Nonlocal Total Variation with Primal Dual Algorithm and Stable Simplex Clustering in Unsupervised Hyperspectral Imagery Analysis

Wei Zhu∗ Victoria Chayes† Alexandre Tiard† Stephanie Sanchez∗
Devin Dahlberg§ Da Kuang∗ Andrea Bertozzi∗ Stanley Osher∗
Dominique Zosso∗

Abstract

We focus on implementing a nonlocal total variational method for unsupervised classification of hyperspectral imagery. We minimize the energy directly using a primal dual algorithm, which we modified for the non-local gradient and weighted centroid recalculation. By squaring the labeling function in the fidelity term before re-calculting the cluster centroids, we can implement an unsupervised clustering method with random initialization. We stabilize this method with stable simplex clustering. To better differentiate clusters, we use a linear combination of the cosine and Euclidean distance between spectral signatures instead of the traditional cosine distance. Finally, we speed up the calculation using a k-d tree and approximate nearest neighbor search algorithm for calculation of the weight matrix for distances between pixel signatures. We implement our method on six different datasets and compare results to traditional clustering methods like k-means, non-negative matrix factorization, and the graph-based MBO scheme.

Keywords: nonlocal total variation, primal dual algorithm, hyperspectral imagery, unsupervised clustering method, stable simplex clustering, approximate nearest neighbor.

1 Introduction

Hyperspectral imagery is a burgeoning domain in the field of remote sensing with numerous applications in agriculture, environmental science, mineralogy, medical imaging, and surveillance. Hyperspectral sensors capture information of intensity of reflection at different wavelengths, from the infrared to ultraviolet. They typically take measurements approximately 10-30nm bandwidths apart, and as many as 200 sample layers for a single image. Thus each pixel has a unique spectral signature. These spectral signatures can be used to differentiate objects that cannot be distinguished on sight, for example: invisible gas plumes, oil or chemical spills over water, or healthy from unhealthy crops. However, the storage and processing of this information can be costly because of the sheer amount of data. Ideally, one would wish to run some sort of algorithm that

∗Department of Mathematics, University of California, Los Angeles
†Bard College
‡ENSE3, Grenoble Institute of Technology
§University of California, San Diego
segments an image and detects anomalies automatically, then returns the segmented image for further human classification. Techniques for classifying hyperspectral imagery optimally maximize speed, accuracy, and robustness. We compare our new technique, unsupervised non-local total variation (NLTV) with the existing methods of k-means, non-negative matrix factorization (NMF) and hierarchical rank-2 non-negative matrix factorization (H2NMF), and the graph-based modified Merriman-Bence-Osher (MBO) scheme.

There are two classes of hyperspectral segmentation: unmixing methods and clustering methods. Unmixing methods attempt to segment an image, as well as extract the spectral signature of the endmembers, or the pure spectral signature of the material present. Clustering methods do not extract endmembers; while they can return the spectral signatures of the centroids of the clusters, these centroids are merely the mean of the signatures of all the pixels of the cluster, and do not necessarily represent the true signature of the original material. The non-local total variation method that we developed is a clustering method.

We compared our algorithm to other popular segmenting techniques in both the unmixing and the clustering categories. The existing methods to which we compare our technique all have various drawbacks, which was our inspiration to develop our new clustering technique. K-means is a traditional algorithm that is popular in a myriad of data mining applications, that is a method of vector quantization. However, k-means is sensitive to noise because it uses square means [8], uses hard clustering (i.e. data points that might be considered “borderline” are given the same weight in re-calculating the centroid in subsequent iterations) [9], and can get trapped in local minima, all of which can result in inaccurate clustering of the data. NMF and H2NMF are both linear unmixing models. They make two key assumptions about the data: firstly, that the signal for each pixel is composed of a linear combination of the endmembers of the clusters, and secondly, the “pure pixel” assumption that at least one pixel exists that is composed purely of each endmember. These two assumptions have been shown to be physically inaccurate [4]. Bilinear unmixing models, which add a term to account for reflectance between objects of different endmembers in a pixel still do not account for intimate mixtures, and physics-based techniques which use a molecular model for intimate mixtures such as radiative transfer theory are an ill-posed problem because of sparse scene parameters [6].

In 2012, Bertozzi and Flenner proposed a graph-based clustering model with applications of classifications of large datasets, including hyperspectral imagery, which Merkurjev et al. expanded to apply specifically to hyperspectral imagery [1] [3]. This method minimizes a graph version of the Ginzburg Landau energy functional because it gamma-converges to the total variation semi-norm, then uses Nyström extension to speed up calculating the eigenvectors. This method performed excellently against other algorithms in the detection of chemical plumes in hyperspectral data, which is a difficult problem because of the diffuse nature of the gas [16] [15]. However, the functional that this model minimizes has a double-well term that makes the problem non-convex; this means that the algorithm can get stuck in local minima. This issue can be circumvented by running the algorithm 10 times with different initial conditions, then hand-picking the best result, making this a semi-supervised method.

Our method is a data-based clustering technique which took inspiration from Bertozzi and Flenner’s model; however, instead of using the Ginzberg-Landau functional, we minimize the total-variational semi-norm directly. This allows the problem to remain convex. Furthermore, it is more accurate, because the Ginzberg-Landau functional is used to approximate the total variation semi-norm. Our segmenting scheme avoids the need to calculate the graph Laplacian or any eigenvectors. To optimize this scheme for hyperspectral imagery, we use the non-local gradient, and we introduce
the novel idea of squaring the labeling function in the fidelity term, which ensures that anomalies converge to their own clusters and allows for random pixel initialization of our algorithm. Our direct usage of the total variation semi-norm makes our clustering method extremely accurate, and our quadratic model for random initialization with stable simplex clustering is a completely new addition to the field of hyperspectral image classification.

This paper is organized in the following manner: in Section 2 background is provided on total variation, nonlocal operators, and the primal dual method. In Section 3 we introduce our research: first we apply primal dual to the non-local linear total variation model, then we apply primal dual to our new innovation of the quadratic non-local total variation model. We also introduce a stable simplex clustering scheme, which allows the non-local total variation image segmentation to converge in fewer iterations. Section 4 outlines a computer science technique for speeding up the calculation of the weight matrix, namely, the formation of a k-d tree then the application of an approximate nearest neighbor search, which vastly reduces the number of pixels that need to be compared. Section 5 presents our results. We run our algorithm on the Urban, San Diego Airport, Pavia University, Pavia Centre, Deepwater Horizon Oil Spill, and Chemical Plume datasets, Section 6 presents our conclusions, and Section 7 states directions for future work.

2 Total Variation and Primal Dual

2.1 Total Variation and Non-Local Operators

Total variation method is an image processing technique introduced by Rudin et al in 1992 [7]. The concept is simple: the presence of noise in an image is directly reflected by the gradient of the intensity at each pixel $u(x)$. By minimizing norm of the gradient of $u(x)$, one can remove noise from the image and preserve edges in the image at the same time. The fidelity term $F(u)$ then penalizes differences between the new image and the original image. The total variation model for image denoising is:

$$E(u) = \| \nabla u \|_{L^1} + \frac{\lambda}{2} F(u)$$

(1)

The parameter $\lambda$ can be adjusted to give higher priority to the de-noising, or the similarity to the original image.

The total variation method was traditionally used for image denoising; nonetheless, it has applications in image reconstruction, inpainting correction and image segmentation [25]. However, this is still a local method. More specifically, when the gradient of a pixel is being calculated, this is done using at most four directly adjacent pixels. As a result, local image processing techniques fail to produce satisfactory results when the image has repetitive structures. In their work [22][23], Zhou and schölkopf constructed a theory of nonlocal operators. Gilboa and Osher [19] later formalized a systematic framework for nonlocal image processing. Nonlocal image processing produces much better results because theoretically any pixel in the image can interact with any other, which better preserves texture and fine detail.

In hyperspectral image analysis, or in any image segmenting problem, clusters can have elements that are not spatially connected. Thus to use total variation as a segmenting technique, it was necessary to develop a non-local method of calculating the gradient at each pixel. We provide a review of non-local operators for readers unfamiliar with the prior work in the rest of this section. Note that in our utilization, the model is continuous and the weights are not necessarily symmetric [20].
Let $\Omega$ be a region in $\mathbb{R}^n$, and $u : \Omega \to \mathbb{R}$ be a real function. We define the nonlocal derivative:

$$\frac{\partial u}{\partial y}(x) := \frac{u(y) - u(x)}{d(x,y)}, \text{ for all } x, y \in \Omega$$

where $d$ is a positive distance between $x$ and $y$. With the following non-local weight defined as (2.3), we can re-write the non-local derivative as (2.4).

$$w(x,y) = d^{-2}(x, y)$$

$$\frac{\partial u}{\partial y}(x) = \sqrt{w(x,y)}(u(y) - u(x))$$

With this non-local derivative, we can define the non-local gradient $\nabla_w u$ for $u \in L^2(\Omega)$ as a function from $\Omega$ to $L^2(\Omega)$; we therefore use the notation $\nabla_w u \in L^2(\Omega, L^2(\Omega))$. Then $\nabla_w u \in L^2(\Omega, L^2(\Omega))$ is the collection of all partial derivatives:

$$\nabla_w u(x)(y) = \frac{\partial u}{\partial y}(x) = \sqrt{w(x,y)}(u(y) - u(x))$$

We use standard $L^2$ inner product on Hilbert spaces $L^2(\Omega)$ and $L^2(\Omega, L^2(\Omega))$. More specifically,

$$\langle u_1, u_2 \rangle := \int_{\Omega} u_1(x)u_2(x)dx, \text{ for all } u_1, u_2 \in L^2(\Omega)$$

$$\langle v_1, v_2 \rangle := \int_{\Omega} \int_{\Omega} v_1(x)(y)v_2(x)(y)dydx, \text{ for all } v_1, v_2 \in L^2(\Omega, L^2(\Omega))$$

With the above definition of inner products and nonlocal gradient, the nonlocal divergence $\text{div}_w$ is defined as the negative adjoint of the nonlocal gradient:

$$\text{div}_w v(x) := \int_{\Omega} \sqrt{w(x,y)}v(x)(y) - \sqrt{w(y,x)}v(y)(x)dy$$

We can also define an $L^1$ and $L^\infty$ norm on the space $L^2(\Omega, L^2(\Omega))$:

$$\| v \|_{L^1} := \int_{\Omega} \| v(x) \|_{L^2} dx = \int_{\Omega} \int \| v(x)(y) \|_{L^2}^2 dy^{1/2} dx$$

$$\| v \|_{L^\infty} := \sup_x \| v(x) \|_{L^2}$$

So particularly, if $u \in L^2(\Omega)$ is an image on the region $\Omega$, then

$$\| \nabla_w u \|_{L^1} = \int_{\Omega} \int \nabla_w u(x)(y)^2 dy^{1/2} dx$$

$$\| \nabla_w u \|_{L^\infty} := \sup_x \| \nabla_w u(x) \|_{L^2}$$
2.2 Primal Dual Algorithm

Using total variation semi-norm as a regularizer in image processing involves $L^1$ norm on the gradient, which is nontrivial to minimize. Specifically, using the gradient descent method on $\|\nabla u\|_{L^1}$ involves calculating $\text{div}(\frac{\nabla u}{|\nabla u|})$, which is highly unstable because $|\nabla u|$ can be equal to zero.

In 2011, Chambolle and Pock introduced a first-order primal dual algorithm, which they proved converged to a saddle point with a rate of $O(1/N)$ in finite dimensions for the complete class of convex problems. We introduce the framework here to contextualize our extension to nonlocal model for hyperspectral imagery.

Let $X$ and $Y$ be two finite-dimensional real vector spaces. Let $F$ and $G$ be proper convex lower semi-continuous functions $F : Y \rightarrow [0, \infty], G : X \rightarrow [0, \infty]$, and $F^*$ the convex conjugate, and $K : X \rightarrow Y$ a continuous linear operator with the operator norm:

$$\|K\| = \sup\{\|Kx\| : x \in X, \|x\| \leq 1\} \quad (13)$$

For our total variation energy functional (1), we define the gradient $\nabla$ as $K$, $\|\cdot\|_{L^1}$ as $F$, and the fidelity term $\frac{\lambda}{2}F(u)$ as $G$. Then minimizing (1) is equivalent to minimizing the primal problem:

$$\min_{x \in X} \{F(Kx) + G(x)\} \quad (14)$$

This minimization is the nonlinear primal problem with the corresponding dual problem:

$$\max_{y \in Y}\{-(G^*(-K^*y) + F^*(y))\} \quad (15)$$

Athen the primal dual formulation of equations (14) and (15) is the saddle-point problem:

$$\min_{x \in X} \max_{y \in Y}\{\langle Kx, y \rangle - F^*(y) + G(x)\} \quad (16)$$

The following saddle-point problem is then solved using subsequent iterations of Algorithm 1 in [10] that converges with a rate of $O(1/N)$.

**Primal Dual Algorithm**

- Initialization: Choose $\tau, \sigma > 0, \theta \in [0, 1]$, $(x^0, y^0) \in X \times Y$, and set $\bar{x}^0 = x^0$
- Iterations ($n \geq 0$): Update $x^n, y^n, \bar{x}^n$ as follows:

$$\begin{align*}
  y^{n+1} &= (I + \sigma \partial F^*)^{-1}(y^n + \sigma K\bar{x}^n) \\
  x^{n+1} &= (I + \tau \partial G)^{-1}(x^n - \tau K^*y^{n+1}) \\
  \bar{x}^{n+1} &= x^{n+1} + \theta(x^{n+1} - x^n)
\end{align*}$$
3 Two Total Variational Models for Classification of HSI

3.1 Linear Model

We extend the idea from [10] to formulate a linear model for classification on hyperspectral imagery. The linear model seeks to minimize the following energy

\[ E_1(u) = \| \nabla w u \|_{L^1} + \langle u, f \rangle \]

\[ = \| \nabla w u \|_{L^1} + \sum_{l=1}^{k} \int_{\Omega} u_l(x) f_l(x) dx \]  

(17)

where \( u = (u_1, u_2, \ldots, u_k) : \Omega \rightarrow \mathbb{K}^k \) is the labeling function, \( k \) is the number of clusters, \( \mathbb{K}^k \) is the unit simplex in \( \mathbb{R}^k \), \( \nabla w u \) = \( (\nabla w u_1, \nabla w u_2, \ldots, \nabla w u_k) \) such that \( \| \nabla w u \|_{L^1} = \sum_{l=1}^{k} \| \nabla w u_l \|_{L^1} \).

Then let \( f_l(x) \) be the error function defined as

\[ f_l(x) = \frac{\lambda}{2} | g(x) - c_l |^2 \]  

(18)

where \( g(x) \) is the spectral signature at pixel \( x \) and \( c_l \) is the spectral signature of the \( l \)th centroid, which is either picked randomly on the hyperspectral image or generated by any fast unsupervised unmixing model. The distance in the above definition of \( f_l(x) \) is a linear combination of cosine distance and Euclidean distance:

\[ | g(x) - c_l | = 1 - \frac{< g(x), c_l >}{\| g(x) \|_2 \| c_l \|_2} + \mu \| g(x) - c_l \|_2 \]  

(19)

Motivation for the linear combination of cosine and Euclidean distance is discussed in Section 4.1.

We call this the linear model since the power of the labeling function \( u_l \) is 1. Our extension of [10] is two-fold: First, we replace the local gradient with the nonlocal one for accuracy in image segmentation. We will see that the usage of sparse weight matrices and primal dual algorithm can considerably reduce the CPU run time compared to the numerical scheme introduced in [19]. Second, we are classifying a hyperspectral image instead of an RGB image.

First, we address the creation of a weight matrix measuring differences in spectral signatures for calculating the nearest neighbors of a pixel in hyperspectral imagery. Following the idea from [19], we define the patch distance as:

\[ d_{\sigma}(x, y) = \int_{\Omega} G_{\sigma}(t)|g(x+t) - g(y+t)|^2 dt \]  

(20)

where \( G_{\sigma} \) is a Gaussian of standard deviation \( \sigma \). Patch distance in RGB image processing is known to address the situation when the image is noisy and repetitive in space. Then we discretize and binarize the weights in the following way: we take a patch and a search window around every pixel \( i \), and compute the patch distance \( (d_{\sigma})_{i,j} \) to all the points within the search window and select the \( m \) closest ones. For those \( m \) closest pixels, \( w_{i,j} \)'s are set to be 1, while all the other weights are set to be 0. In our experiments, we have used either \( 3 \times 3 \) or \( 11 \times 11 \) patches, and \( 21 \times 21 \) search windows, and \( m \) is set to be 10. There is one more thing to mention about the patch size: unlike RGB image processing, the patch size for hyperspectral image processing doesn’t really have to be very large, and \( 3 \times 3 \) patches, or in some extreme cases \( 1 \times 1 \) patches, can generate similar results.
to that of $11 \times 11$ patches. The reason is in RGB images, one cannot tell what a pixel represents by simply looking into its RGB bands, and the identity of that pixel is revealed by a large patch of surrounding pixels; however, this is not the case for hyperspectral imagery. Every material has its unique spectral signature, thanks to a large number of bands collected in the data cube, and surrounding pixels are not required to identify the pixel in the middle as long as the hyperspectral imagery is not contaminated by noise. Of course, a larger patch size will still be preferable when significant noise is present.

We can also discretize the labeling function and nonlocal operators. More specifically, $u = (u_1, u_2, \ldots, u_k)$ is a matrix of size $r \times k$, where $r$ is the number of pixels in the hyperspectral image, and $(u_l)$ is the $l$-th labeling function at $j$-th pixel; $(\nabla_w u_l)_{i,j} = \sqrt{\frac{v_{i,j}}{v_{i,j}}}(u_l)_{j} - (u_l)_{i}$ is the nonlocal gradient of $u_l$; $(\nabla_w u_l)_{i,j} = \sum_j \sqrt{w_{i,j}}v_{i,j} - \sqrt{w_{i,j}}v_{i,j}$ is the divergence of $v$ at $i$-th pixel; and the discrete $L^1$ and $L^\infty$ norm of $\nabla_w u_l$ are defined as:

\[
\| \nabla_w u_l \|_{L^1} = \sum_i (\sum_j ((\nabla_w u_l)_{i,j})^2)^{\frac{1}{2}} \quad (21)
\]

\[
\| \nabla_w u_l \|_{L^\infty} = \max_{i,j} (\sum_j ((\nabla_w u_l)_{i,j})^2)^{\frac{1}{2}} \quad (22)
\]

Next, we will use similar idea from [10] to derive an algorithm for minimizing the energy $E_1$ in (17). By adding an indicator function $\delta_{U_1}$, minimizing $E_1$ is equivalent to solving the following primal problem:

\[
\min_u \| \nabla_w u \|_{L^1} + \langle u, f \rangle + \delta_{U_1}(u) \quad (23)
\]

Where $U = \{ u = (u_1, u_2, \ldots, u_k) \in \mathbb{R}^{r \times k} \}$, and $\delta_{U_1}$ is the indicator function on $U$. More specifically:

\[
\delta_{U_1}(u) = \begin{cases} 
0 & \text{if } u \in U \\
\infty & \text{otherwise}
\end{cases} \quad (24)
\]

The primal-dual formulation of (23) is:

\[
\min_{u=(u_l)_{l=1}^k} \max_{p=(p_l)_{l=1}^k} \langle \nabla_w u, p \rangle + \langle u, f \rangle + \delta_{U_1}(u) - \delta_{P}(p) \quad (25)
\]

Where $\delta_P$ is the convex conjugate of $\| \cdot \|_{L^1}$, and the set $P = \{ p \in \mathbb{R}^{(r \times r) \times k} : \| p_l \|_\infty \leq 1 \}$.

We can then use the primal dual algorithm to solve the above saddle point problem:

**Primal-Dual Iterations for Linear Model**
- Iterations ($n \geq 0$): Update $u^n, p^n, \bar{u}$ as follows:

\[
\begin{align*}
\rho^{n+1} &= \text{proj}_P(p^n + \sigma \nabla_w \bar{u}^n) \\
u^{n+1} &= \text{proj}_{U_1}(u^n + \tau \nabla_w p^{n+1} - \tau f) \\
u^{n+1} &= u^{n+1} + \theta(u^{n+1} - u^n)
\end{align*}
\]

We specify the two orthogonal projections in the algorithm above: let $\tilde{p} = \text{proj}_P(p)$, where $p = (p_l)_{l=1}^k \in \mathbb{R}^{(r \times r) \times k}$. Then for every $i \in \{1,2,\ldots,r\}$ and every $l \in \{1,2,\ldots,k\}$, the $i$-th row of $\tilde{p}$ is the projection of the $i$-th row of $p_l$ on to the unit ball in $\mathbb{R}^r$. Similarly, if $\tilde{u} = \text{proj}_{U_1}(u)$, then for every $i \in \{1,2,\ldots,r\}$, $((\tilde{u}_1)_i, (\tilde{u}_2)_i, \ldots, (\tilde{u}_k)_i)$ is the projection of $((u_1)_i, (u_2)_i, \ldots, (u_k)_i)$
onto the unit simplex $K^k$ in $\mathbb{R}^k$. And from [24], we know at most $k$ steps are needed to project an arbitrary vector in $\mathbb{R}^k$ onto $K^k$.

Lastly, we address centroid updates and stopping criteria for our linear model. The concept of centroid update in the linear model is not a new one; in fact, the standard k-means algorithm consists of two steps: first, it assigns each point to a cluster whose mean yields the least within-cluster sum of squares, then it re-calculates the means from the centroids, and terminates when assignments no longer change.[9] Especially for data-based methods, re-calculating the centroid is essential for making the algorithm less sensitive to initial conditions and more likely to find the “true” clusters.

After every few steps of primal-dual iterations (we used five steps in our experiments), the output $u$ will be thresholded to $u_{\text{hard}}$. More specifically, for every $i \in \{1, 2, \ldots, r\}$, we pick the largest element among $((u_1)_i, (u_2)_i, \ldots, (u_k)_i)$ and set it as 1, while leaving the others 0, and we say the $i$-th pixel belongs to that particular cluster. Then we update the $l$-th centroid by taking the mean of all the pixels in that cluster. We repeat the process until the difference between two consecutive $u_{\text{hard}}$’s drops below a certain threshold. The following is the final algorithm for our linear model on HSI:

```
Algorithm 1 for Linear Model
Data: hyperspectral cube, $(c_l)_{l=1}^k$ (randomized or generated by fast unsupervised endmember-extraction algorithm)
Result: classification on HSI
initialization: Choose $\tau, \sigma > 0$, $\theta \in [0, 1]$, random $u^0 \in \mathbb{R}^{r \times k}$ and $p^0 \in \mathbb{R}^{(r \times r) \times k}$, set $\bar{u}_0 = u^0$, $u_{\text{hard}} = u_{\text{hard}_0} = \text{threshold}(u^0)$, $\text{count} = 0$;
while $\|u_{\text{hard}} - u_{\text{hard}_0}\| > \text{tol}$ or $\text{count} < 5$ do
  $u_{\text{hard}_0} = u_{\text{hard}}$;
  $\text{count} = \text{count} + 1$;
  $N$ steps of primal-dual iterations for linear model;
  $u_{\text{hard}} = \text{threshold}(u)$;
  update $(c_l)_{l=1}^k$;
end
```

3.2 Quadratic Model

3.2.1 Intuition

Our linear model performs very well when the centroids are initialized by accurate endmember-extraction algorithms. As we will show in Section 5.2, the linear model can have a significant boost to the accuracy of other algorithms if the endmember extraction algorithm is accurate, without sacrificing speed. However, if endmembers are not extracted accurately, or if random pixel initialization is used, the segmenting results are no longer accurate, and the algorithm takes far more iterations to converge to a stable segmentation.

To reduce the times of centroid updates and merge similar clusters automatically and simultaneously, we propose the following quadratic model:

$$E_2(u) = \|\nabla_w u\|_{L_1} + \sum_{l=1}^k \int w_l^2(x) f_l(x) dx$$  \hspace{1cm} (26)
Similarly to before, $u = (u_1, u_2, \ldots, u_k): \Omega \rightarrow \mathbb{R}^k$ is the labeling function, $k$ is the number of clusters, $\mathbb{R}^k$ is the unit simplex in $\mathbb{R}^k$, $\nabla_w u = (\nabla_w u_1, \nabla_w u_2, \ldots, \nabla_w u_k)$ such that $\|\nabla_w u\|_{L^1} = \sum_{l=1}^k \|\nabla_w u_l\|_{L^1}$, and $f_l(x)$ is the error function term defined as:

$$f_l(x) = \frac{\lambda}{2} \|g(x) - c_l\|^2$$

Note that the only difference between (17) and (26) is that the power of the labeling function $u_l$ here is 2. The intuition for this is as follows:

Consider for simplicity we have a hyperspectral image with a ground truth of only two clusters, $A_1$ and $A_2$. Suppose we initialize the centroids randomly with both the linear and quadratic models, minimizing energies $E_1$ (17) and $E_2$ (26). Suppose our initial pixels are chosen such that $c_1 \approx c_2 \in A_1$; or, that the two random initial pixels are of very similar spectral signature and belong to the same ground truth cluster.

Let $x$ be a pixel from $A_2$. Then $0 \ll |g(x) - c_1|^2 \approx |g(x) - c_2|^2$. When we apply (17), the fidelity term $\langle u, f \rangle$ does not change when $u(x)$ moves on the simplex in $\mathbb{R}^2$, and thus pixels of $A_2$ will be scattered randomly on the simplex. After thresholding, an approximately equal number of pixels from cluster $A_2$ will belong to clusters $C_1$ and $C_2$, so the new centroids $\bar{c}_1$ and $\bar{c}_2$ that are the mean of the spectral signatures of the current clusters will once again be approximately equal.

This situation changes dramatically when we minimize (26). Observe that the fidelity term is minimized for a pixel $x \in A_2$ when $u_1 \approx u_2 \approx \frac{1}{2}$. Therefore, the pixels of cluster $A_2$ will be “pushed” toward the center of the simplex. With a stable simplex clustering method (explained in Section 3.2.4), we divide the clusters such that all of these pixels in the center belong to either $C_1$ or $C_2$; without loss of generality suppose they belong to $C_2$. Then the updated centroid $\bar{c}_1$ is essentially $c_1$, while the updated centroid $\bar{c}_2$ is a linear combination of the spectral signature endmembers of $A_1$ and $A_2$, and thus quite different from the original $c_2$. After a few iterations of primal-dual algorithm, pixels from $A_1$ will be clustered in $C_1$, and pixels from $A_2$ will be pushed to $C_2$. Therefore, we will finish the clustering in just two steps.

The quadratic model not only reduces the number of iterations needed to find the “true” clustering because of its anomaly distinction, but it allows for random initialization as well, making it a more robust technique.

### 3.2.2 Primal Dual

We use primal dual algorithm to minimize the energy $E_2$ in (26). As in the linear model, we get an unconstrained primal problem by adding an indicator function $\delta_U$:

$$\min_u \|\nabla_w u\|_{L^1} + \sum_{l=1}^k \int u_l^2(x) f_l(x) dx + \delta_U(u)$$

where $U = \{u = (u_1, u_2, \ldots, u_k) \in \mathbb{R}^{r \times k}\}$. The primal-dual formulation of the above problem is:

$$\min_{u=(u_l)} \max_{p=(p_l)} (\nabla_w u, p) + \sum_{l=1}^k \int u_l^2(x) f_l(x) dx + \delta_U(u) - \delta_D(p)$$

Figure 1: "pushing" mechanism of the quadratic model.
where again $\delta_P$ is the convex conjugate of $\| \cdot \|_{L^1}$, and the set $P = \{ p \in \mathbb{R}^{(r \times r) \times k} : \| p \|_\infty \leq 1 \}$. Note that this saddle point problem cannot be cast into the form in [10], so we build from the concepts to derive a new algorithm that can solve the problem.

Assume that we have already obtained $u^n$, $\bar{u}^n$ and $p^n$. Then we implement the following measures:

**Step 1: Determine Max of $p$:**

We are to solve the following optimization problem with respect to $p$:

$$p^{n+1} = \arg\max_p \langle \nabla_w \bar{u}^n, p \rangle - \delta_P(p) - \frac{1}{2\sigma} \| p - p^n \|^2$$

(29)

We add the last term $\frac{1}{2\sigma} \| p - p^n \|^2$ so that $p^{n+1}$ will be close to $p^n$. Taking the subgradient, we have:

$$0 \in -\nabla_w \bar{u}^n + \partial \delta_P(p^{n+1}) + \frac{1}{\sigma} (p^{n+1} - p^n)$$

(30)

$$p^{n+1} = (I + \sigma \partial \delta_P)^{-1}(p^n + \sigma \nabla_w \bar{u}^n)$$

(31)

Or equivalently,

$$p^{n+1} = \text{proj}_P(p^n + \sigma \nabla_w \bar{u}^n)$$

(32)

**Step 2: Determine Min of $u$:**

We will now alternate the direction and minimize $u$:

$$u^{n+1} = \arg\min_u \langle \nabla_w p^{n+1}, u \rangle + \langle u, f \circ u \rangle + \delta_U(u) + \frac{1}{2\tau} \| u - u^n \|^2$$

(33)

where $f \circ u$ denotes the pointwise product between $f$ and $u$. We define a linear operator $A : \mathbb{R}^{r \times k} \rightarrow \mathbb{R}^{r \times k}$ such that $\frac{1}{2} Au = f \circ u$, then $A$ is a positive semidefinite diagonal matrix of size $rk \times rk$. Taking the subgradient of the above optimization problem, we have:

$$0 \in -\tau \text{div}_w p^{n+1} + \tau Au^{n+1} + \tau \partial \delta_U(u^{n+1}) + (u^{n+1} - u^n)$$

(34)

$$0 \in (I + \tau A)u^{n+1} + \tau \partial \delta_U(u^{n+1}) - (u^n + \tau \text{div}_w p^{n+1})$$

(35)

Or equivalently,

$$u^{n+1} = \arg\min_u \delta_U(u) + \frac{1}{2} \| (I + \tau A)^{\frac{1}{2}} u - (I + \tau A)^{-\frac{1}{2}} (u^n + \tau \text{div}_w p^{n+1}) \|^2$$

(36)

Noticing that the matrix $A$ is diagonal and positive semidefinite, it’s easy to compute the inverse and square root of $(I + \tau A)$. Problem (36) is like a preconditioned projection on to a unit simplex $\mathbb{K}^k$, and we will discuss how to solve it exactly in at most $k$ steps in the next section.

Combining (32) and (36), we have the primal-dual iterations for our quadratic model. We present the following algorithm:
Primal-Dual Iterations for Quadratic Model

- Iterations \((n \geq 0)\): Update \(u^n, p^n, \bar{u}^n\) as follows:

\[
\begin{aligned}
    p^{n+1} &= \text{proj}_{p}(p^{n} + \sigma \nabla w \bar{u}^{n}) \\
    u^{n+1} &= \underset{u}{\text{argmin}} \, \delta_U(u) + \frac{1}{2} \| (I + \tau A)^{1/2} u - (I + \tau A)^{-1/2} (u^{n} + \tau \text{div} w p^{n+1}) \|^2 \\
    \bar{u}^{n+1} &= u^{n+1} + \theta (u^{n+1} - u^{n})
\end{aligned}
\]

3.2.3 Preconditioned Projection onto the Unit Simplex

In this section, we address how to solve problem (36). By looking at every row of \(u\), we only need to know how to solve the following problem:

\[
\begin{aligned}
    \min_{u \in \mathbb{R}^k} \delta_U(u) + \frac{1}{2} \| Au - y \|^2
\end{aligned}
\]

where \(A\) is positive definite diagonal matrix of size \(k \times k\), \(U = \mathbb{K}^k\) is the unit simplex in \(\mathbb{R}^k\), and \(y \in \mathbb{R}^k\). Our method to solve the problem above is a direct generalization of [24].

Taking the subgradient, we have:

\[
\begin{aligned}
    0 &\in \partial \delta_U(u) + A(Au - y) \\
    \implies A(y - Au) &\in \partial \delta_U(u) \\
    \implies Au &\in A \partial \delta_U(U)(y - Au) \\
    \implies y - Au = \text{prox}_{1, \delta_U(U)}(y) &= \underset{z}{\text{argmin}} \{ \delta_U(Az) + \frac{1}{2} \| z - y \|^2 \} \\
    \implies u &= A^{-1}(y - \underset{z}{\text{argmin}} \{ \delta_U(Az) + \frac{1}{2} \| z - y \|^2 \})
\end{aligned}
\]

Therefore we must compute \(\underset{z}{\text{argmin}} \{ \delta_U(Az) + \frac{1}{2} \| z - y \|^2 \}\). If \(A = \text{diag}(a_1, a_2, \ldots, a_k)\), then:

\[
\begin{aligned}
    \underset{z}{\text{argmin}} \{ \delta_U(Az) + \frac{1}{2} \| z - y \|^2 \} &= \underset{z}{\text{argmin}} \max_{1 \leq l \leq k} \{ a_l z_l \} + \frac{1}{2} \| z - y \|^2 
\end{aligned}
\]

We will show that it suffices to solve the following problem:

\[
\begin{aligned}
    \min_t \min_z \{ t + \frac{1}{2} \| z - y \|^2, z_t \leq \frac{t}{a_t} \}
\end{aligned}
\]

**Theorem 1.** If \(z^*\) is a solution of (39), then \(z^t := \underset{z}{\text{argmin}} \{ t^* + \frac{1}{2} \| z - y \|^2, s.t. \, z_t \leq \frac{t^*}{a_t} \}\) satisfies \(\max_{1 \leq l \leq k} \{ a_l z_l^* \} = t^*\), and \(z^t\) is a solution of (38).
Proof. First, to prove that $\max \{z_l^* a_l\} = t^*$. If not, then $z_l^* a_l < t^*, \forall l$. Hence, there exists a $\tilde{t} < t^*$ such that $z_l^* a_l \leq \tilde{t}$, and we have:

$$\min_{z} \left\{ \tilde{t} + \frac{1}{2} \| z - y \|_2^2, \text{s.t. } z_l \leq \frac{\tilde{t}}{a_l} \right\} \leq \tilde{t} + \frac{1}{2} \| z^* - y \|_2^2$$

$$< t^* + \frac{1}{2} \| z^* - y \|_2^2$$

$$= \min_{z} \left\{ t^* + \frac{1}{2} \| z - y \|_2^2, \text{s.t. } z_l \leq \frac{t^*}{a_l} \right\}$$

$$= \min_{\tilde{t}} \min_{z} \left\{ \tilde{t} + \frac{1}{2} \| z - y \|_2^2, \text{s.t. } z_l \leq \frac{\tilde{t}}{a_l} \right\}$$

And that is a contradiction.

Next, we prove that $z_t^*$ is a solution of (38). If not, there exists a $\tilde{z}$ such that:

$$\max_{1 \leq l \leq k} \{a_l \tilde{z}_l\} + \frac{1}{2} \| \tilde{z} - y \|_2^2 < \max_{1 \leq l \leq k} \{a_l z_t^* l\} + \frac{1}{2} \| z_t^* - y \|_2^2$$

Let $\tilde{t} = \max_{1 \leq l \leq k} \{a_l \tilde{z}_l\}$, then $\tilde{z}_l \leq \frac{\tilde{t}}{a_l}$, and we have:

$$\min_{\tilde{z}} \left\{ \tilde{t} + \frac{1}{2} \| \tilde{z} - y \|_2^2, \text{s.t. } \tilde{z}_l \leq \frac{\tilde{t}}{a_l} \right\} \leq \tilde{t} + \frac{1}{2} \| \tilde{z} - y \|_2^2 < t^* + \frac{1}{2} \| z_t^* - y \|_2^2$$

$$= \min_{\tilde{t}} \min_{\tilde{z}} \left\{ \tilde{t} + \frac{1}{2} \| \tilde{z} - y \|_2^2, \text{s.t. } \tilde{z}_l \leq \frac{\tilde{t}}{a_l} \right\}$$

which is a contradiction. Therefore, $z_t^*$ is a solution of (38).

Now it suffices to solve (39). For any given $t$, if $z^t$ is the solution of the inner minimization of (39):

$$z^t := \arg\min_{z} \left\{ t + \frac{1}{2} \| z - y \|_2^2, \text{s.t. } z_l \leq \frac{t}{a_l} \right\} \quad (40)$$

then $z^t$ can be solve exactly:

$$(z^t)_l = \min (y_l, \frac{t}{a_l}) \quad (41)$$

For any given $y = (y_1, y_2, \ldots, y_k) \in \mathbb{R}^k$, we sort the components of $a$ and $y$ in the ascending order $a_{(1)} y_{(1)} \leq a_{(2)} y_{(2)} \leq \ldots \leq a_{(k)} y_{(k)}$. Let:

$$f(t) = \min_{z} \left\{ t + \frac{1}{2} \| z - y \|_2^2, \text{s.t. } z_l \leq \frac{t}{a_l} \right\} = t + \frac{1}{2} \| z^t - y \|_2^2 \quad (42)$$
then:
\[
f(t) = \begin{cases} 
  t + \frac{1}{2} \sum_{l=1}^{k} |y(l) - \frac{t}{a(l)}|^2 & t \leq a(1)y(1) \\
  t + \frac{1}{2} \sum_{l=1}^{k} |y(l) - \frac{t}{a(l)}|^2 & a(i-1)y(i-1) \leq t \leq a(i)y(i) \\
  t & t \geq a(k)y(k)
\end{cases}
\] (43)

In [24], it is shown that \( f \) is a piecewise quadratic function and \( f \in C^1(\mathbb{R}) \). The derivative \( f' \) of \( f \) is:
\[
f'(t) = \begin{cases} 
  1 + \sum_{l=1}^{k} \frac{1}{a(l)} \left( \frac{t}{a(l)} - y(l) \right) & t \in I_1 = (-\infty, a(1)y(1)] \\
  1 + \sum_{l=1}^{k} \frac{1}{a(l)} \left( \frac{t}{a(l)} - y(l) \right) & t \in I_i = [a(i-1)y(i-1), a(i)y(i)], 2 \leq i \leq k \\
  1 & t \in I_{k+1} = [a(k)y(k), \infty)
\end{cases}
\] (44)

Therefore, the solution \( t^* \) of (39) satisfies \( f'(t^*) = 0 \), and it is the unique \( t_i \) among \( \{t_1, t_2, \ldots, t_k\} \) that falls into the corresponding interval \( I_i \), where \( t_i = \left( \sum_{l=i}^{k} \frac{y(l)}{a(l)} - 1 \right) / \left( \sum_{l=i}^{k} \frac{1}{a(l)} \right) \). And the solution \( z^* \) of (38) is \( z^* = z^{*'} \), i.e. \( (z^*)_l = \min \left( y_l, \frac{t^*}{n_l} \right) \). Therefore, the solution \( u^* \) of (37) is \( u^* = A^{-1}(y - z^*) \).

### 3.2.4 Stable Simplex Clustering

As we have mentioned in the previous section, our quadratic model pushes anomalies into the middle of the unit simplex. Therefore, it would be ill-conceived to simply classify the pixels based on the largest component of the fitting function \( u(x) = (u_1(x), u_2(x), \ldots, u_k(x)) \). Instead, we need a stable simplex clustering.

![Figure 2: Stable Simplex Clustering](image)
The concept behind stable simplex clustering is the follows: to minimize the number of iterations of re-calculating the centroids, we wish to choose a division that puts all the data points in the “middle” into a single cluster. Figure 2 demonstrates this in the simple two-cluster case; note that as the dividing bar is fully on the left side of the anomaly. In the first stage, the anomaly is pushed to the middle with the quadratic minimization model, and the clusters are divided with the anomaly entirely on one side. In the second stage, the updated centroids of the clusters remain the true first centroid on the left, and a linear combination of the first centroid and the anomaly on the right. In the next iteration of primal dual, all points are in their “true” cluster; thus the stable simplex clustering allows the algorithm to converge in just two iterations. Refer to section 3.2.1 for explanation of the “pushing” process.

Our idea to accomplish this goal comes from [14]. Mathematically speaking, we first create a grid on a $k$-dimensional simplex, where $k$ is the number of endmembers, then we project both the data points and a particular grid point onto all the edges of that simplex to obtain a simplex clustering with respect to that grid point. The clustering segmentation is chosen among those grid points in a manner that minimizes an energy consisting of two terms: the first which rewards keeping clusters approximately the same size, ensuring no skewed data from clusters far too small, and the second which rewards sparsity of points on the edges near the division, which is the Y-shaped region on the 3-simplex in Figure 2. The energy that we use is:

$$g(\delta) = -\log(\prod_{l=1}^{k} F_l(\delta)) + \eta \exp(G(\delta))$$  \hspace{1cm} (45)$$

where $\delta$ is a grid point on the simplex, $F_l(\delta)$ is the percentage of data points in cluster $l$ when we are choosing $\delta$ as the thresholding grid point. $G(\delta)$ is the percentage of data points in the
intermediate regime, or the “Y-shaped region”. We choose the constant $\eta$ large enough such that stability has a bigger weight in the energy.

The quadratic model combined with stable simplex clustering produced remarkable results; Figure 3 demonstrates how this detected the chemical plumes in a frame with background centroids pre-calculated and random initialization for the final endmember; by the twelfth iteration, the gas plume is nearly perfectly segmented.

Finally, we present the results of the linear model and the quadratic model on the Urban dataset with random pixel initialization in Figure 4. The stopping criteria for number of iterations was visual confirmation of all six clusters appearing. The linear model took about 50 iterations and 273 seconds to converge, and the quadratic model took 4 iterations and 75 seconds to converge.

4 ANN Scheme For Weight Matrix Calculation

One of the largest time-sinks in our algorithm was the calculation of the sparse weight matrix, a measure of similarity between any pixel and its $m$ nearest neighbors, which is used in the primal dual calculation. Using the straightforward local window approach described in Section 3.1, for the Urban dataset on a Linux machine with Intel core i5, 3.3hz with 2GB of DDR3 ram, it took 78.5 seconds for computing the weight matrix with cosine distance and 111 seconds for the linear combination of the cosine and Euclidean distance to be calculated. Speeding up this calculation would thus improve the usability of our algorithm.

When building the weight matrix, we determine the $m$ nearest neighbors of a pixel by calculating
the distances from all the pixels in a \( w \times w \) local window \((m = 10 \text{ and } w = 21 \text{ in our experiments})\). In light of this, using an approximate nearest neighbor (ANN) search algorithm can be far faster than calculating the distance between all the pixels. We use a k-d tree approach as well as an ANN search algorithm to reduce the run-time.

A k-d tree, or k-dimensional tree, is a binary tree that recursively partitions a \( k \)-dimensional space \([13]\) \((k \text{ corresponds to the number of bands sampled in our case})\). The tree is organized as follows: the root node corresponds to the entire space, and each node in the tree corresponds to a partition in space. At each non-leaf node, we choose a dimension and a hyperplane perpendicular to that dimension’s axis which divides the partition into two sub-partitions. Data points on one side of the hyperplane belong to the left sub-tree, and data points on the other side belong to the right sub-tree. Modern algorithms to build a balanced k-d tree generally at worst converge in \( O(kn \log n) \) time, where \( k \) is the number of dimensions and \( n \) is the number of points \([12]\).

The space partitioning scheme offered by the k-d tree significantly reduces the time cost of nearest neighbor search. Let us consider the case of 1-nearest-neighbor search for simplicity. Intuitively, given a query data point \( q \), if we have found a candidate data point \( x \) that lies on one side of a hyperplane corresponding to a non-leaf node, and the hypersphere centered at \( q \) with radius \( \|q - x\|_2 \) does not intersect with the hyperplane, then all the data points lying on the other side of the hyperplane can be excluded from consideration. This way, branches can be eliminated from the search space quickly. This algorithm converges in \( O(\log n) \) time \([13]\). We employ a randomized and approximate version of this algorithm \([27]\) implemented in the open source VLFeat package\(^1\).

We terminate the algorithm by placing an upper bound (say, 256) on the number of distance comparisons.

The benefit of the ANN scheme is two-fold:

1. We compute the distances and similarity values only after the \( m \) (approximate) nearest neighbors are identified. Thus the cost of computing distances becomes \( O(mnd) \) (excluding the cost of building and querying the k-d tree), as opposed to \( O(w^2nd) \) in the local window approach.

2. Without the presence of local windows, the ANN scheme enables “global search”, that is, any pixel in the entire image could be a candidate of the \( m \) nearest neighbors.

4.1 Cosine Vs. Euclidean Distance

When determining the distance between two spectral signatures, there are two metrics used: the cosine distance, which is sensitive to spectral angle, and the euclidean distance, which is sensitive to spectral amplitude. The cosine distance is generally used because it is more robust in the face of atmospheric interference, time of day, shade, and topographical features \([18]\).

When initially running experiments on the Urban dataset, we noticed that two classes, “road” and “dirt” had very similar spectral angles, but vastly different spectral amplitudes. We implemented a linear combination of the cosine distance and the Euclidean distance in calculating the weight matrix with our preliminary algorithm, which showed promising results on the Urban dataset. Figure 5 shows our initial experiments with supervised endmember extraction and no updating of the centroids. Simply using a linear combination of cosine and Euclidean distance allows the algorithm to correctly identify all of the road in the lower left corner instead of classifying them as dirt. Furthermore, it captures the dirt in the lower corner of the parking lot in the

\(^1\)http://www.vlfeat.org
The k-d tree generally uses the Euclidean distance to determine both the splitting of the leaf nodes, and the distance to the nearest neighbors. Thus using the ANN scheme with a non-Euclidean distance requires modification of the ANN search algorithm. We first normalize the spectral signatures using the L2 norm, and then the cosine distance between two points becomes:

\[ \| x - y \|_2^2 = \| x \|_2^2 + \| y \|_2^2 - 2(x, y) = 2(1 - (x, y)) \] (46)

the final term being the cosine distance. Therefore, we are able to use the cosine distance to construct the k-d tree and run the ANN search. Once the nearest neighbors are identified, the weight assigned to each of them is a linear combination of the cosine and Euclidean distances balanced by the parameter \( \mu \). Overall, this approximate way of nearest neighbor search and weight computation contributes to significantly reduced run time of computing the weight matrix.

5 Results

5.1 Comparison Methods

We ran the following algorithms on real-world hyperspectral datasets:

1. **K-means**: built in MatLab Code.
2. **NMF**: Non-negative Matrix Factorization, proposed in [26].

3. **H2NMF**: Hierarchical Non-negative Matrix Factorization, proposed in [14].

4. **CV-MBO**: Chan-Vese Merriman-Bence-Osher scheme, proposed in [3], with code provided by Zhaoyi Meng and Justin Sunu.

5. **NLTV**: Non-Local Total Variation, Cosine-Euclidean distance, Quadratic Model with random pixel initialization.

6. **NLTV, H2NMF init**: Non-Local Total Variation, Cosine-Euclidean distance, with end-members extracted from H2NMF. Quadratic Model unless otherwise specified as NLTV-1.

All experiments were run on Intel core i5, 3.3hz with 2GB of DDR3 ram.

### 5.2 Urban Data Set

The first dataset we examined was the Urban dataset from HYDICE, which is 307 x 307 pixels and contains 162 clean spectral bands. This dataset has the advantage of only having six classes of material: road, dirt, house, metal, tree, grass, and a seventh cluster detected by all algorithms of a mixture of grass and dirt.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Run-Time</th>
<th>6 Clusters</th>
<th>Accuracy, uncorrected</th>
<th>Accuracy, corrected</th>
</tr>
</thead>
<tbody>
<tr>
<td>K-Means</td>
<td>6.68 s</td>
<td>49.58%</td>
<td>49.58%</td>
<td>47.98%</td>
</tr>
<tr>
<td>NMF</td>
<td>87.45 s</td>
<td>65.67%</td>
<td>58.70%</td>
<td></td>
</tr>
<tr>
<td>H2NMF</td>
<td>7.28 s</td>
<td>73.20%</td>
<td>79.28%</td>
<td></td>
</tr>
<tr>
<td>CV-MBO</td>
<td>92.53 s</td>
<td>65.23%</td>
<td>76.89%</td>
<td></td>
</tr>
<tr>
<td>NLTV</td>
<td>292.11 s</td>
<td>65.80%</td>
<td>87.61%</td>
<td></td>
</tr>
<tr>
<td>NLTV H2NMF init</td>
<td>100s</td>
<td>70.40%</td>
<td>89.09%</td>
<td></td>
</tr>
<tr>
<td>NLTV-1 H2NMF init</td>
<td>47s</td>
<td>71.2%</td>
<td>87.21%</td>
<td></td>
</tr>
</tbody>
</table>

![Figure 6: Urban Dataset Reconstructed RGB Image](image)

There was no ground-truth provided for the data, we used a structured sparse algorithm to initialize a ground truth division, then corrected areas pixel by pixel to provide a framework for numerical analysis of accuracy. As this “ground truth” was hand-corrected, it does not necessarily represent the most accurate segmentation of the image; however, it provides a basis for quantitative comparison. Values for $\lambda$ and $\mu$ were $10^9$ and $10^{-5}$ respectively; $\lambda$ represents the weight on the fidelity term, and $\mu$ represents the balance of the cosine and Euclidean distance. As the magnitude of the squared cosine distance is generally very small, a small $\mu$ and a large $\lambda$ are required to balance the scale.
The percent accuracies are displayed in the table above, as well as run-times. As can be seen, although our method took longer to run than the competing algorithms, it performed consistently at higher accuracy. Some of the accuracy percentages are misleading; for example, in the six clusters uncorrected for a grass and dirt cluster, H2NMF performs at 73.2% accuracy and NLTV performs at 65.8% accuracy, but this is only because the grass and dirt cluster is larger in NLTV, and thus more pixels are treated as “misclassified” in this situation; the corrected ground truth for six clusters with combined road and metal, and then a dirt and grass separate cluster, displays a more relevant accuracy, because all five algorithms detected dirt and grass as a separate cluster.

It is easier to see visually in Figure 7 that the NLTV algorithm performs best of the five algorithms tested; specifically, the NLTV algorithm alone distinguished all of the dirt beneath the parking lot and the intricacies of the road around the parking lot. The total variation process gives the segmented image smoother and more distinct edges, allowing for easier human identification of the clusters. The NLTV algorithms best picked out all the rooftops from the trees in the lower left segment of the image (note that in the ground truth, some of them are segmented in the road cluster because of the combination of road and metal). The NLTV algorithm also nearly perfectly segments the dirt, road, and roof in the park near all the houses.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>K-means</th>
<th>NMF</th>
<th>H2NMF</th>
<th>CV-MBO</th>
<th>NLTV</th>
<th>NLTV, H2NMF init.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pavia U</td>
<td>1.6s</td>
<td>4s</td>
<td>50s</td>
<td>203s</td>
<td>141s</td>
<td>146s</td>
</tr>
<tr>
<td>Pavia Centre</td>
<td>11s</td>
<td>30s</td>
<td>60s</td>
<td>575s</td>
<td>528s</td>
<td>499s</td>
</tr>
<tr>
<td>Deepwater Horizon</td>
<td>7s</td>
<td>15s</td>
<td>17s</td>
<td>224s</td>
<td>451 s</td>
<td>352 s</td>
</tr>
<tr>
<td>San Diego Airport</td>
<td>9s</td>
<td>4s</td>
<td>13s</td>
<td>329s</td>
<td>92s</td>
<td>112s</td>
</tr>
</tbody>
</table>

Table 1: Run-Times for Pavia University, Pavia Centre, Deepwater Horizon, and San Diego Airport

5.3 San Diego Airport Dataset

We examined the San Diego Airport dataset, provided by the HYDICE sensor, which is 400 x 400 pixels and contains 158 clean spectral bands. We had no provided ground truth; however, there was a RGB image provided in [14] Segmenting accuracy therefore must be determined visually. We used seven clusters. Values of $\lambda$ and $\mu$ were $10^6$ and $5^{-7}$ respectively.

After plotting the spectral signatures of various pixels in the scene, we managed to pinpoint some errors that were common for each algorithm. We will not go into detail about the NMF and H2NMF algorithms, which clearly don’t perform well on this dataset. K-means obtains some decent results, but splits the rooftops of the four buildings on the bottom right of the image into two distinct clusters, and fails to separate two different buildings at the top. The CV-MBO scheme fails on two accounts: it doesn’t cluster the building on the top
Figure 7: Clustering Results for Urban Dataset, number of cluster k=6
left and doesn’t account for the different rooftop types. Our algorithm with H2NMF initialization is significantly more accurate than H2NMF and CVMBO. It successfully picks out two different types of roof (cluster 3, light blue and 4, yellow on the RGB image), two different types of road (cluster 6, medium blue and 7, green), although the other type of road (cluster 5, dark blue) is mixed with one type of roof (cluster 3, light blue). The best result was obtained with the random initialization of NLTV, with the only problem that tree and grass (clusters 1 and 2, red) are mixed up. However, this mixing of grass and tree is actually the case for all the other algorithms. This means that our algorithm alone was able to identify six of the seven clusters correctly.

5.4 Pavia University Dataset

We examined the Pavia University dataset, provided by the ROSIS sensor, which is 610 x 340 pixels and contains 103 clean spectral bands. The ground truth provided for this dataset stated that nine endmembers were present in the image: meadows, trees, asphalt, bricks, bitumen, gravel, painted metal sheets, shadows, and bare soil. The ground truth for this dataset was very sparse; 80% of the data was marked as a part of a “zero” cluster that indicated the data was not being counted as part of the ground truth. This meant that the true number of clusters in the image was unknown, as there could be an unspecified number of different clusters in the “zero” cluster. Furthermore, when we examined the spectral signatures of the hand-labeled ground truth, we found that it was highly inaccurate; certain clusters that were marked as different in the ground truth had the same spectral signatures, and certain parts of single clusters had very different spectral signatures. Therefore, all determination of the usefulness of clustering algorithms must be done by
qualitative visual assessment. Values of $\lambda$ and $\mu$ used for NLTV were $10^7$ and $10^{-7}$ respectively.

As can be seen in the clustered result, the CV-MBO, NLTV, and H2NMF initialized NLTV methods both produce smoother results with less noise pixels. K-means and NMF fail to distinguish the parking lot asphalt in the lower left corner, and instead classify it as a different cluster; H2NMF, CV-MBO, NLTV, and H2NMF initialized NLTV all pick this up correctly. H2NMF and NLTV best pick up the circle of dry grass in the center of the field in the upper left corner; CV-MBO misses it entirely, and K-means and NMF classify the grass around it as multiple classes.
5.5 Pavia Centre Dataset

We examined the Pavia Centre dataset, provided by the ROSIS sensor, which is 1096 x 715 pixels and contains 102 clean spectral bands. The ground truth provided for this dataset stated that nine endmembers were present in the image: meadows, trees, asphalt, bricks, bitumen, water, bricks, tiles, shadows, and bare soil. Once again the ground truth was sparse, with 81% of the data in a “zero” cluster. Similar to the Parvia University dataset, with comparison of spectral signatures, we determined that the ground truth was inaccurate and could not be used as a measure of accuracy, so determination of the usefulness of clustering algorithms must be done by qualitative visual assessment. Values of $\lambda$ and $\mu$ used for NLTV were $10^{10}$ and $10^{-8}$ respectively.

Once again, the CV-MBO, NLTV, and H2NMF initialized NLTV produce smoother results, allowing for better human interpretation of the segmented images. NLTV also best picks out the road that horizontally crosses the bottom of the image. All in all, the clusters of the NLTV are the sharpest results.
5.6 Deepwater Horizon Oil Spill Dataset

We examined the Deepwater Horizon Oil Spill Dataset, collected with a spatial resolution of 2.2m, containing 1160 x 320 pixels and 360 spectral channels, covering the range 390-2450nm. Of these bands, 48 were removed due to water absorption noise, and we focused on the lower part of the data, retaining an image of 660 x 320 pixels on which we ran experiments. One of the problems we ran into was that it was impossible to know precisely what output was expected from the algorithms since the scene contains no information discernible to the human eye, and no RGB image. Since no ground truth is available, we interpreted the results very cautiously, based on the little information we did have. After looking at the spectral signatures of the pixels in the scene we determined that 3 distinct clusters were present in the image. The CV-MBO, NLTV, and H2NMF initialized NLTV all detected the presence of what appears to be oil in the reconstructed RGB image, and when hand-examining spectral signatures we determined to be different than water. Values of $\lambda$ and $\mu$ used were $10^9$ and $10^{-5}$ respectively.
5.7 Chemical Plume Dataset

We examined the chemical plume dataset, which consists of frames taken from a hyper-spectral video of the release of chemical plumes provided by the John Hopkins University Applied Physics Laboratory. These images were taken by long wave infrared spectrometers placed 2km from the release of the plume at an elevation of approximately 1300 feet. The image is 128 x 320 pixels, with 129 clean spectral bands. There was no ground truth provided for this data, so we assumed segmentation into four classes: chemical plume, sky, foreground, and mountain, and added a fifth cluster so that the noise pixels would not interfere with the segmenting.

Analyzing images for chemical plumes is a far more difficult problem than merely segmenting images, because the gas in the plumes is generally diffuse and thus very difficult to detect. This class of problem can thus be seen as anomaly detection, with applications in military surveillance, environmental protection, or a warning system for chemical manufacturers. Chemical plume detection generally faces the challenge of a high presence of noise in the picture that requires intensive preprocessing before segmentation schemes can be implemented. We ran all the algorithms on the image before it was denoised and the results are shown in Figure 16. Parameters for NLTV were \( \lambda = 10^{10} \) and \( \mu = 10^{-2} \) respectively.

We saw extremely promising results using hand-selected endmembers, but had a harder time picking up the plume with randomized endmembers due to its diffuse nature. It occurred to us that for an anomaly detection problem, the desired background would be known; we could extract the four background centroids from the provided background frame using our algorithm, and use those as endmembers along with a fifth arbitrary endmember, then apply this algorithm to the sets with chemical plumes. This worked with limited success. Finally, we attempted to use the H2NMF extract the initial endmembers, then run our algorithm, with similar levels of success of hand-selected endmembers.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Run-Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>K-Means</td>
<td>2.33 s</td>
</tr>
<tr>
<td>NMF</td>
<td>2.59 s</td>
</tr>
<tr>
<td>H2NMF</td>
<td>2.24 s</td>
</tr>
<tr>
<td>CV-MBO</td>
<td>18 s</td>
</tr>
<tr>
<td>NLTV, rand</td>
<td>98 s</td>
</tr>
<tr>
<td>NLTV, hand sel.</td>
<td>52 s</td>
</tr>
<tr>
<td>NLTV, H2NMF init</td>
<td>227 s</td>
</tr>
<tr>
<td>NLTV, back. init</td>
<td>80 s &amp; 64 s</td>
</tr>
</tbody>
</table>

Table 2: Chemical Plume Run-Times

With hand-selected endmembers, our algorithm converged in just six iterations, which meant that it only took 52 seconds to run, and it fully detected the chemical plume. Similarly, with H2NMF initialization, it took 227 seconds, but properly identified the chemical plume. Background identification took 80 seconds, then another 64 seconds for plume detection. With both background and random initialization, our algorithm is not robust; sometimes it performs extremely well, as seen in Figure 3 in Section 3.2.4, while sometimes it performs less accurately. However, whenever there was at least one initial centroid in the chemical plume, our algorithm could fully detect it.

We refer to CV-MBO as a sort of a “ground truth” for this dataset because the algorithm was developed working on this dataset, and it performs very well. Note that CV-MBO, while unsupervised, requires hand-picking the best from ten results, and thus is semi-supervised. K-means, NMF, and our algorithm with random endmembers all fail to produce desired results. H2NMF splits the chemical plume in two, and cannot detect the hole in the center. When we run our algorithm with hand-selected endmembers (namely, one endmember from within the plume), we pick up the plume with similar accuracy as CV-MBO. Similarly, if we use the endmembers extracted with H2NMF as initialization, we detect the plume. Finally, when we used our algorithm to detect four endmembers on a background frame before the chemical release, then pick a random
fifth endmember *not* in the chemical plume, we still see that we get better results than K-means, NMF, and random initialization.

## 6 Conclusion

In this paper we present the framework for a non-local total variational method for image segmentation, which we solve with the primal dual algorithm. We develop a linear and a quadratic version of this model; the linear version updates more quickly and can refine results produced by an endmember extraction algorithm like hierarchical non-negative matrix factorization, and the quadratic model provides a robust means of classifying hyperspectral images with randomized pixel initialization. To reduce computational time, we introduce a stable simplex clustering scheme, and utilize a k-d tree and approximate nearest neighbor search to compute the weight matrix.

We test this algorithm on six datasets, with very promising results. Our algorithm consistently performed with highest accuracy on urbanized datasets (Urban, San Diego Airport, Pavia U, Pavia C), both producing smoother results with easier visual identification of segmentation, and distinguishing classes of material that other algorithms were not able to differentiate. Our algorithm performed well on anomaly detection scenarios like the Deepwater Horizon Oil Spill and the Chemical Plume datasets; while we had no ground truth for Deepwater Horizon, our non-local total variational method was one of two in the five tested able to pick up on anything other than the water. With proper initialization on the chemical plume dataset, it performed on par with the Chan-Vese Merriman-Bence-Osher scheme developed specifically for this dataset. While the runtimes were not yet idealized, they were still comparable to the other methods, and the consistent higher accuracy on different types of datasets suggests that this technique is a more robust and
Nonlocal Total Variation and Primal Dual Algorithm on HSI

precise means of classifying hyperspectral imagery.

7 Future Directions

Firstly, while our algorithm performs with higher accuracy than the competing algorithms, it did take longer to converge; similarly it could not handle more than 7 centroids using the stable simplex clustering algorithm without slowing down immensely. Speeding up the algorithm without losing accuracy would be the next step in making its performance more competitive against existing algorithms.

Also, there are two parameters in our algorithm, $\lambda$ and $\mu$, that need to be determined experimentally. $\mu$ represents the balance between the cosine and Euclidean distance, and $\lambda$ represents the balance between the smoothing term and the fidelity term in the total variation semi-norm. While the values for these parameters showed clear patterns ($\lambda$ was optimized at either $10^6$ or $10^7$, and $\mu$ generally was within an order of magnitude of balancing the cosine and Euclidean distances at the same order of magnitude), it is a non-trivial problem to determine the optimal parameters.

Finally, our clustering method requires a predetermined number clusters to segment the image. It is beyond the scope of current clustering methods to determine the optimal number of clusters without human input. However, in terms of hyperspectral anomaly detection, the open question of if the image contains an anomaly or not means that the number of classes is unknown, even if a ground truth exists. Or, if an image contains no hand-labeled ground truth, the number of classes is not known. Ideally, clustering algorithms should be able to segment an image with minimal human input; therefore, coming up with a means of automatically detecting the number of clusters would set our method apart from others.

Acknowledgments

Thanks to Zhaoyi Meng and Justin Sunu, for providing and help with the CV-MBO code. This research is supported by NSF grants DMS-1118971, DMS-1045536, DMS-1118971, DMS-0914856, ONR grant N00014-12-0838 and AFOSR MURI FA9550-10-0569

References


Nonlocal Total Variation and Primal Dual Algorithm on HSI


[25] Antonin Chambolle, Vincent Caselles, Matteo Novaga, Daniel Cremers, Thomas Pock. An introduction to Total Variation for Image Analysis. 2009. hal-00437581
