finally be rewritten in the expanded form as:

$$\begin{aligned}\mathbf{U}^{(k+1)} &= \arg\min_{\mathbf{U}} \ \frac{\mu}{2}\|\mathbf{M}\mathbf{U} - \mathbf{V}_1^{(k)} - \mathbf{D}_1^{(k)}\|_F^2 + \frac{\mu}{2}\|\mathbf{U} - \mathbf{V}_2^{(k)} - \mathbf{D}_2^{(k)}\|_F^2 + \frac{\mu}{2}\|\mathbf{U} - \mathbf{V}_3^{(k)} - \mathbf{D}_3^{(k)}\|_F^2 \\ \mathbf{V}_1^{(k+1)} &= \arg\min_{\mathbf{V}_1} \ \frac{1}{2}\|\mathbf{V}_1 - \mathbf{C}\|_F^2 + \frac{\mu}{2}\|\mathbf{M}\mathbf{U}^{(k+1)} - \mathbf{V}_1 - \mathbf{D}_1^{(k)}\|_F^2 \\ \mathbf{V}_2^{(k+1)} &= \arg\min_{\mathbf{V}_2} \ \chi_\Delta(\mathbf{V}_2) + \frac{\mu}{2}\|\mathbf{U}^{(k+1)} - \mathbf{V}_2 - \mathbf{D}_2^{(k)}\|_F^2 \\ \mathbf{V}_3^{(k+1)} &= \arg\min_{\mathbf{V}_3} \ \frac{\beta}{2}\mathrm{tr}(\mathbf{V}_3\mathbf{L}\mathbf{V}_3^T) + \frac{\mu}{2}\|\mathbf{U}^{(k+1)} - \mathbf{V}_3 - \mathbf{D}_3^{(k)}\|_F^2 \\ \mathbf{D}_1^{(k+1)} &= \mathbf{D}_1^{(k)} - \mathbf{M}\mathbf{U}^{(k+1)} + \mathbf{V}_1^{(k+1)} \\ \mathbf{D}_2^{(k+1)} &= \mathbf{D}_2^{(k)} - \mathbf{U}^{(k+1)} + \mathbf{V}_2^{(k+1)} \\ \mathbf{D}_3^{(k+1)} &= \mathbf{D}_3^{(k)} - \mathbf{U}^{(k+1)} + \mathbf{V}_3^{(k+1)}.\end{aligned} \tag{38}$$

The updates for $\mathbf{U}$ and $\mathbf{V}_1$ have closed form solutions due to convexity and differentiability of the Frobenius norm [REF]. Both updates can derived by taking the partial derivatives with respect to the individual terms, setting them equal to $\mathbf{0}$, and solving accordingly. For the $\mathbf{U}$ update,

$$\begin{aligned}\mathbf{0} &= \frac{\partial}{\partial \mathbf{U}}\left[\frac{\mu}{2}\|\mathbf{M}\mathbf{U} - \mathbf{V}_1 - \mathbf{D}_1\|_F^2 + \frac{\mu}{2}\|\mathbf{U} - \mathbf{V}_2 - \mathbf{D}_2\|_F^2 + \frac{\mu}{2}\|\mathbf{U} - \mathbf{V}_3 - \mathbf{D}_3\|_F^2\right] \\ \mathbf{0} &= \mu\left(\mathbf{M}^T(\mathbf{M}\mathbf{U} - \mathbf{V}_1 - \mathbf{D}_1) + (\mathbf{U} - \mathbf{V}_2 - \mathbf{D}_2) + (\mathbf{U} - \mathbf{V}_3 - \mathbf{D}_3)\right) \\ \mathbf{M}^T\mathbf{M}\mathbf{U} + 2\mathbf{U} &= \mathbf{M}^T(\mathbf{V}_1 + \mathbf{D}_1) + (\mathbf{V}_2 + \mathbf{D}_2) + (\mathbf{V}_3 + \mathbf{D}_3) \\ \mathbf{U} &= (\mathbf{M}^T\mathbf{M} + 2\mathbf{I})^{-1}(\mathbf{M}^T(\mathbf{V}_1 + \mathbf{D}_1) + (\mathbf{V}_2 + \mathbf{D}_2) + (\mathbf{V}_3 + \mathbf{D}_3)).\end{aligned}$$

As $\mathbf{M}$ is known and unchanged, $(\mathbf{M}^T\mathbf{M} + 2\mathbf{I})^{-1}$ can be calculated and cached once for the entire runtime. For the $\mathbf{V}_1$ update,

$$\begin{aligned}\mathbf{0} &= \frac{\partial}{\partial \mathbf{V}_1}\left[\frac{1}{2}\|\mathbf{V}_1 - \mathbf{C}\|_F^2 + \frac{\mu}{2}\|\mathbf{M}\mathbf{U} - \mathbf{V}_1 - \mathbf{D}_1\|_F^2\right] \\ \mathbf{0} &= (\mathbf{V}_1 - \mathbf{C}) + \mu(\mathbf{V}_1 - (\mathbf{M}\mathbf{U} - \mathbf{D}_1)) \\ \mathbf{V}_1 &= \frac{1}{1+\mu}\left(\mathbf{C} + (\mathbf{M}\mathbf{U} - \mathbf{D}_1)\right).\end{aligned}$$

While the update for $\mathbf{V}_2$ does not have a closed form solution, it is important to note that the update $\mathbf{V}_2$ can be equivalently rewritten as:

$$\mathbf{V}_2^{(k+1)} = \arg\min_{\mathbf{V}_2 \in \Delta} \ \frac{\mu}{2}\|\mathbf{U}^{(k+1)} - \mathbf{V}_2 - \mathbf{D}_2^{(k)}\|_F^2.$$

The update, in non-formulaic terms, requires finding $\mathbf{V}_2$ that minimizes $\frac{\mu}{2}\|\mathbf{U}^{(k+1)} - \mathbf{V}_2 - \mathbf{D}_2^{(k)}\|_F^2$, then projecting the solution onto Δ. The non-projected minimum can be found in the same way as the updates for $\mathbf{V}$ and $\mathbf{U}$

$$\begin{aligned}\mathbf{0} &= \frac{\partial}{\partial \mathbf{V}_2}\left[\frac{\mu}{2}\|\mathbf{U} - \mathbf{V}_2 - \mathbf{D}_2\|_F^2\right] \\ \mathbf{0} &= \mu(\mathbf{V}_2 - (\mathbf{M}\mathbf{U} - \mathbf{D}_2)) \\ \mathbf{V}_2 &= \mathbf{M}\mathbf{U} - \mathbf{D}_2.\end{aligned}$$

The orthogonal projection of a matrix $\mathbf{C}$ onto the convex and closed set Δ is defined as the finding the matrix $\tilde{\mathbf{C}} \in \Delta$ that minimizes the least-squares error between the two matrices. Multiple numerical methods exist for computing the projection [REF]. Formally,

$$\mathrm{proj}_\Delta(\mathbf{C}) = \arg\min_{\tilde{\mathbf{C}} \in \Delta} \ \|\tilde{\mathbf{C}} - \mathbf{C}\|_F^2.$$

Thus, applying the projection, the update for $\mathbf{V}_2$ is given as

$$\mathbf{V}_2 = \text{proj}_\Delta(\mathbf{MU} - \mathbf{D}_2).$$

The approach for deriving update for $\mathbf{V}_3$ follows the same as $\mathbf{U}$ and $\mathbf{V}_1$ due to the convexity and differentiability of the regularization term $\text{tr}(\mathbf{V}_3\mathbf{L}\mathbf{V}_3^T)$.

$$\begin{aligned}
\mathbf{0} &= \frac{\partial}{\partial \mathbf{V}_3}\left[\frac{\beta}{2}\text{tr}(\mathbf{V}_3\mathbf{L}\mathbf{V}_3^T) + \frac{\mu}{2}\|\mathbf{U} - \mathbf{V}_3 - \mathbf{D}_3\|_F^2\right] \\
\mathbf{0} &= \frac{\beta}{2}(\mathbf{V}_3\mathbf{L}^T + \mathbf{V}_3\mathbf{L}) + \mu(\mathbf{V}_3 - (\mathbf{U} - \mathbf{D}_3)) \\
\mathbf{0} &= \beta\mathbf{V}_3\mathbf{L} + \mu\mathbf{V}_3 - \mu(\mathbf{U} - \mathbf{D}_3) \\
\mathbf{V}_3\left(\mathbf{L} + \frac{\mu}{\beta}\mathbf{I}\right) &= \frac{\mu}{\beta}(\mathbf{U} - \mathbf{D}_3) \\
\mathbf{V}_3 &= \frac{\mu}{\beta}(\mathbf{U} - \mathbf{D}_3)\left(\mathbf{L} + \frac{\mu}{\beta}\mathbf{I}\right)^{-1}.
\end{aligned}$$

As $\mathbf{L}$ is a real valued, symettric matrix, it can be eigendecomposed into the product $\mathbf{L} = \mathbf{S}\mathbf{\Sigma}\mathbf{S}^T$, where $\mathbf{S}$ is a matrix whose columns are the eigenvectors of $\mathbf{L}$, and $\mathbf{\Sigma}$ is a matrix whose diagonal elements are the eigenvalues of $\mathbf{L}$. Additionally, $\mathbf{S}$ is an orthogonal matrix, as such, $\mathbf{S}^T = \mathbf{S}^{-1}$ and $\mathbf{S}\mathbf{S}^T = \mathbf{I}$. [REF] One important note to be made about the update step is that computing the inverse of the term $(\mathbf{L} + \mu/\beta\mathbf{I})$ is slower than computing the eigendecomposition $\mathbf{L}$ due to the naturally sparse definition of $\mathbf{L}$ and it's underlying distance matrix $\mathbf{W}_\kappa$. Using that information, a more efficent update can be performed by calculating the eigendecomposition $\mathbf{L} = \mathbf{S}\mathbf{\Sigma}\mathbf{S}^T$ and simplifying the update for $\mathbf{V}_3$ as follows:

$$\begin{aligned}
\mathbf{V}_3 &= \frac{\mu}{\beta}(\mathbf{U} - \mathbf{D}_3)\left(\mathbf{L} + \frac{\mu}{\beta}\mathbf{I}\right)^{-1} \\
\mathbf{V}_3 &= \frac{\mu}{\beta}(\mathbf{U} - \mathbf{D}_3)\left(\mathbf{S}\mathbf{\Sigma}\mathbf{S}^T + \frac{\mu}{\beta}\mathbf{S}\mathbf{S}^T\right)^{-1} \\
\mathbf{V}_3 &= \frac{\mu}{\beta}(\mathbf{U} - \mathbf{D}_3)\left(\mathbf{S}\left(\mathbf{\Sigma} + \frac{\mu}{\beta}\mathbf{I}\right)\mathbf{S}^T\right)^{-1} \\
\mathbf{V}_3 &= \frac{\mu}{\beta}(\mathbf{U} - \mathbf{D}_3)\mathbf{S}\left(\mathbf{\Sigma} + \frac{\mu}{\beta}\mathbf{I}\right)^{-1}\mathbf{S}^T.
\end{aligned}$$

The values along the diagonal in $\mathbf{\Sigma}$ are all nonnegative due to $\mathbf{L}$ having nonnegative eigenvalues [REF]. As such, the matrix $(\mathbf{\Sigma} + \mu/\beta\mathbf{I})$ is a diagonal matrix with the it's elements being all strictly positive. The inverse of $(\mathbf{\Sigma} + \mu/\beta\mathbf{I})$ can be directly calculated as by taking the reciprocal of it's diagonal elements. As $\mathbf{L}$ is known at the onset of the algorithm and the parameters μ and β do not change between iterations, $\mathbf{S}(\mathbf{\Sigma}+\mu/\beta\mathbf{I})^{-1}\mathbf{S}^T$ can be cached and reused across iterations.

The derived updates in (38), for parameters $\mu > 0$ and $\beta > 0$ are subsequently given as:

$$\mathbf{U}^{(k+1)} = (\mathbf{M}^T\mathbf{M} + 2\mathbf{I})^{-1}\left(\mathbf{M}^T\left(\mathbf{V}_1^{(k)} + \mathbf{D}_1^{(k)}\right) + \left(\mathbf{V}_2^{(k)} + \mathbf{D}_2^{(k)}\right) + \left(\mathbf{V}_3^{(k)} + \mathbf{D}_3^{(k)}\right)\right) \tag{39a}$$

$$\mathbf{V}_1^{(k+1)} = \frac{1}{1+\mu}\left(\mathbf{C} + \left(\mathbf{M}\mathbf{U}^{(k+1)} - \mathbf{D}_1^{(k)}\right)\right) \tag{39b}$$

$$\mathbf{V}_2^{(k+1)} = \text{proj}_\Delta\left(\mathbf{M}\mathbf{U}^{(k+1)} - \mathbf{D}_2^{(k)}\right) \tag{39c}$$

$$\mathbf{V}_3^{(k+1)} = \frac{\mu}{\beta}\left(\mathbf{U}^{(k+1)} - \mathbf{D}_3^{(k)}\right)\mathbf{S}\left(\mathbf{\Sigma} + \frac{\mu}{\beta}\mathbf{I}\right)^{-1}\mathbf{S}^T \tag{39d}$$

$$\mathbf{D}_1^{(k+1)} = \mathbf{D}_1^{(k)} - \mathbf{M}\mathbf{U}^{(k+1)} + \mathbf{V}_1^{(k+1)} \tag{39e}$$

$$\mathbf{D}_2^{(k+1)} = \mathbf{D}_2^{(k)} - \mathbf{U}^{(k+1)} + \mathbf{V}_2^{(k+1)} \tag{39f}$$

$$\mathbf{D}_3^{(k+1)} = \mathbf{D}_3^{(k)} - \mathbf{U}^{(k+1)} + \mathbf{V}_3^{(k+1)}. \tag{39g}$$

The algorithm is set to terminate when $\|\mathbf{U}^{(k+1)} - \mathbf{U}^{(k)}\|_F / \|\mathbf{U}^{(k)}\|_F$ falls below a set tolerance ϵ or the algorithm reaches a maximum iterative index of $k_{\max}$. After such point, $\mathbf{U}^{(k+1)}$ is given as the final result, representing the

approximate solution of $\mathbf{A}$ to the minimization problem described in (30). In it's entirety, the abundance estimation algorithm can be described with the following algorithm outline.

Algorithm 4: Graph Regularized Abundance Estimation

Input:
$\mathbf{C}$, $\mathbf{M}$, $\mathbf{W}_\kappa$, $\beta > 0$, $\mu > 0$, $k_{\max} > 0$, $\epsilon > 0$.
Initialize:
Precompute and cache $\mathbf{S}(\mathbf{\Sigma} + \mu/\beta\mathbf{I})^{-1}\mathbf{S}^T$ and $(\mathbf{M}^T\mathbf{M} + 2\mathbf{I})^{-1}$
$\mathbf{U}^{(0)} \in \Delta$
$\mathbf{V}_1^{(0)} = \mathbf{M}\mathbf{U}^{(0)}$
$\mathbf{V}_2^{(0)} = \mathbf{U}^{(0)}$
$\mathbf{V}_3^{(0)} = \mathbf{U}^{(0)}$
$\mathbf{D}_1^{(0)} = \mathbf{0}$
$\mathbf{D}_2^{(0)} = \mathbf{0}$
$\mathbf{D}_3^{(0)} = \mathbf{0}$
For $k = 0$ to $k_{\max}$:
Update $\mathbf{U}^{(k+1)}$ according to (39a)
Update $\mathbf{V}_1^{(k+1)}$ according to (39b)
Update $\mathbf{V}_2^{(k+1)}$ according to (39c)
Update $\mathbf{V}_3^{(k+1)}$ according to (39d)
Update $\mathbf{D}_1^{(k+1)}$ according to (39e)
Update $\mathbf{D}_2^{(k+1)}$ according to (39f)
Update $\mathbf{D}_3^{(k+1)}$ according to (39g)
Break if $\|\mathbf{U}^{(k+1)} - \mathbf{U}^{(k)}\|_F / \|\mathbf{U}^{(k)}\|_F < \epsilon$
Output:
Abundance Matrix $\mathbf{A} = \mathbf{U}$

The algorithm described above gives a efficient solution to the abundance estimation problem in a relatively low number of iterations. Additional inquiry shows that the updates for $\mathbf{V}$ and $\mathbf{D}$ can be done completely in parallel, allowing for further performance optimization. In practical applications, β is the only parameter to be tuned, corresponding to the strength of the spatial regularization, κ is predefined in Section 3.3, as such it is not the focus of tuning in this step.

3.5 Feature Vector Creation

3.6 Compilation

3.7 Parameter Selection

4 Experimental Results

4.1 Implementation Details

4.2 Geospatial Testing Datasets

4.2.1 Samson Datasets

4.2.2 Salinas Datasets

4.2.3 Biomedical Autofluorescence Dataset

4.3 Algorithm Evaluation

4.3.1 Qualitative Evaluation on Samson

4.3.2 Quantitative Evaluation on Salinas

4.3.3 Qualitative Evaluation on Biomedical Autofluorescence Data

4.4 Algorithm Comparison

5 Conclusions