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

2 Background

2.1 Hyperspectral Imaging: Basics

In hyperspectral imaging, near contiguous narrow band spectral information is measured for each spatial pixel of an image collected over a scene. Spectral information can be quantified by multiple measures. Traditionally, spectral radiance, being the energy emitted or reflected by a surface over a large number of spectral wavelengths, has been the measure of choice for physical applications. Utilizing this spectral information, physical and chemical properies of materials can be deduced and insights can be made about the overall composition of the images, resulting in applications across various fields.

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 that are common in hyperspectral image analysis.

2.1.1 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.

The tensor $\mathbf{X}$ can be transformed into a two-dimensional matrix of shape (n_p, n_λ), denoted $\mathbf{X}_f$, where $n_p = n_x n_y$ is the total number of pixels in the image, with each column representing the pixel at index (i, j) in $\mathbf{X}$, arranged across the first spectral dimension, then the second. Formally,

$$\mathbf{X}_f = \left[\mathbf{x}_{(0,0)}, \cdots, \mathbf{x}_{(n_x,0)}, \cdots, \mathbf{x}_{(0,n_y)}, \cdots, \mathbf{x}_{(n_x,n_y)}\right].$$

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, we restructure each pixel $\mathbf{x}_{(i,j)} = [\mathbf{x}_l, \mathbf{x}_a, \mathbf{x}_b]$ into the form $\tilde{\mathbf{x}}_{(i,j)} = [\mathbf{x}_l, \mathbf{x}_a, \mathbf{x}_b, i, j]$. 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 $\mathbf{C}_n = [\mathbf{c}_l, \mathbf{c}_a, \mathbf{c}_b, i, j]$ 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: $\mathbf{X}_f$, $m > 0$, $n_s > 0$, $n_{\text{iters}} > 0$
Initialize: $\mathbf{C}_n = [\mathbf{c}_l, \mathbf{c}_a, \mathbf{c}_b, i, j]$ 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 of all pixels belonging to cluster.
end
Optional: Relabel disjoint segments.

The SLIC algorithm is shown to produce meaningful and noise-robust segments in traditional computer vision applications. This algorithm proves useful in Section 3.2.1 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 $\mathbf{M} = [\mathbf{m}_1|\mathbf{m}_2|\cdots|\mathbf{m}_{n_e}] \in \mathbb{R}_+^{n_b \times n_e}$ 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 $\mathbf{X} = [\mathbf{x}_1|\mathbf{x}_2|\cdots|\mathbf{x}_{n_p}] \in \mathbb{R}_+^{n_b \times n_p}$, with each pixel $\mathbf{x}_i$ having a corresponding abundance vector $\mathbf{a}_i$. Arranging the abundance vectors into an abudance matrix $\mathbf{A} = [\mathbf{a}_1|\mathbf{a}_2|\cdots|\mathbf{a}_{n_p}] \in \mathbb{R}^{n_e \times n_p}$, 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 where

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

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].

2.3.3 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.

2.3.4 Abundance Estimation using ADMM

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:

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

The goal of this 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]

Operating under the assumption that J is a convex function, since for nonconvex choices of J, convergence is not guarenteed. The approach to transforming (8) is to first 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 as follows:

$$
\begin{aligned}
\underset{\mathbf{U},\mathbf{V}_1,\mathbf{V}_2,\mathbf{V}_3}{\text{minimize}} \quad & \frac{1}{2}\|\mathbf{V}_1 - X\|_F^2 + \chi_\Delta(\mathbf{V}_2) + J(\mathbf{V}_3) \\
\text{subject to} \quad & \mathbf{V}_1 = \mathbf{MU} \\
& \mathbf{V}_2 = \mathbf{U} \\
& \mathbf{V}_3 = \mathbf{U}
\end{aligned} \tag{13}
$$

where

$$
g(\mathbf{V}) = \frac{1}{2}\|\mathbf{V}_1 - X\|_F^2 + \chi_\Delta(\mathbf{V}_2) + J(\mathbf{V}_3)
$$

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

The problem depicted in (13) can then be rewritten in equivalent form:

$$
\begin{aligned}
\underset{\mathbf{U},\mathbf{V}}{\text{minimize}} \quad & g(\mathbf{V}) \\
\text{subject to} \quad & \mathbf{GU} + \mathbf{BV} = \mathbf{0}
\end{aligned} \tag{14}
$$

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

$$
\mathcal{L}_\mu(\mathbf{U}, \mathbf{V}, \mathbf{D}) = g(\mathbf{V}) + \frac{\mu}{2}\|\mathbf{GU} + \mathbf{BV} - \mathbf{D}\|_F^2 \tag{15}
$$

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:

$$\begin{aligned}
\mathbf{U}^{(k+1)} &= \arg\min_{\mathbf{U}} \ \frac{\mu}{2}\|\mathbf{G}\mathbf{U} + \mathbf{B}\mathbf{V}^{(k)} - \mathbf{D}^{(k)}\|_F^2 \\
\mathbf{V}^{(k+1)} &= \arg\min_{\mathbf{V}} \ g(\mathbf{V}) + \frac{\mu}{2}\|\mathbf{G}\mathbf{U}^{(k+1)} + \mathbf{B}\mathbf{V} - \mathbf{D}^{(k)}\|_F^2 \\
\mathbf{D}^{(k+1)} &= \mathbf{D}^{(k)} - \mathbf{G}\mathbf{U}^{(k+1)} - \mathbf{B}\mathbf{V}^{(k+1)}.
\end{aligned} \tag{16}$$

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 (15), 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 (16) can be rewritten. The $\mathbf{U}$ update becomes

$$\begin{aligned}
\mathbf{U}^{(k+1)} = \arg\min_{\mathbf{U}} \ & \frac{\mu}{2}\|\mathbf{M}\mathbf{U} - \mathbf{V}_1^{(k)} - \mathbf{D}_1^{(k)}\|_F^2 \\
& + \frac{\mu}{2}\|\mathbf{U} - \mathbf{V}_2^{(k)} - \mathbf{D}_2^{(k)}\|_F^2 \\
& + \frac{\mu}{2}\|\mathbf{U} - \mathbf{V}_3^{(k)} - \mathbf{D}_3^{(k)}\|_F^2.
\end{aligned} \tag{17}$$

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

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

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

$$\begin{aligned}
\mathbf{V}_1^{(k+1)} &= \arg\min_{\mathbf{V}_1} \ \frac{1}{2}\|\mathbf{V}_1 - X\|_F^2 + \frac{\mu}{2}\|\mathbf{M}\mathbf{U}^{(k+1)} - \mathbf{V}_1 - \mathbf{D}_1^{(k)}\|_F^2 \\
\mathbf{V}_2^{(k+1)} &= \arg\min_{\mathbf{V}_2} \ \chi_\Delta(\mathbf{V}_2) + \frac{\mu}{2}\|\mathbf{U}^{(k+1)} - \mathbf{V}_2 - \mathbf{D}_2^{(k)}\|_F^2 \\
\mathbf{V}_3^{(k+1)} &= \arg\min_{\mathbf{V}_3} \ J(\mathbf{V}_3) + \frac{\mu}{2}\|\mathbf{U}^{(k+1)} - \mathbf{V}_3 - \mathbf{D}_3^{(k)}\|_F^2
\end{aligned} \tag{18}$$

Lastly, in similar fashion to $\mathbf{V}$, the $\mathbf{D}$ update in (16) can also be split component wise:

$$\begin{aligned}
\mathbf{D}_1^{(k+1)} &= \mathbf{D}_1^{(k)} - \mathbf{M}\mathbf{U}^{(k+1)} + \mathbf{V}_1^{(k+1)} \\
\mathbf{D}_2^{(k+1)} &= \mathbf{D}_2^{(k)} - \mathbf{U}^{(k+1)} + \mathbf{V}_2^{(k+1)} \\
\mathbf{D}_3^{(k+1)} &= \mathbf{D}_3^{(k)} - \mathbf{U}^{(k+1)} + \mathbf{V}_3^{(k+1)}.
\end{aligned} \tag{19}$$

Taking into account (17), (18), (19), the updates in (16) can finally be rewritten in the expanded form as:

$$\begin{aligned}
\mathbf{U}^{(k+1)} &= \arg\min_{\mathbf{U}} \ \frac{\mu}{2}\|\mathbf{MU} - \mathbf{V}_1^{(k)} - \mathbf{D}_1^{(k)}\|_F^2 + \frac{\mu}{2}\|\mathbf{U} - \mathbf{V}_2^{(k)} - \mathbf{D}_2^{(k)}\|_F^2 + \frac{\mu}{2}\|\mathbf{U} - \mathbf{V}_3^{(k)} - \mathbf{D}_3^{(k)}\|_F^2 \\
\mathbf{V}_1^{(k+1)} &= \arg\min_{\mathbf{V}_1} \ \frac{1}{2}\|\mathbf{V}_1 - X\|_F^2 + \frac{\mu}{2}\|\mathbf{MU}^{(k+1)} - \mathbf{V}_1 - \mathbf{D}_1^{(k)}\|_F^2 \\
\mathbf{V}_2^{(k+1)} &= \arg\min_{\mathbf{V}_2} \ \chi_\Delta(\mathbf{V}_2) + \frac{\mu}{2}\|\mathbf{U}^{(k+1)} - \mathbf{V}_2 - \mathbf{D}_2^{(k)}\|_F^2 \\
\mathbf{V}_3^{(k+1)} &= \arg\min_{\mathbf{V}_3} \ J(\mathbf{V}_3) + \frac{\mu}{2}\|\mathbf{U}^{(k+1)} - \mathbf{V}_3 - \mathbf{D}_3^{(k)}\|_F^2 \\
\mathbf{D}_1^{(k+1)} &= \mathbf{D}_1^{(k)} - \mathbf{MU}^{(k+1)} + \mathbf{V}_1^{(k+1)} \\
\mathbf{D}_2^{(k+1)} &= \mathbf{D}_2^{(k)} - \mathbf{U}^{(k+1)} + \mathbf{V}_2^{(k+1)} \\
\mathbf{D}_3^{(k+1)} &= \mathbf{D}_3^{(k)} - \mathbf{U}^{(k+1)} + \mathbf{V}_3^{(k+1)}.
\end{aligned} \tag{20}$$

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 first partial derivatives with respect to the individual terms, setting it equal to $\mathbf{0}$, and solving accordingly. For the $\mathbf{U}$ update,

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

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

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

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 $\mathbf{U}$

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

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

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

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

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

After deriving solutions for $\mathbf{V}_1$, $\mathbf{V}_2$, $\mathbf{U}$, the updates in (16) can be written as:

$$
\begin{aligned}
\mathbf{U}^{(k+1)} &= (\mathbf{M}^T\mathbf{M} + 2\mathbf{I})^{-1}(\mathbf{M}^T(\mathbf{V}_1^{(k)} + \mathbf{D}_1^{(k)}) + (\mathbf{V}_2^{(k)} + \mathbf{D}_2^{(k)}) + (\mathbf{V}_3^{(k)} + \mathbf{D}_3^{(k)})).\\
\mathbf{V}_1^{(k+1)} &= \frac{1}{1+\mu}\left(\mathbf{X} + (\mathbf{M}\mathbf{U}^{(k+1)} - \mathbf{D}_1^{(k)})\right)\\
\mathbf{V}_2^{(k+1)} &= \text{proj}_{\Delta}(\mathbf{M}\mathbf{U}^{(k+1)} - \mathbf{D}_2^{(k)})\\
\mathbf{V}_3^{(k+1)} &= \arg\min_{\mathbf{V}_3} \quad J(\mathbf{V}_3) + \frac{\mu}{2}\|\mathbf{U}^{(k+1)} - \mathbf{V}_3 - \mathbf{D}_3^{(k)}\|_F^2\\
\mathbf{D}_1^{(k+1)} &= \mathbf{D}_1^{(k)} - \mathbf{M}\mathbf{U}^{(k+1)} + \mathbf{V}_1^{(k+1)}\\
\mathbf{D}_2^{(k+1)} &= \mathbf{D}_2^{(k)} - \mathbf{U}^{(k+1)} + \mathbf{V}_2^{(k+1)}\\
\mathbf{D}_3^{(k+1)} &= \mathbf{D}_3^{(k)} - \mathbf{U}^{(k+1)} + \mathbf{V}_3^{(k+1)}.
\end{aligned}
\tag{21}
$$

Leaving the update for $\mathbf{V_3}$ as the only minimization problem for practicioners to derive the update for. Many variations of the abundance estimation problem exist where one or more regularization terms on $\mathbf{A}$ are added for further control over the final solution. This technique can be further extended to consider m regularization terms $J_1, \ldots, J_m$, in which the result is $m+2$ additional updates for the subproblems arising from splitting $\mathbf{V}$ and $m+2$ additional updates for the subproblems arising from splitting $\mathbf{D}$. The alternating direction method of multipliers technique is most useful in situations where multiple regularization terms exist in the loss function and no closed form solution exist, demonstrating it's superiority in the field of hyperspectral image analysis.

2.4 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

$$\mathbf{W}_{(i,j)} = d(\mathbf{x}_i, \mathbf{x}_j). \tag{22}$$

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.

$$\begin{aligned} d_{L_2}(\mathbf{x}_i, \mathbf{x}_j) &= \|\mathbf{x}_i - \mathbf{x}_j\|_2 \\ d_\theta(\mathbf{x}_i, \mathbf{x}_j) &= \arccos\left(\frac{\mathbf{x}_i \mathbf{x}_j^T}{\|\mathbf{x}_i\|_2 \|\mathbf{x}_j\|_2}\right) \end{aligned} \tag{23}$$

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 $\mathbf{W}(i,j) = 1$.

$$\mathbf{W}_{(i,j)} = \exp\left(-\frac{d(\mathbf{x}_i, \mathbf{x}_j)^2}{\sigma^2}\right). \tag{24}$$

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 $G_{\mathbf{W}}$. Spectral Analysis itself is the study of $\mathbf{W}$ using linear algebra techniques to determine insights on the structure of $G_{\mathbf{W}}$ using the eigenvalues and eigenvectors of $\mathbf{W}$. A fundamental matrix in spectral analysis is the graph Laplacian matrix $\mathbf{L}$. $\mathbf{L}$ is calculated as the difference between the diagonal matrix $\mathbf{D}$, calculated as

$$\mathbf{D}_{(i,j)} = \begin{cases} \sum_j \mathbf{W}_{(i,j)} & \text{if } i = j, \\ 0 & \text{if } i \neq j \end{cases} \tag{25}$$

and $\mathbf{W}$. Formally,

$$\mathbf{L} = \mathbf{D} - \mathbf{W} \tag{26}$$

2.4.1 Normalized Cuts

3 Adaptive Superpixel Cuts

3.1 Dataset Preprocessing

3.1.1 Singular Value Decomposition

3.1.2 Layer Normalization

3.2 Algorithm Overview

3.2.1 Superpixel Generation

3.2.2 Construction of the Affinity Matrix W

3.2.3 Normalized Cuts Algorithm

3.2.4 Graph Regularized Fully Constrained Unmixing

3.2.5 Feature Vector Creation

3.2.6 Parameter Selection

4 Experimental Results

4.1 Implementation Details

4.2 Geospatial Testing Datasets

4.2.1 Samson Datasets

4.2.2 Salinas Datasets

4.3 Algorithm Evaluation

4.3.1 Qualitative Evaluation on Samson

4.3.2 Quantitative Evaluation on Salinas

4.4 Algorithm Comparison

5 Applications in Biomedical Imaging

5.1 Dr Yucel Motivations

5.2 Results on Biomedical Hyperspectral Images

6 Conclusions