ABSTRACT

APPLICATIONS OF K-MEANS AND SPECTRAL CLUSTERING TO HYPERSPECTRAL VIDEO SEQUENCES

By

Justin Sunu

August 2014

In this work, hyperspectral images are images that capture n different wavelengths in the infrared spectrum, where $200 \geq n \geq 10$, as opposed to the three visible light wavelengths captured in a standard image. We work with long wave infrared (LWIR) hyperspectral images, which inhabit the 8-15 μm wavelengths. The primary advantage of working with hyperspectral images is the ability to capture data that exists outside the visible spectrum, allowing for the extraction of data like invisible gases and anomalous harmful particles. With threats in the modern age that range from chemical attacks to radiation leakage, a method of isolating and extracting important features in hyperspectral images becomes a necessity.

The current attempts at extracting gas plume locations is to utilize a probability model and compares a dictionary of malicious gas plume signatures to all of the signals in a hyperspectral image. The main problem with this method is that it requires a dictionary of possible signatures, which may not be conceivable given that a synthetic gas can be created or an unknown gas can be used. I will utilize clustering methods, such as k-means and spectral clustering, instead to isolate the gas plume signatures and extract features of the landscape.

K-means successfully performs its job of obtaining useful information from a data set, being able to quickly demonstrate the better distance metric, as well as the viability of clustering methods to segment hyperspectral data. Due to the nature of k-means, it is not a candidate for consistently segmenting gas plume, as the results can vary greatly due to a small change in parameters. On the other hand, spectral clustering is able to accurately and consistently segment the gas plume. The open nature of spectral clustering also allows for the ability to further refine its results, allowing for segmentation of the gas to occur even when the gas becomes diffuse.

APPLICATIONS OF K-MEANS AND SPECTRAL CLUSTERING TO HYPERSPECTRAL VIDEO SEQUENCES

A THESIS

Presented to the Department of Mathematics and Statistics

California State University, Long Beach

In Partial Fulfillment

of the Requirements for the Degree

Master of Science in Applied Mathematics

Committee Members:

Jen-Mei Chang, Ph.D. (Chair)
Tangan Gao, Ph.D.
Chung-Min Lee, Ph.D.

College Designee:

Robert Mena, Ph.D.

By Justin Sunu

B.S., 2009, University of California, Davis

August 2014

WE, THE UNDERSIGNED MEMBERS OF THE COMMITTEE,

HAVE APPROVED THIS THESIS

APPLICATIONS OF K-MEANS AND SPECTRAL CLUSTERING TO HYPERSPECTRAL VIDEO SEQUENCES

By

Justin Sunu

COMMITTEE MEMBERS

Jen-Mei Chang, Ph.D. (Chair)	Mathematics & Statistics
Tangan Gao, Ph.D.	Mathematics & Statistics
Chung-Min Lee, Ph.D.	Mathematics & Statistics

ACCEPTED AND APPROVED ON BEHALF OF THE UNIVERSITY

Robert Mena, Ph.D.
Department Chair, Department of Mathematics and Statistics

California State University, Long Beach

August 2014

ACKNOWLEDGEMENTS

I would like to thank my family for their support.

I would like to thank Dr Jen-Mei Chang and Dr Andrea Bertozzi for their help in my research.

I would like to thank my committee members, Dr Chung-Min Lee and Dr Tangan Gao.

This work is supported by NSF grants DMS-0914856, DMS-1045536, and DMS-1118971.

TABLE OF CONTENTS

Page

ACKNOWLEDGEMENTS iii

LIST OF TABLES v

LIST OF FIGURES vi

LIST OF ALGORITHMS viii

CHAPTER

1. INTRODUCTION 1

2. BACKGROUND LITERATURE 3

    Hyperspectral Data Set 3
    Distance Metrics 10

3. METHODS 15

    Clustering 15
    K-Means 16
    Spectral Clustering 23
    Nyström Extension 27

4. EXPERIMENTAL RESULTS AND DISCUSSION 33

    Experiments 33

5. SUMMARY AND CONCLUSIONS 49

    Summary 49

APPENDIX 51

BIBLIOGRAPHY 58

LIST OF TABLES

TABLE		Page
1	Different Measurements of Light	7
2	Table of Different Distance Metrics	12
3	An Example of a Graph Laplacian	27
4	Eigenpairs Computed From Graph Laplacian in Table 3	28

LIST OF FIGURES

FIGURE		Page
1	Decomposition of a clown image.	4
2	Sample hyperspectral image frame in one spectrum.	5
3	Two and three layer models for hyperspectral images.	6
4	Example of a linear mixture model.	6
5	Image of camera locations.	9
6	Scaled and unscaled sample data.	9
7	Euclidean distance between points.	11
8	Cosine distance between vectors.	11
9	Bad distance example.	11
10	Comparison between the cosine and modified cosine distance metric.	14
11	Sample hard clustering result.	15
12	Sample fuzzy clustering result.	16
13	One sample iteration of K-means.	18
14	Example of good and bad initialization in K-means clustering.	21
15	Example of good and bad K initialization.	22
16	An example of K-means clustering Two Moons data set.	23
17	Three different σ's.	26
18	Sample connected graph.	27
19	Plot of second eigenvector demonstrating a line of separation.	28
20	Plot of third eigenvector with separation line.	29
21	Showing data points in $\mathbb{R}^2$, along with corresponding coordinates.	30

FIGURE Page

22 Results after K-means. 30
23 Example of 3-d median filter. 35
24 Romeo and Victory summation images. 35
25 Histogram of 3D median filtered data. 35
26 Mountain circled in solid line. 35
27 Examples of K-means result with different distance metrics. 37
28 Components from K-means Victory results on a frame. 37
29 Different components from K-means Romeo results. 38
30 Mountain border with the Euclidean and cosine metrics. 38
31 Challenges arising from implementing K-means. 40
32 Spectral clustering results from a background frame. 42
33 Spectral clustering results from a gas plume frame. 43
34 Analyzing the first three spectral clustering eigenvectors. 44
35 Two spectral clustering eigenvectors of gas plume. 45
36 Nyström extension results from a gas plume frame. 46
37 First four eigenvectors of multiframe Nyström extension. 47

LIST OF ALGORITHMS

1 K-Means Algorithm 17

2 Modified K-Means Algorithm 24

3 Spectral Clustering [3] [5] 25

4 Nyström Extension [**?**] 32

5 3-d median filter 34

CHAPTER 1

INTRODUCTION

In the modern age, a great need arises for detection of anomalies in a given situation. Biochemical weapons, radiation leakage, and more are capable of occurring anywhere at any given time, and it is a necessity to detect these anomalies. In the cases of chemicals, these gases may be colorless, like carbon monoxide, so the ability to detect colorless components is important. A modern solution to this problem is the utilization of hyperspectral imaging. A hyperspectral image consists of multiple spectrum, anywhere from twenty-five to over one-hundred, that are outside the visible light spectrum. These images are able to incorporate electromagnetic bands that contain information on gases, information that can not be seen in the visible light spectra.

The most common approach to locate gases in hyperspectral images consists of working with individual pixels and treating each pixel as the only source of information. Two of the currently popular methods are Automated Matched Subspace Detector (AMSD) and Adaptive-Cosine Estimator (ACE). These methods work by looking at an individual pixel and generating a probability distribution for each pixel, giving the likelihood that a specific pixel is a target signature. These methods can be very effective in approximating the presence of certain target signatures, however they do not look at the image as a whole. This is where the classical clustering methods are utilized. The approach considered in this thesis are classical clustering algorithms that incorporate the image geometry and patterns present in the image, namely K-means and spectral clustering.

K-means is an iterative algorithm that focuses on finding k clusters by assigning points based upon the closest distance to centroid calculation. It is considered an entry level clustering method, that is used to obtain general information on the data as well as test to see if clustering is possible on the data set. On the other hand, spectral clustering is a more sophisticated clustering method that searches for the minimum number of cuts to separate the graph into two or more parts. It uses the eigenvectors of the graph Laplacian to formulate these cuts.

The data set is provided by Johns Hopkins Applied Physics Laboratory [2]. It is a hyperspectral movie, a series of images taken at a set time interval, of a gas plume release at the Dugway Proving ground. Due to the sensitive nature of the hyperspectral cameras, each image is quite noisy and the dusty terrain may cause problems with the camera lens.

In this thesis I will start by introducing hyperspectral data, as well as some specific properties to the data set that I investigated. Then I will discuss the two distance metrics that are used for classification. After introducing and explaining the clustering algorithms, I will finish off the thesis with the experimental results produced from the clustering algorithm.

CHAPTER 2

BACKGROUND LITERATURE

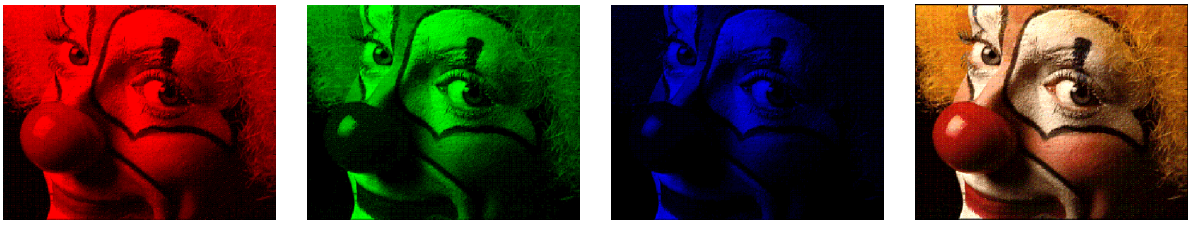
Hyperspectral Data Set

Standard Images

A standard color image is represented by three components, where each component corresponds to the intensity values of the red, green, and blue visible light electromagnetic frequency. Another way to think of this is that the image is essentially an n by m graph of points with three components each, forming a set of points in $\mathbb{R}^3$. An example of an image decomposed into its three components is shown in Figure 1. As can be seen, each "layer" represents the set of intensity values that correspond to one of the red, green, or blue frequencies. For example, in the red component, when the red layer is very "bright," such as the clown's nose, it indicates a high intensity value. Therefore, the actual image would have a high amount of red at the location. Areas of black indicate no contribution of the corresponding component to the image, such as the hair having a black blue component, meaning there is an absence of blue at the location. In addition, each layer is essentially a replica of the original image, the clown can be clearly seen.

Long Wavelength Infrared Hyperspectral Images

Like a standard image, a LWIR image consists of n by m pixels, with k components for each pixel, where k can be anywhere from 10 to 200 [1]. As in a standard image, a hyperspectral image can be thought of as a set of points in $\mathbb{R}^k$, or having k different spectral bands. Each band value corresponds to an intensity value for an electromagnetic frequency within the infrared spectrum, usually



Red Component Green Component Blue Component Clown Image

FIGURE 1. Decomposition of a clown image.

spaced equidistantly apart over a narrow area. Unfortunately, each layer of a LWIR image does not show any discernible image, the layers appear to be noise. An example of a spectral band is shown in Figure 2. Appearing quite random, each of the bands appear to be static, and do not allow for any "eyeball" inspection of the data.

The formation of these intensity values are made from a censor recording the data through what is known as a two layer model. Most objects on earth emit an electromagnetic wavelength spanning all frequencies, from the visible to invisible, all being transmitted. The censor picks up all of this information, over the specified frequencies. Now the challenge with this data lies in the fact that invisible frequencies of light can travel through objects, meaning that when a signal is received, its not just an object, but the background behind. In addition, the location of the camera could add interference, due to atmospheric signals that also emit radiation. There have been two proposed models for these situations, the Two-Layer Model or the Three-Layer Model [2], shown in Figure 3. The Two-Layer Model assumes no atmosphere and the Three-Layer Model takes into consideration atmospheric interference. Therefore, all signals received by the censor is actually a mixture of all the signals that inhabit the pixel. For simplicity, it is assumed that when a signal passes through an object, the signals are added

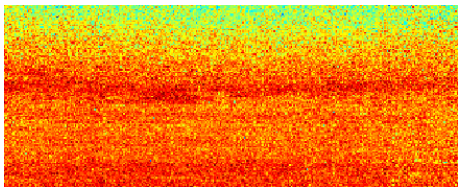


Image with color enhanced contrast

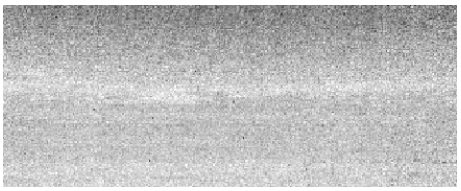


Image as standard grayscale

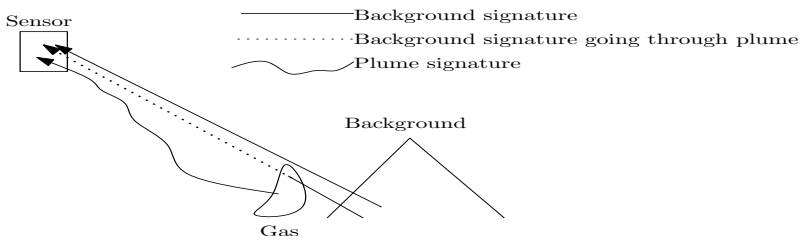
FIGURE 2. Sample hyperspectral image frame in one spectrum.

together to form a new signal, the dotted line in Figure 3, labeled as "Background signature going through plume." This is called the linear mixture model, because the signals are linearly combined. A simple example of this is shown in Figure 4, a pixel in a hyperspectral image is comprised of all the signals shown in the enlarged area, with the linear mixture model, all of these signals are summed together.

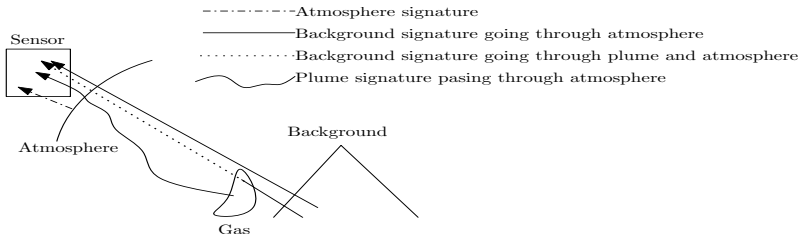
The second challenge that comes with hyperspectral images is the formatting of the information. There are five different measurements of light, shown in Table 1. Each of them have their uses, however in [2], it is mentioned that spectral emissivity works best when trying to locate gas plumes. The censors record the data in spectral radiance, therefore the data needs to be converted. However, this conversion from spectral radiance to spectral emissivity has problems that need to be addressed. The conversion requires a representative temperature over the entire image. Since most images do not have a uniform temperature, the data gets projected over one temperature and this results in values outside of the [0,1] range.

Dugway Proving Ground Data

The hyperspectral data set used for this project was provided by the Applied Physics Laboratory at Johns Hopkins University. It consists of a series of video sequences recording the release of chemical plumes taken at the Dugway



Two Layer Model



Three Layer Model

FIGURE 3. Two and three layer models for hyperspectral images.

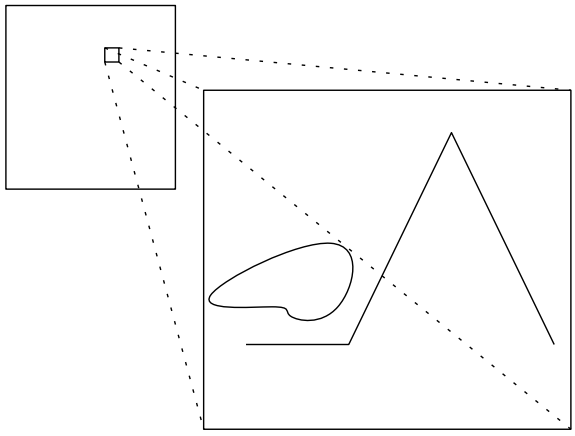


FIGURE 4. Example of a linear mixture model.

TABLE 1. Different Measurements of Light

Measurement	applies to form	definition
Irradiance	solid/gas	Amount of light reaching a surface
Exitance	solid/gas	Amount of light leaving a surface
Transmittance	gas	Ratio of exitance leaving over irradiance entering
Reflectance	solid	Ratio of exitance on surface over irradiance on surface
Absorption	solid	Ratio of light to another form energy over irradiance onto surface

Proving Ground. Videos were captured by three long wave infrared spectrometers (Romeo, Victory and Tango) placed at different locations to track the release of known chemicals. Each camera was approximately two kilometers away from each release at an elevation of about 1,300 feet. Figure 5 shows the arrangement of the three cameras. The sensors capture one hyperspectral image every five seconds consisting of measurements at wavelengths in the long wave infrared (LWIR) portion of the electromagnetic spectrum. Each layer in the spectral dimension depicts a particular frequency starting at 7,830 nm and ending with 11,700 nm with a channel spacing of 30 nm. The spatial dimension of each frame is 128×320 pixels, while spectral dimension is 129 channels. In other words, each image can be thought of as a 3D matrix, also known as a data cube, of size $128 \times 320 \times 129$.

Now onto the challenges of this specific data set and LWIR hyperspectral images in general. The camera records spectral radiance. As mentioned earlier, it needs to be converted to spectral emissivity. This process takes a single

temperature value and projects the rest of the data around this temperature, causing the spectral emissivity values that are supposed to be between [0,1], to extend to $\mathbb{R}$. Some ways to deal with this problem are to either scale all of the spectral signatures to be between [0,1] or to truncate the values into the range [0,1]. Looking at a plot of all the individual spectral emissivity values of the data in Figure 6, it would appear that scaling all of the data points would not work, since scaling the data would cause a majority of the points to be close to a single point, due to the extreme outliers. Therefore we opt for the alternative option of rounding off data points; however choosing 0 and 1 as roundoff points changes the distribution of the data too much. The goal of removing outliers is to maintain the data structure, not to change it, so choosing a different set of cutoff points might work better. In addition, if the assumption is made that there is negligible temperature deviation across the video sequence, then a case could be made that there is no need to round off the data points, as the temperature used to scale the data is the same throughout the video.

In addition to the outlying emissivity values, there are issues with the censors themselves. Certain cameras have scratches on them, dirt or grime, or pixels in which the censor no longer works, causing values to be incorrect. Locating these pixels in one frame is the same as finding them in all frames, since it is the same camera. The methods to locate these pixels are to find ones that either do not change throughout multiple frames, find pixels whose values are not a number, or to find pixels whose spectral sum is greater then three standard deviations from the surrounding neighborhood pixels. After locating such pixels, a method of replacement called 3D median filter is applied, different from the standard median filter. This preserves the integrity of signals while fixing the problem of unusual/outlier pixels, at the cost of some repeated pixels.

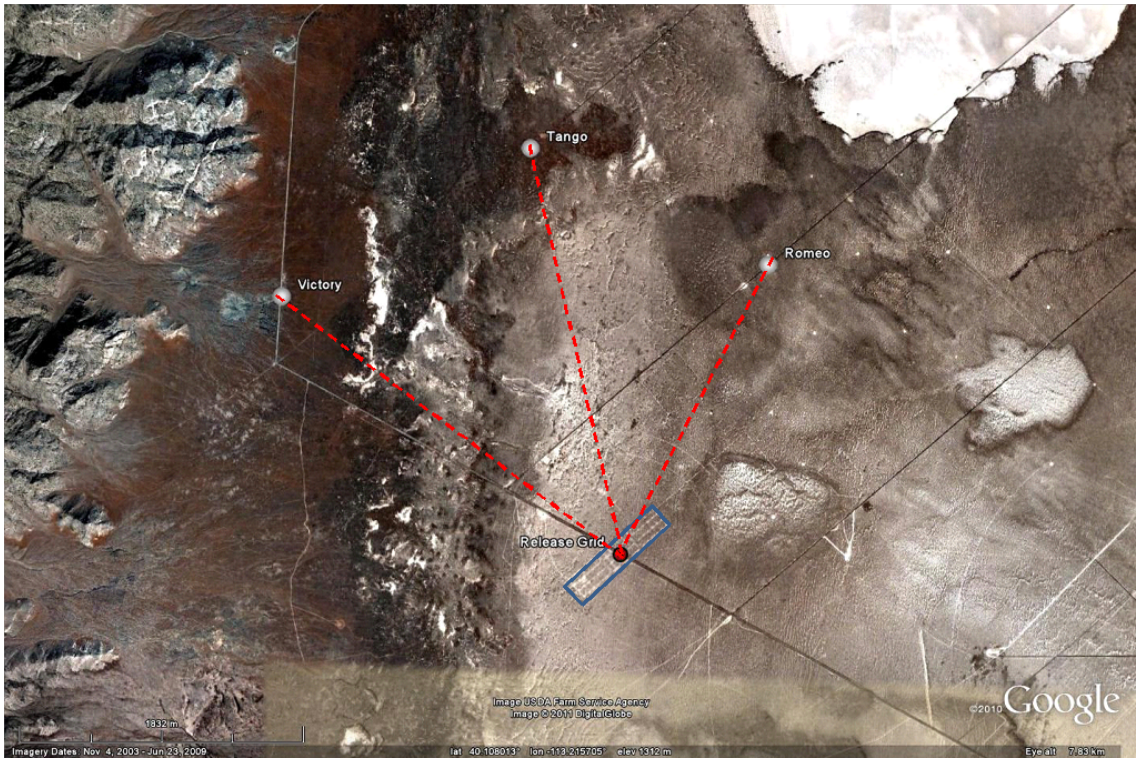
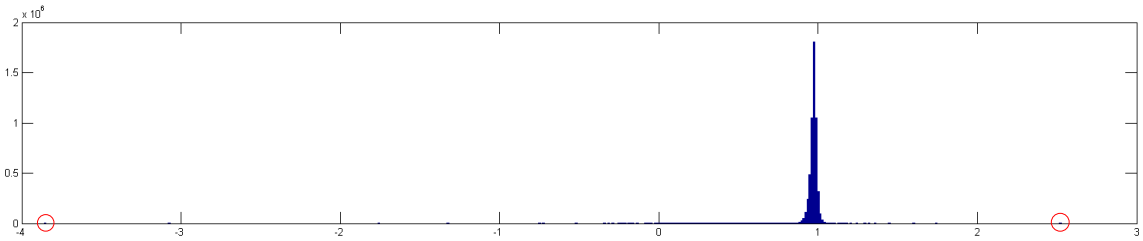
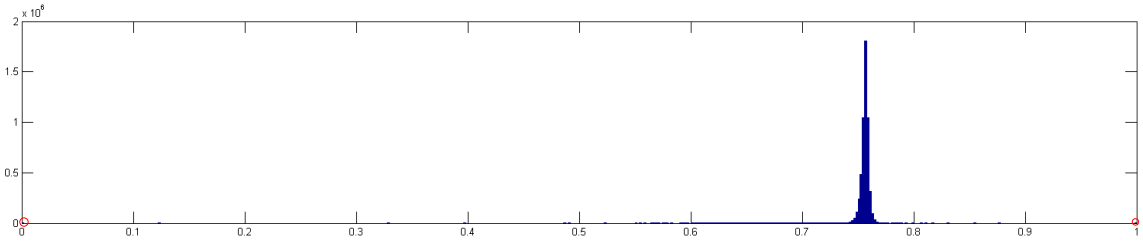


FIGURE 5. Image of camera locations.



Unscaled emissivity values for a sample frame



Scaled emissivity values for a sample frame

FIGURE 6. Scaled and unscaled sample data.

Distance Metrics

Introduction to Distances

All data has a specific best geometry representation, that is application dependent, allowing for an accurate calculation of distances between points of the data set. For example, for the points in $\mathbb{R}^3$space the best method of calculating distances is with the straight point to point distance, shown in Figure 7. When comparing vectors, the best representation is the angle between them, as shown in Figure 8. Finding a good method to calculate distances between points allows for an accurate comparison between points, which helps to determine closeness. Take for example Figure 9, the Euclidean distance is d while the cosine distance is 0. If A and B are treated as points, the cosine distance would give an incorrect measuring. On the other hand, treating A and B as vectors and utilizing the Euclidean distance gives an incorrect measurement. Thus a number of different distance metrics should be tested taking into account the time of computation as well as the accuracy of the results. Having the most accurate method of computing distances, but taking a long time to make one comparison does not sound reasonable. Taking into account for our large data set, a relatively fast method is preferred.

Distance Metrics for Hyperspectral Data

There are two main reasons for the lack of a best distance metric in differentiating the gas plume from background, the geometry of hyperspectral data is unknown and it has a high dimensionality. Therefore, any metric that can be extended into multiple dimensions can be considered. Table 2 gives an overview of four reasonable distance metrics for this data set.

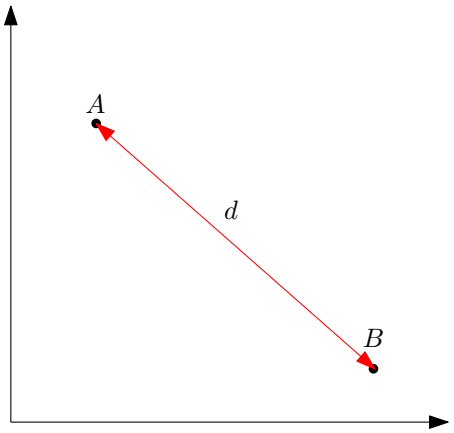


FIGURE 7. Euclidean distance between points.

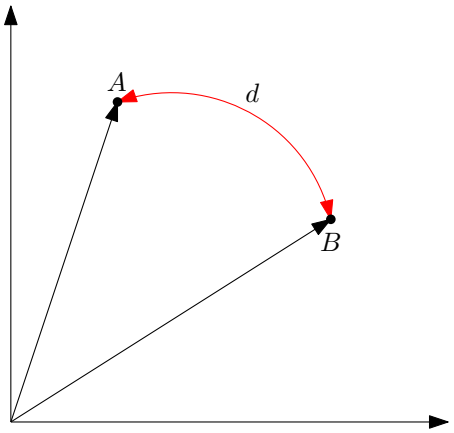


FIGURE 8. Cosine distance between vectors.

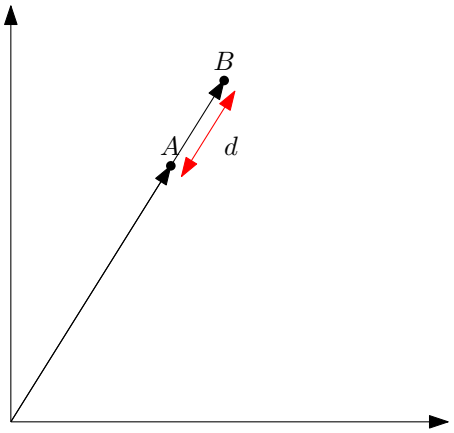


FIGURE 9. Bad distance example.

TABLE 2. Table of Different Distance Metrics

Euclidean Distance	Computes the straight line distance between points.
Cosine Distance	Also known as the spectral angle. Computes the angle formed between the point vectors. Invariant to scaling.
Spectral Gradient Angle	Computes the angle between the gradients of two points.
Hausdorff Distance	Computes the best distance between two spaces.

Euclidean Distance

The Euclidean distance between hyperspectral data points, $x = (x_1, x_2, \cdots, x_n)$ and $y = (y_1, y_2, \cdots, y_n)$, is defined as

$$d_{euc}(x, y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \cdots + (x_n - y_n)^2}.$$

The Euclidean distance is the straight line distance from point to point, and is considered the standard distance metric. It works best when data points do not have a specific pattern, such as those in space. It also scales easily as the dimension increases, having an extra two additions and a multiplication for each extra dimension.

Cosine Distance

The cosine distance between hyperspectral data points, $x = (x_1, x_2, \cdots, x_n)$ and $y = (y_1, y_2, \cdots, y_n)$, is defined as

$$d_{cos}(x, y) = \arccos\left(\frac{< x, y >}{||x||\ ||y||}\right).$$

The cosine metric, also known as spectral angle, computes the angle between the vectors formed from two points. This metric is invariant to scaling, which might be a problem with the data. A real problem, however, is the arc cosine function, which can be very time consuming over many iterations, therefore a modified version is used instead

$$d_{mcos}(x, y) = 1 - \frac{< x, y >}{||x||\ ||y||}.$$

It utilizes the fact that as long as both the arc cosine and the approximation are nondecreasing functions on the interval of possible values, the proper comparisons are able to be made.

Figure 10 shows the results of applying the cosine and modified cosine over the same domain of $[0, \pi]$. Notice that very little difference between the two metrics was observed. The actual difference between the two measurements is shown in the figure as d. Considering the size of the hyperspectral data set, a small decrease in computation time can be greatly compounded. For example, computing all of the distance between each set of pixels in an image of size 128 would result in over a billion arc cosine calls. Performing an experiment with 100 million points, the average time it takes to compute arc cosine is .046 seconds while the approximation takes .013 seconds. The approximation takes roughly a third of the time to compute than the arc cosine. This is a great saving on a large scale computation.

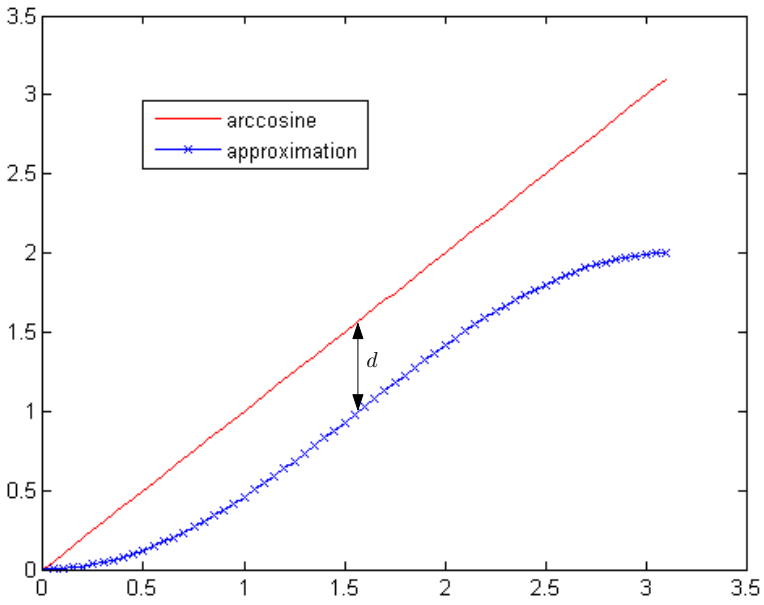


FIGURE 10. Comparison between the cosine and modified cosine distance metric.

CHAPTER 3

METHODS

Clustering

Clustering is defined as the act of combining groups of similar object together, into what are known as clusters. There are various methods of clustering, some utilize complex graph structures, while others use a simple distance comparison. In general, methods can be classified into two main categories, hard algorithms and soft (or fuzzy) algorithms. Hard algorithms result in each point being in a single cluster, for example breaking an image into discrete parts. An example of the results of a hard clustering algorithm can be seen in Figure 11. On the other hand, fuzzy clustering algorithms provide the ability for a point to be in multiple clusters, allowing for the possibility of multiple hard clustering results. An example can be seen in Figure 12, a simple scenic picture with clouds and mountain. There are two parts to the fuzzy clustering results. The first result separates the sky region from non-sky region, and the second splits the clouds from the rest. Take, for example, point A in the first result, which is a part of the ground. It is then a part of the non-cloud region in the second result.

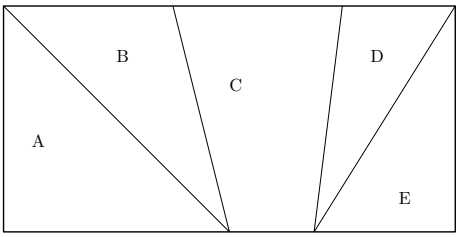
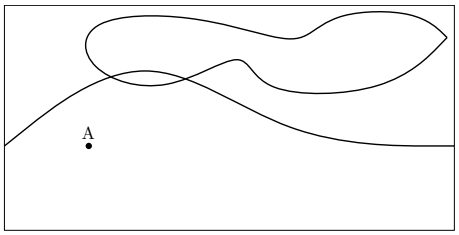
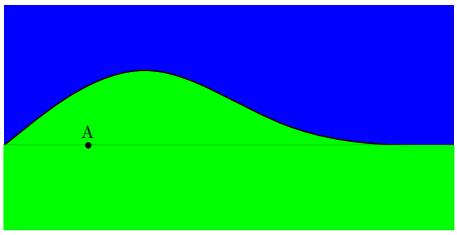
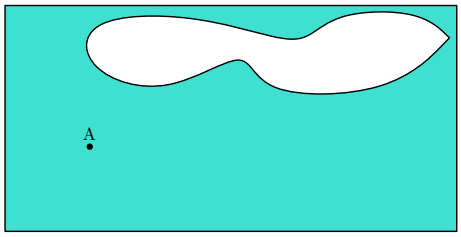


FIGURE 11. Sample hard clustering result.



Original image, with cloud and mountain





First result, isolating sky Second result, isolating cloud

FIGURE 12. Sample fuzzy clustering result.

In the case of hyperspectral images, due to the mixing model, it would be best to utilize a fuzzy clustering algorithm in order to find the presence of a specific signature in the result, e.g., all of the mountain signatures or all of gas signatures. On the other hand, the hard clustering algorithm will yield an image that does not require any additional processing. K-means is an example of a hard clustering algorithm and spectral clustering is an example of a fuzzy clustering algorithm. It is also possible to perform a hard clustering algorithm on fuzzy clustering results in order to either refine your fuzzy results or to simply find a hard clustering representation. For example, applying K-means onto spectral clustering results will give a hard clustering representation to the fuzzy clustering algorithm.

K-Means

Introduction

K-means clustering is an iterative two step classification process by first calculating centroids and then forming clusters around these centroids. The exact

algorithm for K-means is explained in Algorithm 1. This simple algorithm is incredibly fast and flexible, allowing for the testings of many different aspects of the data.

Algorithm 1 K-Means Algorithm

Input: data points, number of clusters (k), centroids (optional)
Output: centroids, centroid membership
if centroids are not initialized **then**
  Randomly select k points as centroids
end if
loop
  Calculate the distance from each centroid to each data point
  Assign each data point to the closest centroid
  Recalculate centroid: calculate mean within a cluster
  if centroids do not change and each data point maintains centroid classification **then**
    End loop
  else
    Continue loop
  end if
end loop

Simple Example

Figure 13 shows a sample result of an iteration of K-means. The first step is to randomly initialize K centroids, if an initialization is not given. The second step is to compute the distances between the centroids and all other points,

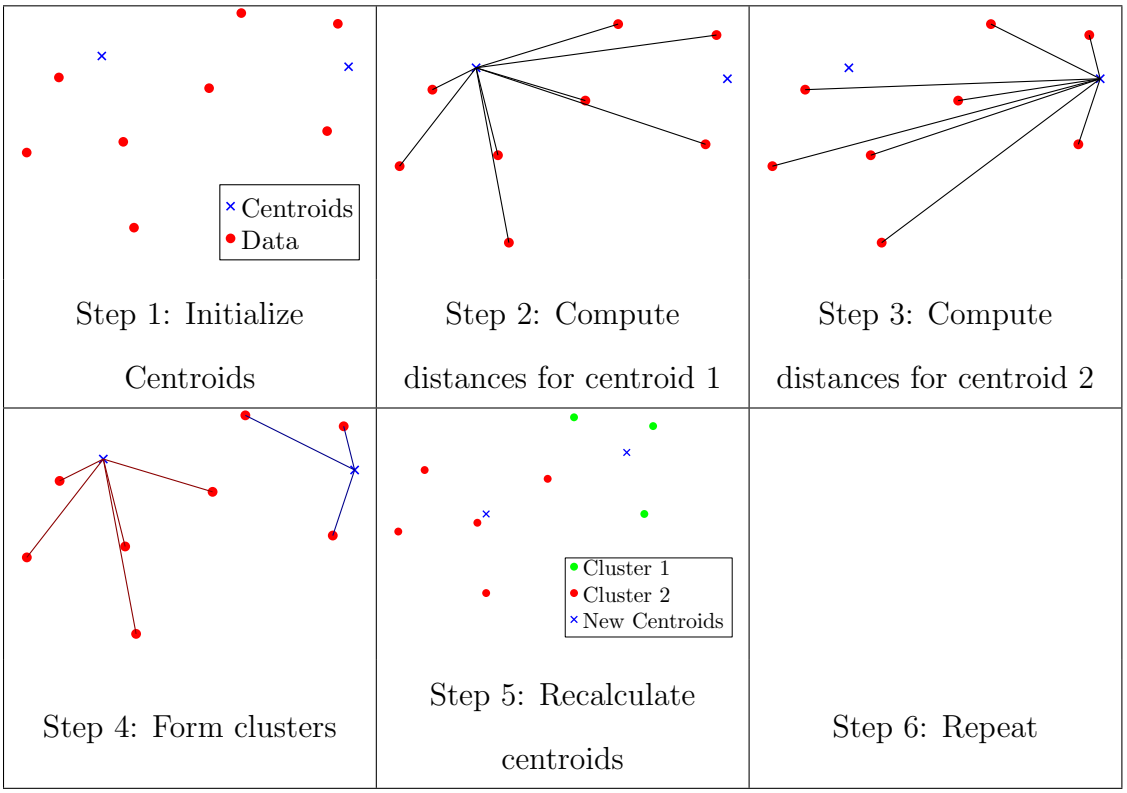


FIGURE 13. One sample iteration of K-means.

utilizing a distance metric mentioned before. After this point, we assign the points to the closest centroid, forming clusters around each centroid. After this step, you will recompute the centroid by calculating the mean of the points in a cluster. This process is repeated until a condition is met, usually either when the points do not change clusters or when the centroids do not change. The terminating conditions are not mutually exclusive, as there are instances where one is met but the other is not. However, the difference in resulting clusters from the different conditions is seemingly negligible.

Advantages

K-means is a very simple algorithm that can provide great insight into an unknown data set. The primary advantages are the simplicity and speed of the algorithm. The speed of the algorithm allows for a great amount of testing to be performed.

Distance metric – many different distance metrics can be tested since each instance of K-means can be finished in under a minute.

Approximate number of clusters – testing for many different K's allows for an approximation as to the actual number of clusters in an data set.

Viability of clustering – tests for whether clustering methods would provide reasonable results.

Easily interpreted results – results from other clustering algorithms can be much more complex, K-means provides a very easy way to interpret clustering.

Taking more complex algorithms into account, a single instance can take over an hour, with confusing results that give no insight into the data. It could also mean the algorithm was incorrectly implemented. K-means allows for a very efficient means of testing.

Weaknesses

There are several key weaknesses of K-means.

Initialization of centroids – different locations of centroids provide different results.

K centroids – having different number of centroids can provide vastly different results.

Very simple algorithm – more complex data sets are not easily clustered using K-means.

Initialization of centroids is key to proper utilization of K-means. Figure 14 shows an example of good and bad initializations of centroids. The bottom rows gives the ideal results of having three proper clusters. The bad result combines two clusters together and splits a third. If the data is somewhat familiar, the location of the centroids can be picked such that they are in the approximate final locations. On the other hand, if the data is unknown, then multiple instances of K-means with a large variety of initializations might be needed before the desired outcome can be reached. It is also possible that the correct initialization might never be found, due to the sheer size of certain data sets, making this a major weakness for K-means.

The correct number of centroids needs to be utilized in order for the correct formation of clusters. If the number of K is more or less than the correct number of clusters in the data, then K-means forces the number of clusters to be K. It will never realize if the results are correct, but it will always give results, which needs to be interpreted as good or bad results. Figure 15 shows what happens with incorrect K used in forming clusters. The data is supposed to consist of two large clusters, however giving different numbers of K will still get clustering results. If the data is unfamiliar, then the correct clustering will not be known unless the data is analyzed. For example, analysis in Figure 15 will show that the

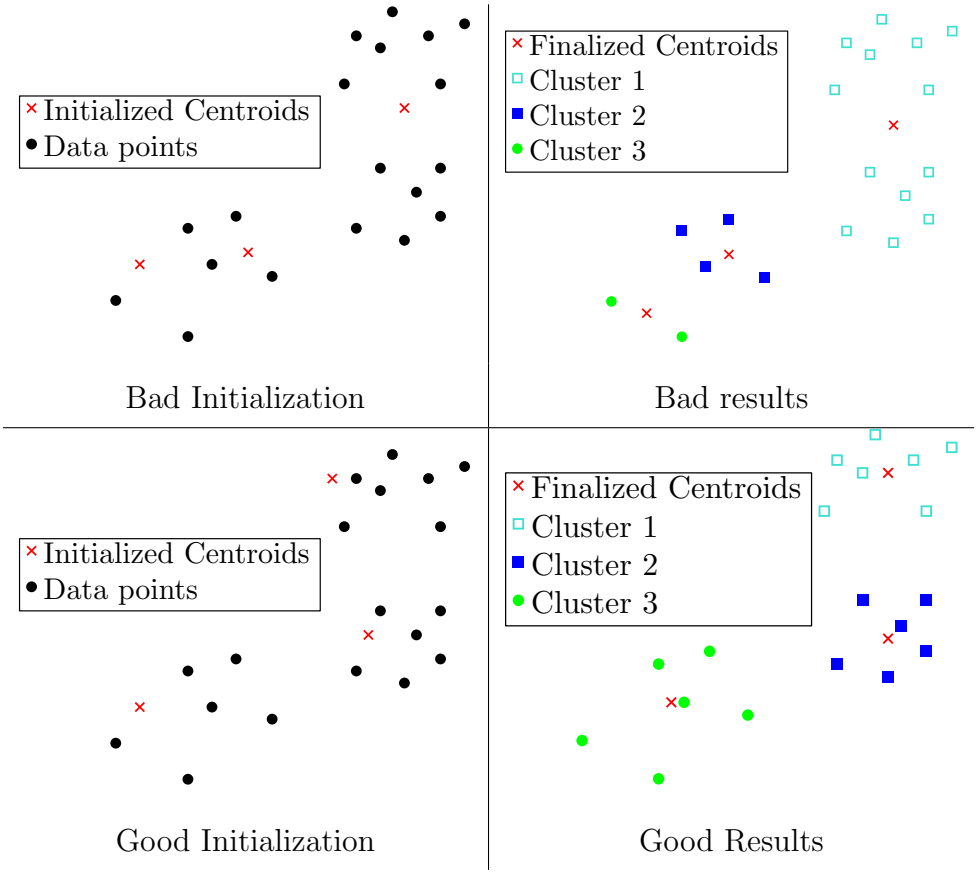


FIGURE 14. Example of good and bad initialization in K-means clustering.

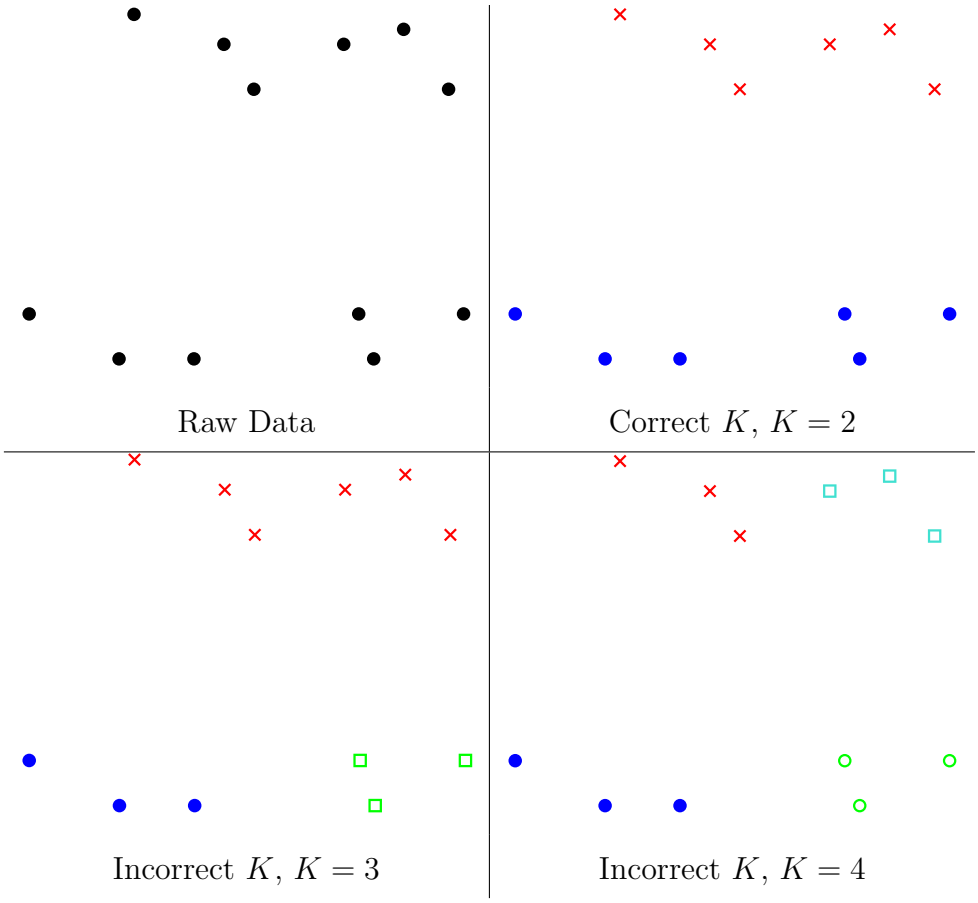


FIGURE 15. Example of good and bad K initialization.

results of $K = 2, 3, 4$ are all valid. The correct result will only be selected if it is known that there are only two clusters, without this prior knowledge, a best guess must be utilized to find the correct number of clusters.

K-means is a very light-weight algorithm that does not consider the graph structure of data, only the graphical distances between points using a chosen distance metric. For example, in the case of the Two Moons data set, K-means is unable to differentiate the two moons even in the best case scenario, as shown in Figure 16. The two moons are quite clearly seen, however K-means is unable to properly cluster this data set because it does not look at the graph structure.

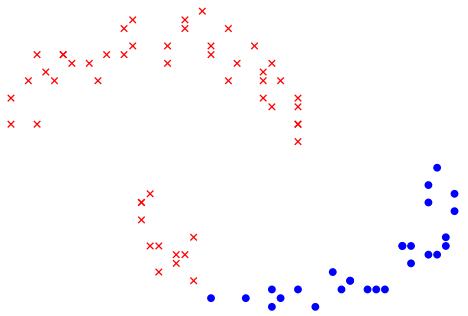


FIGURE 16. An example of K-means clustering Two Moons data set.

Modified K-means

The flexibility of K-means will allow for it to be modified in numerous ways. The primary modification we discuss here will be the ϵ nearest neighbors modification, shown in Algorithm 2. This modification essentially forms clusters of points that are incredibly close together and represents these clusters by the centroid, allowing for the data set to be reduced. The ϵ parameter needs to be adjusted in order to allow for a reduction in data but at the same time not removing critical information in the data. The result of this modification is known as the adaptive K-means, because there is no set K parameter.

Spectral Clustering

Terminology

In order to introduce spectral clustering, some terms must first be defined.

A *fully connected graph* is one where all of the nodes in the graph are connected to all other nodes.

An image can be treated as a fully connected graph, treating each pixel as a node, and forming connections between every single point.

Algorithm 2 Modified K-Means Algorithm

Input: data points

Output: centroids, centroid membership

loop

Select a random point as a starting centroid

Compute all distances between centroid and data points

Assign the data points that are within ϵ distance

if There are no points unassigned points **then**

End loop

else

Continue loop

end if

end loop

A *cut* is defined as removing a connection between two nodes in a connected graph.

Graph Laplacian is the nonlocal graph laplacian operator.

Introduction

Spectral Clustering [3] is a method that splits a data set into two parts using the minimum number of cuts, also known as performing mincut. It is shown in [4] that the solution to the mincut problem can be solved by with the eigenvectors of the graph Laplacian. The algorithm is shown in Algorithm 3.

The similarity between pixels represents how "similar" two points are to each other. A similarity of one means the two points are identical, while a similarity of zero means the two points have no relation to each other. This is done

Algorithm 3 Spectral Clustering [3] [5]

Input I is image data with n spectral components, I_j is a pixel in the image, $I_j = (I_{(j,1)}, I_{(j,2)}, ..., I_{(j,n)}) \in \mathbb{R}^n$

Form D, the matrix of distances between each pixel

Form S, the matrix of similarities between each pixel

Calculate d, the diagonal matrix of row sums. $d_{ii} = \sum S_i$, where S_i is the ith row of S

Form N, the normalized similarity matrix also known as the graph Laplacian, $N = d^{-0.5}Sd^{-0.5}$

Compute the largest eigenvalues and corresponding eigenvectors of matrix N

Analyze the eigenvalues and eigenvectors to form clusters

with the use of a similarity function, the most common one being the Gaussian similarity function:

$$S_{Gaussian}(x, y) = e^{-\frac{d(x,y)^2}{2\sigma^2}},$$

where x and y are two pixels and σ is a parameter that adjusts the similarity. When σ is small, the data needs to be closer to zero in order to have a higher similarity value. On the other hand, a very large σ results in all of the similarity values being at or very near one.

The resulting eigenvectors and eigenvalues provide the insight needed to perform the clustering. The eigenvalue provides the importance of the corresponding eigenvectors, 1 being the most important and 0 being the least important. The corresponding eigenvector provides a fuzzy clustering result for one specific component of the data. For example, it will segment background and non-background.

Let us illustrate this process through Figure 18, where a connected graph is

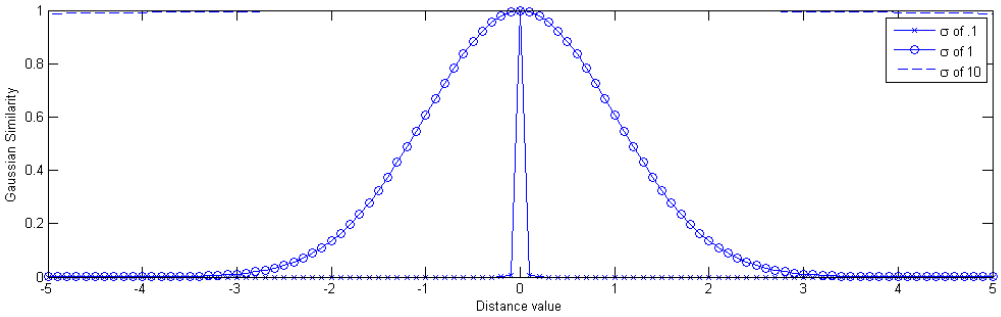


FIGURE 17. Three different σ's.

shown. The goal is to separate this graph into two components that are no longer connected by cutting the minimum number of connections. The graph Laplacian is shown in Table 3. The resulting eigenvectors and eigenvalues are shown in Table 4. The next step would be to interpret the resulting eigenfunctions. The first eigenpair is ignored, as it gives general information about the data and no clustering information. Plotting the second eigenvector gives Figure 19. The goal of spectral clustering is to find eigenvectors that split the data into two or three groups. This is done by assigning a value to each point, in essence it can be thought of as projecting the data onto 1-dimension. With n eigenvectors combined being considered as an n-dimension projection of the data. Looking at Figure 19, there is a clear point of separation between -.1 and .5, which means that the data can be thought of as having two distinct groups, one composed of points above the separation line and the other comprised of points below the separation line. Figure 19 shows the graph with a line separating the two regions, thereby grouping points A and B together, and with C to F as the second group. Figure 20 shows the next eigenvector along with a couple of possible line of separation, this eigenvector does not have a clear line of separation. To cluster using these two eigenvectors, an application of K-means with $k = 2$ to 3 is used, or any other hard clustering

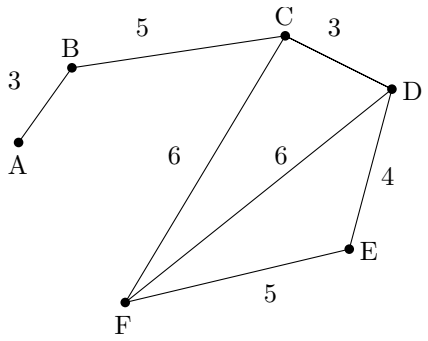


FIGURE 18. Sample connected graph.

TABLE 3. An Example of a Graph Laplacian

	A	B	C	D	E	F
A	0.6370	0.3409	0	0	0	0
B	0.3409	0.5620	0.1145	0	0	0
C	0	0.1145	0.5306	0.2904	0	0.0644
D	0	0	0.2904	0.4895	0.2049	0.0619
E	0	0	0	0.2049	0.6339	0.1400
F	0	0	0.0644	0.0619	0.1400	0.7040

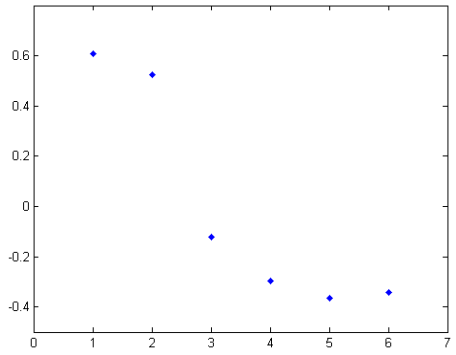
algorithm. The data points are a subset of $\mathbb{R}^2$, the first coordinate is the value in second eigenvector and the second coordinate is the value in the third eigenvector. Figure 21 shows the new ordered pairs as well as their $\mathbb{R}^2$representation. From here, K-means is applied, giving the clusters in Figure 22. It is important to note that unlike Figure 20, the results after K-means is in three clusters instead of two, (AB)(CD)(EF). Spectral clustering tries to cut the longest distance pieces, because those are furthest apart and therefore dissimilar.

Nyström Extension

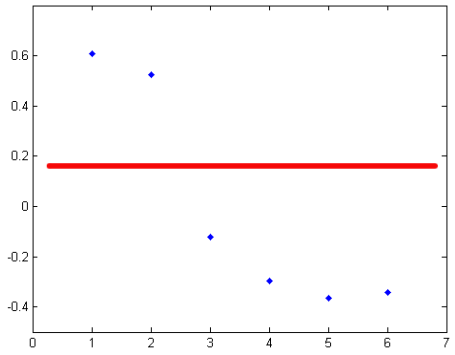
The Nyström Extension is a fast approximation to the eigenvectors and

TABLE 4. Eigenpairs Computed From Graph Laplacian in Table 3

Eigenvalue	Eigenvector					
1	0.3909	0.4161	0.4283	0.4459	0.3918	0.3718
0.9309	0.6081	0.5243	-0.1213	-0.2981	-0.3669	-0.3422
0.6638	-0.1924	-0.0151	0.5597	0.4068	-0.2509	-0.6489
0.5548	-0.1233	0.0298	0.3655	-0.1063	-0.7220	0.56371
0.2691	0.6068	-0.6549	-0.1312	0.3678	-0.2213	0.0384
0.1384	0.2387	-0.3492	0.5812	-0.6327	0.2804	-0.0664



Plot of second eigenvector



Plot of second eigenvector with separation line

FIGURE 19. Plot of second eigenvector demonstrating a line of separation.

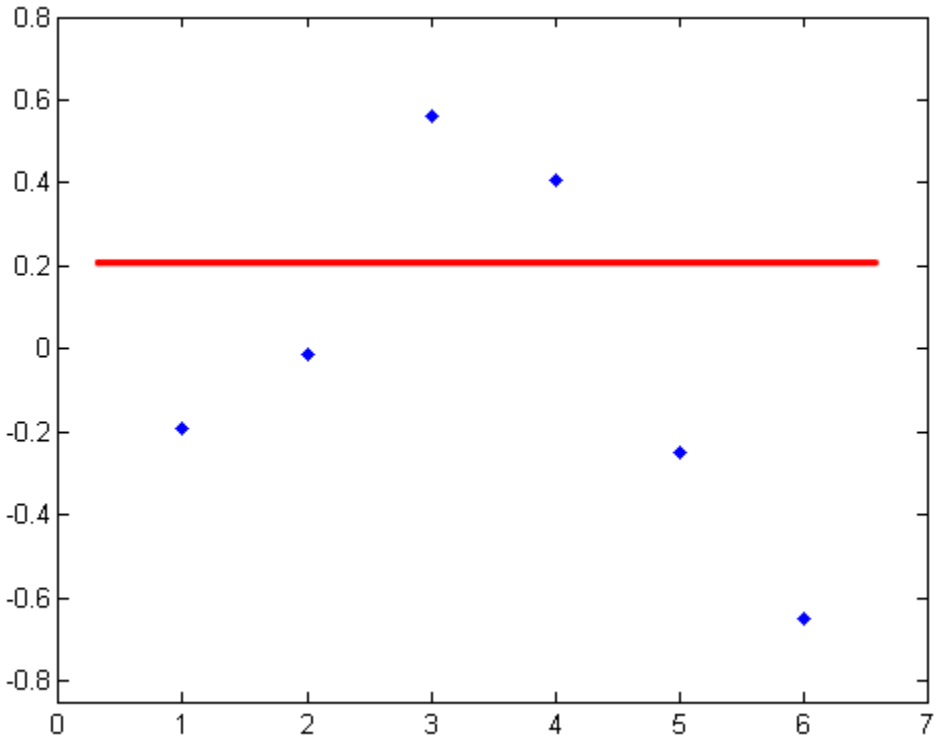


FIGURE 20. Plot of third eigenvector with separation line.

Point	X	Y
A’	0.6081	-0.1924
B’	0.5243	-0.0151
C’	-0.1213	0.5597
D’	-0.2981	0.4068
E’	-0.3669	-0.2509
F’	-0.3422	-0.6489

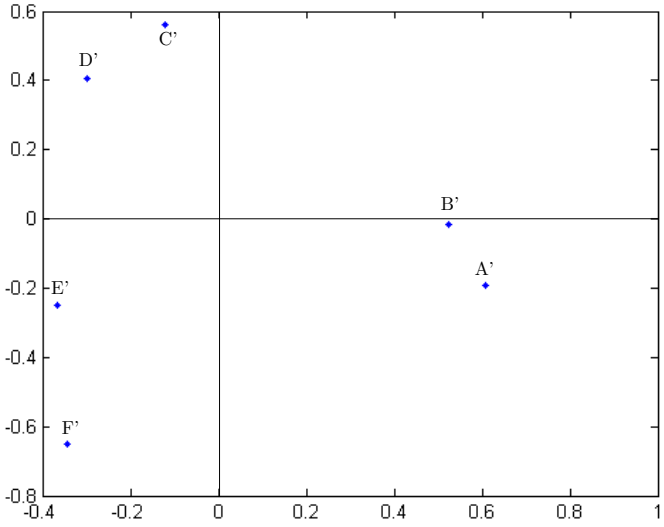


FIGURE 21. Showing data points in $\mathbb{R}^2$, along with corresponding coordinates.

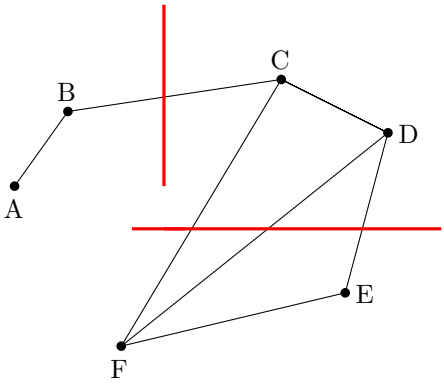


FIGURE 22. Results after K-means.

eigenvalues of the graph Laplacian or the results of spectral clustering. The steps are given in Algorithm 4. A more in-depth discussion can be read in [**?**]. The Nyström Extension was investigated because computation of the graph Laplacian in spectral clustering is very time consuming in large data sets. In the hyperspectral case it is approximately forty thousand points in $\mathbb{R}^{129}$. This means the computation time for the graph Laplacian as well as the eigenpairs are quite time consuming. The algorithm works by utilizing a sample set and matrix completion in order to approximate the graph Laplacian, afterward utilizing eigenfunction properties to compute the eigenpairs. Not only is it faster, the memory usage is also smaller allowing for more complex data sets to be used. The difference in computation time can be as much as ten times. In fact, a spectral clustering on multiple frames can be made feasible with the Nyström Extension [7] [8]. The results of Nyström Extension are interpreted the same way as the spectral clustering results.

Algorithm 4 Nyström Extension [**?**]

Input: Image data I, k

Randomly select k data points to form A, rest of data forms B

Compute distances amongst data in A, D_A

Compute distances between data in A and B, D_B

Approximate the distances amongst data in B, D_C, with $D_C = D_B^T D_A^{-1} D_B$

Compute the row sum of the matrix $d = \begin{bmatrix} D_A & D_B \\ D_B^T & D_C \end{bmatrix}$, where d_i is the ith row of d

Normalize Each element of D_A and D_B, $\overline{D_A}$ and $\overline{D_B}$, where $\overline{D_A}(i,j) = \frac{D_A(i,j)}{d_i d_j}$

$Q = \overline{D_A} + \overline{D_A}^{-.5} * \overline{D_B} * \overline{D_B}^T * \overline{D_A}^{-.5}$

Find the singular value decomposition of Q, $Q = USV^T$

Compute $W = \begin{bmatrix} \overline{D_A} \\ \overline{D_B}^T \end{bmatrix} \overline{D_A}^{-.5} US^{-.5}$

Compute the eigenvector approximation $Eig = \frac{W_i}{W_{1i}(1-S_{ii})^{.5}}$, where W_i is the ith row of W.

CHAPTER 4

EXPERIMENTAL RESULTS AND DISCUSSION

Experiments

Data Set

The hyperspectral data set that I worked with was provided by the Johns Hopkins Applied Physics Laboratory (JHAPL) [2]. The data was available in the form of spectral radiance long wavelength hyperspectral video sequence depicting the release of three different gas plumes at the same location, with three cameras as shown in Figure 5. The first step was to convert the data format from spectral radiance into spectral emissivity, utilizing code given from JHAPL. As previously mentioned, this data comes with scaling issues as well as sensor issues. To fix these problems, a 3D median filtering process was implemented to detect and replace problem pixels. This process is described in Algorithm 5. The alternative to the pixel replacement strategy is to perform a spectral layer by layer median filter, however this poses the problem of creating potentially new signatures and introducing them into the image, which is not ideal. An example of this process is shown in Figure 23, the final step is replacing the center pixel by the median of the surrounding pixels. Figure 24 shows the summation of the pixels down the spectral component for a sample figure in the Romeo and Victory cameras, the darker the pixel the higher the summation. Note the slash pattern in the Romeo camera as well as the splattering of dark points in both cameras, both of these are indications of problem pixels that need to be replaced. Identifying these problem pixels in one image allows for all of the images from the same camera to be located as well,

since it is the same camera with the same problems. The dark points in Figure 24 are essentially the abnormal points that need to be filtered. An image after it has been filtered is free of outlier points, as shown in Figure 25. For comparison, Figure 6 shows the unscaled, and unfiltered histogram.

When working with new data sets, it is also important to look at all possible avenues of information, for example the summation of hyperspectral data down the spectral components gives two important pieces of information about the Romeo data set. The first is the straight slash in image, it would appear that there is an issue with the camera, either a slash or some grime on the lens. This means that all of the pixels at the location across the video sequence would need to be corrected, e.g., by filtering. Another point of interest is the mountain range, circled in Figure 26. The elevated signals means that the particular mountain has a very strong signal that could potentially interfere with signals. These two facts lead to the belief that the Romeo data set may not be the best for testing of algorithms.

Algorithm 5 3-d median filter

Input I is image data with n spectral components, where I_{ij} is a pixel in the image and $I_{ij} = (I_{(ij,1)}, I_{(ij,2)}, ..., I_{(ij,n)})$, size of neighborhood N

Compute the sum of the spectral components in each pixel, $\sum_k I_{ij,k}$

Identify locations where the sum is outside three standard deviations of the neighboring pixels, $N \times N$

Find the median sum value pixel amongst neighboring pixels

Replace the pixel value with the spectral information from the median pixel

K-means

The K-means algorithm can run on the hyperspectral data set without

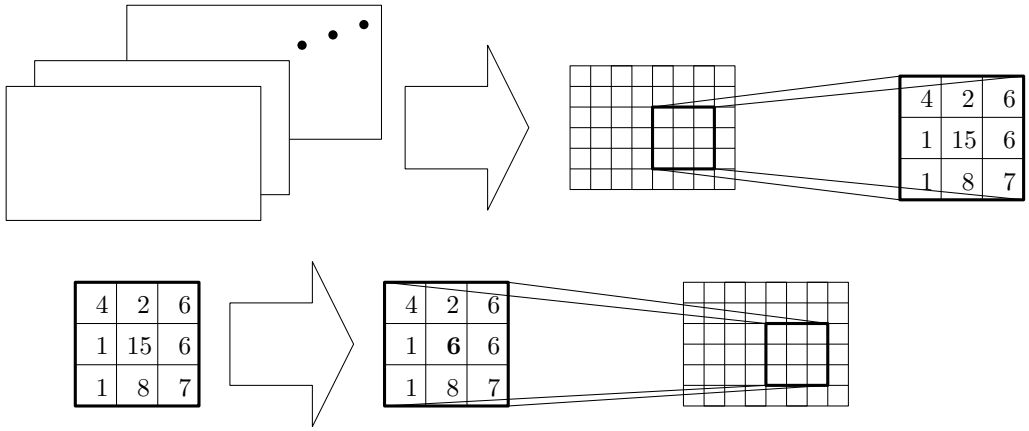
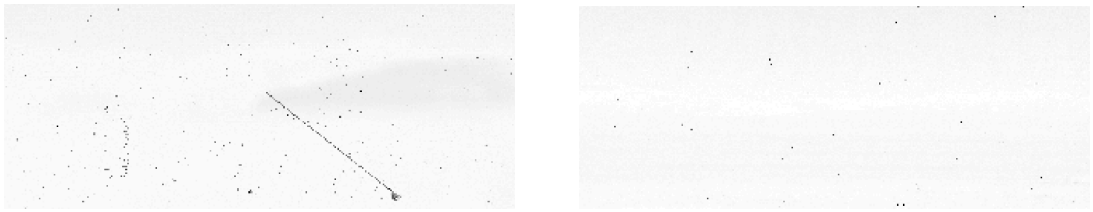


FIGURE 23. Example of 3-d median filter.



Spectral summation for a Romeo frame Spectral summation for a Victory frame

FIGURE 24. Romeo and Victory summation images.

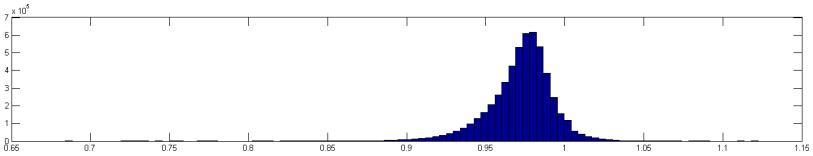


FIGURE 25. Histogram of 3D median filtered data.

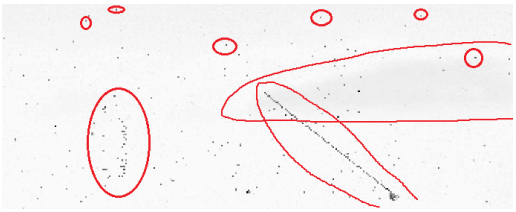


FIGURE 26. Mountain circled in solid line.

modification. The results of K-means were very helpful, as it was the first glimpse into the data that I had ever seen, giving a fairly recognizable picture compared to the raw data, as seen in Figure 2. Figure 27 shows four different frames with K-means applied, $k = 7$, one frame of only background from Victory and Romeo and one frame with gas plume from Victory and Romeo. Between the four sets of clustering results, we are able to gleam some very useful pieces of information. It is important to note that there was no ground truth provided with the data, therefore the following statements are based upon knowledge of the area and information provided to us.

Comparison between the results of the Romeo and Victory data sets

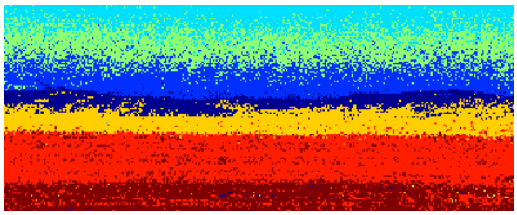
The mountains are visible in both data sets, and a clear separation between mountain, sky, and foreground. With this image, you can start to see the same features in Figure 24. The individual features are separated into their components, as shown in Figure 28.

There is a predominant mountain in the Romeo data set that seems to disrupt the gas plume, it should be much larger than shown in the image. Figure 29 shows the different components from the K-means results. Mount 1 is the disruptive mountain signal, extending into the foreground clusters. The rest of the mountain range is the Mount 2 component.

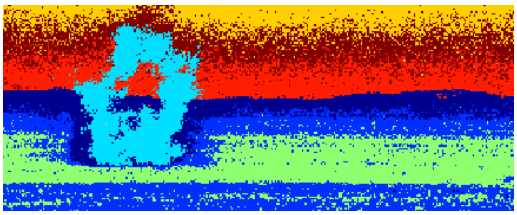
Different levels of atmosphere can be seen in both sets of data. The different layers in the air can be thought of as different atmospheric levels.

Comparison between the two different distance metrics

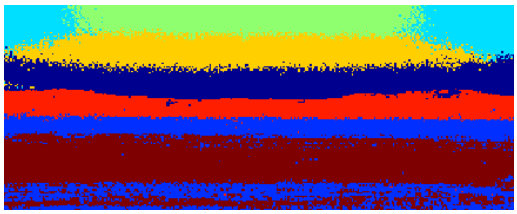
Cosine distance metric allows for a much clearer separation along the



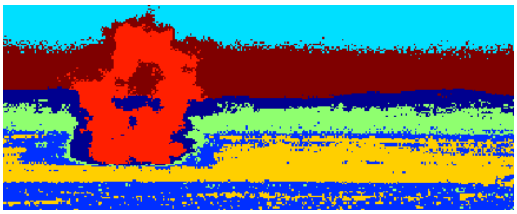
Background Victory frame with Euclidean



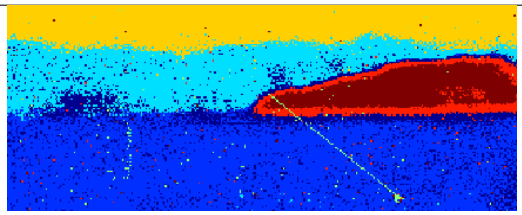
Gas plume Victory frame with Euclidean



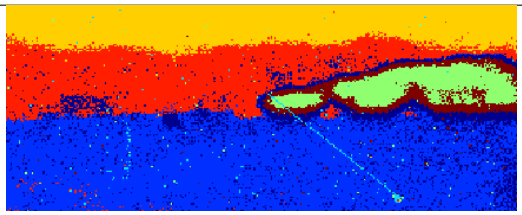
Background Victory frame with cosine



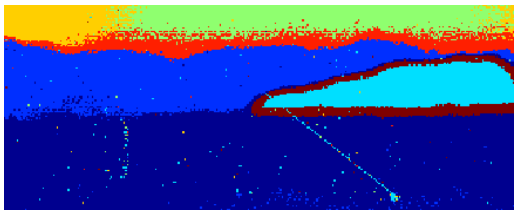
Gas plume Victory frame with cosine



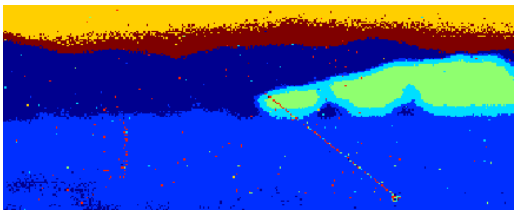
Background Romeo frame with Euclidean



Gas plume Romeo frame with Euclidean



Background Romeo frame with cosine

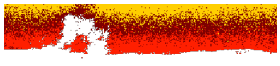


Gas plume Romeo frame with cosine

FIGURE 27. Examples of K-means result with different distance metrics.

Plume component

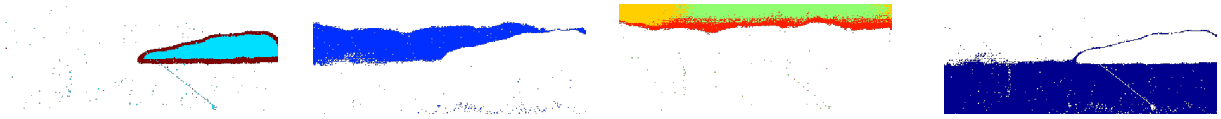
Mount component



Sky component

Foreground component

FIGURE 28. Components from K-means Victory results on a frame.



Mount 1 component Mount 2 component Sky component Foreground component

FIGURE 29. Different components from K-means Romeo results.

Euclidean mountain border Cosine mountain border

FIGURE 30. Mountain border with the Euclidean and cosine metrics.

borders of the different aspects of the image. Figure 30 shows the border between the mountain cluster and the sky cluster.

Cosine distance metric is better able to capture the gas plume.

It is important to note the strengths and weaknesses of the K-means algorithm.

Strengths of the algorithm

Separation of different environmental elements.

More information than previously anticipated, such as different layers of atmosphere as well as two different foreground components.

An actual separation of gas plume, a clear boundary of the gas can be seen.

The cosine distance metric works better at separating the boundaries between different environmental elements.

Very fast algorithm that does not require any special treatment.

Weaknesses of the algorithm

Different initializations of the data can result in poor clusterings.

Results can vary depending on the K selected. In the examples of Figure 27, the gas plume is more or less in one cluster. However this is not the case with a larger K value.

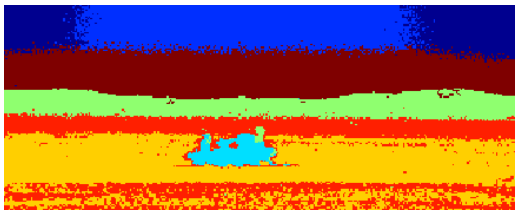
Sometimes the gas plume cannot be isolated, due to the randomness of the algorithm.

The plume does not seem to be entirely clustered, the areas around the plume should also be considered as gas, however there is a thin layer that is grouped with mountain.

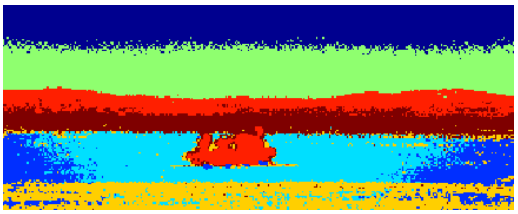
The various weaknesses are displayed in Figure 31. The first row shows the weaknesses of improper initialization of centroids, proper initialization results in the plume being in its own cluster with only a part of it not being clustered with the rest. The bad initialization results in the plume being in the same cluster as one of the mountain layers. The next two rows show what happens with the wrong K initialization, the good example shows a clean two portions of the plume, both in their own clusters. The next picture shows the plume in multiple portions, but the rest of the background portions are fragmented as well, making it difficult to piece the plume together. The last two show the results of having a K value too small, the image gets clustered into large pieces that are not plume. Ideally a $K = 2$ would result in the plume being in one cluster and everything else in another, however that is not the case. This means that the plume signature is closely related to some of the background signatures.

Spectral Clustering

The clustering results from spectral clustering are open to interpretation, unlike K-means, due to the fuzzy clusters. The algorithm must also be modified from the standard spectral clustering algorithm due to the size of the graph



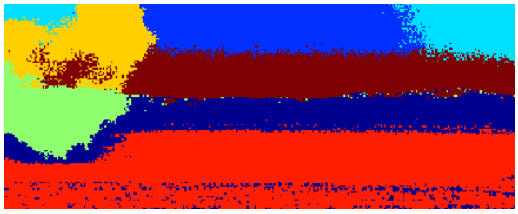
Proper initialization, plume isolated



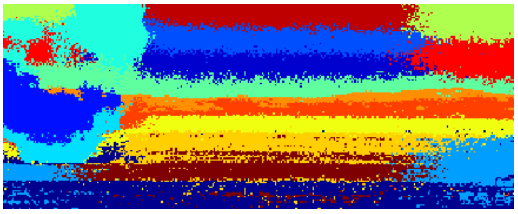
Improper initialization

Only plume with proper initialization

Cluster with gas plume, improper initialization

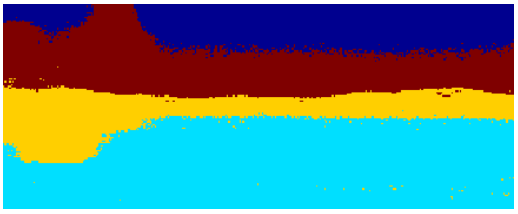


Results of an appropriate K



Results of a K too large

Results of a small K value



Another results of a small K value

FIGURE 31. Challenges arising from implementing K-means.

Laplacian matrices that were to be computed. Therefore, instead of computing the full graph Laplacian, a subset of it must be computed. The method chosen for this procedure is to select only the top K distances, a method known as K-nearest neighbors. This allows for a construction of a sparse matrix which uses less memory, the computation of the graph Laplacian is feasible. The basis for this modification can be thought of as a change to the distance function, setting distance above a threshold to a large n such that the similarity function, $f_s(x)$, would result in $f_s(n) = 0$. Simply put, it ignores connections that are too large, since the effect of these values is negligible. Therefore the total number of variables that need to be tested for spectral clustering is two, the σ from the similarity function and K for the K-nearest neighbors modification. So the goal would be to find the σ parameter that best segments the data, as well as the smallest value of K that gives good results. The larger the K the more processing time it would take to compute the results, therefore the smaller the better. Displayed in Figure 32 is results of spectral clustering on a frame with gas plume. The images are color contrast enhanced, allowing for a better visualization of the different aspects in the image. Figure 33 shows the eigenvectors for a frame that has a gas plume.

Figure 34 shows an example of the analysis that is done on the eigenvectors, specifically the ones shown in Figure 32. The first column is the image of the color contrast enhance eigenvector, before any analysis. The second column the sorted histogram of the eigenvector values, as well as showing the separation of the data into clusters. There are examples of two clusters as well as three clusters that can form from the resulting eigenvectors. The points of separation are fairly subjective, however a sharp turn in the data points is usually an indication of when a separate cluster appears. The first row in Figure 34 is separating the sky from the mountain and foreground. The second row is isolating

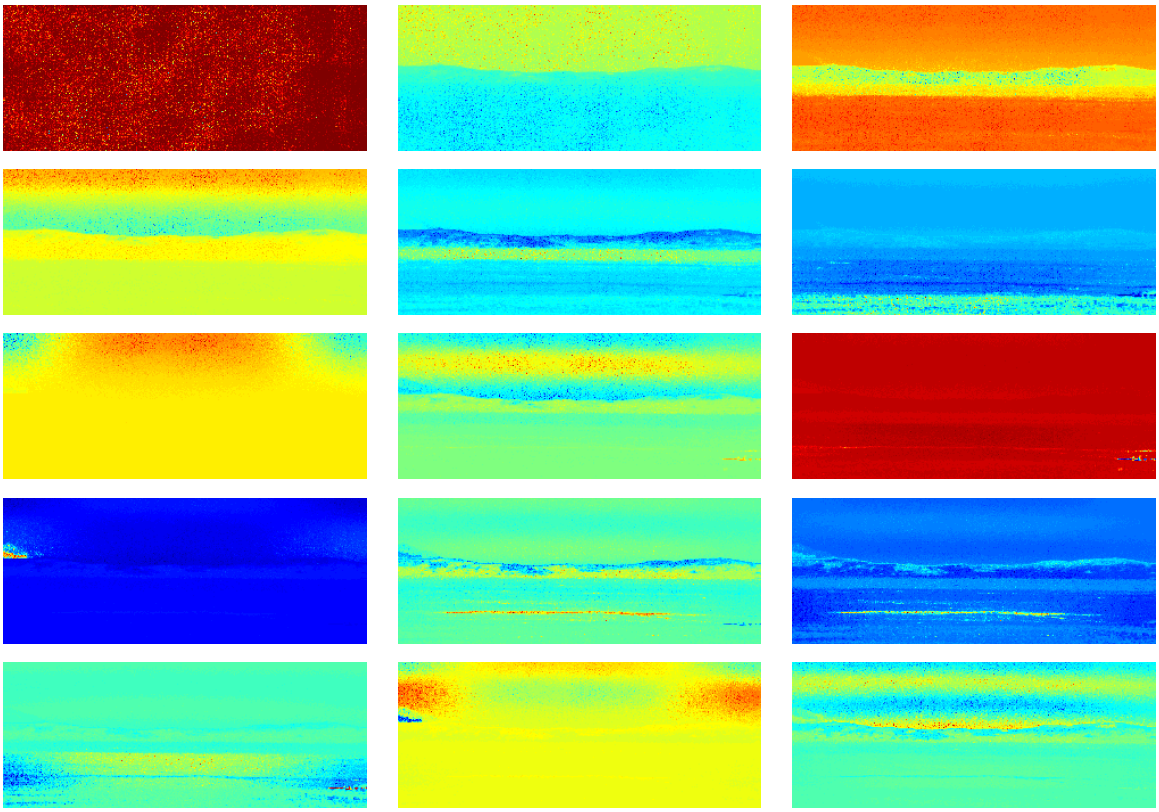


FIGURE 32. Spectral clustering results from a background frame.

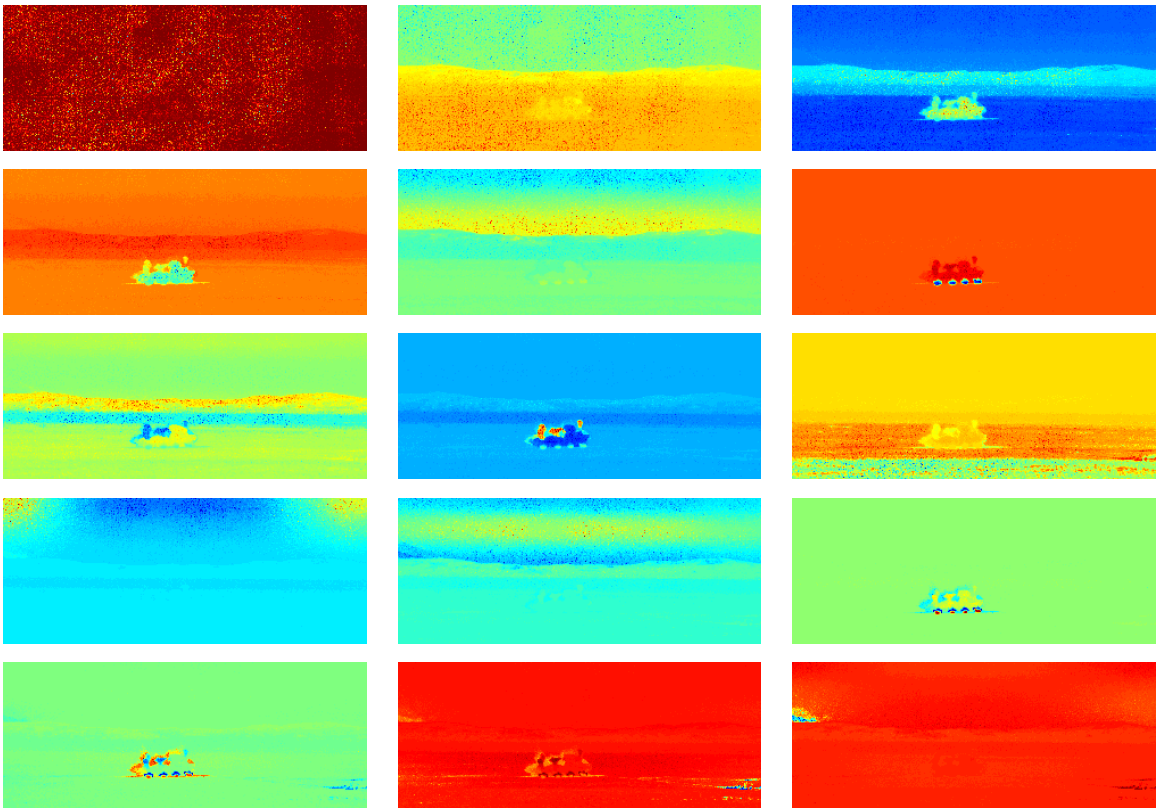


FIGURE 33. Spectral clustering results from a gas plume frame.

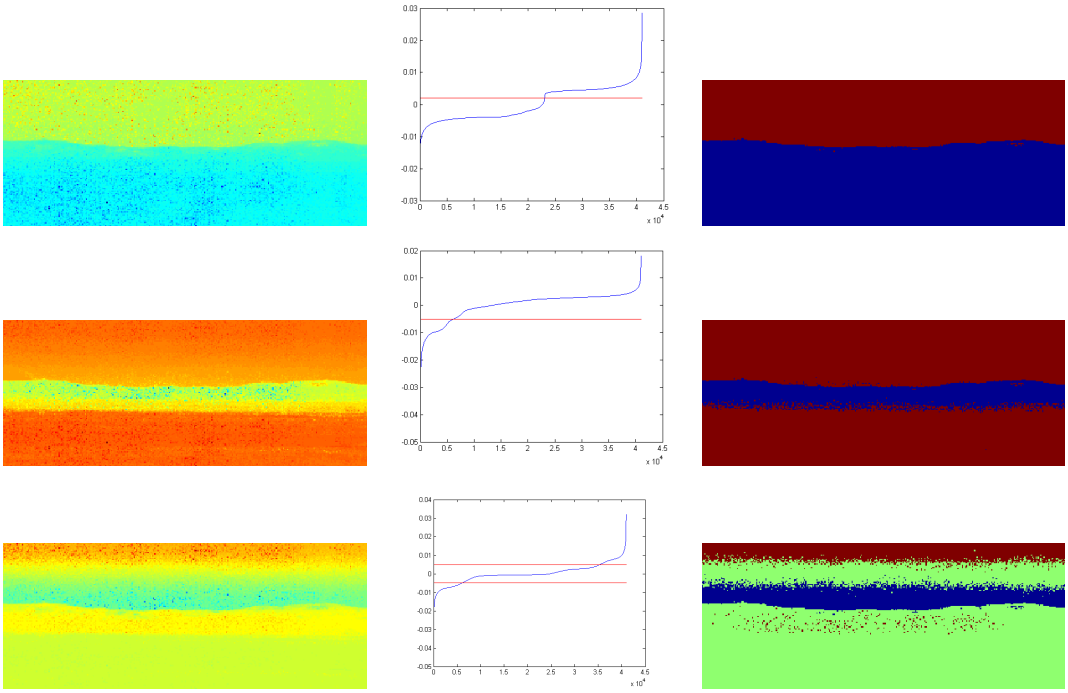


FIGURE 34. Analyzing the first three spectral clustering eigenvectors.

the mountain range from the rest of the image. The third eigenvector separates two distinct areas in the sky, potentially different layers of atmosphere.

Moving onto gas plume frames, Figure 35 shows some of the more interesting eigenvectors. The first row shows the isolated gas plume, the outline is also very "neat", and distinctly formed. However the second row shows that there is four circular regions that are of interest. Given how the gas plumes were explosively released into the air [2], it can be assumed that these four circular regions were the points of detonation. The eighth eigenvector in Figure 33 also seems to separate different areas of the gas plume from the rest, however in this case it is harder to make a guess about the cause of separation. This shows the strength of spectral clustering, not only being able to isolate the gas plume, but able to pinpoint areas of interest in this plume [9].

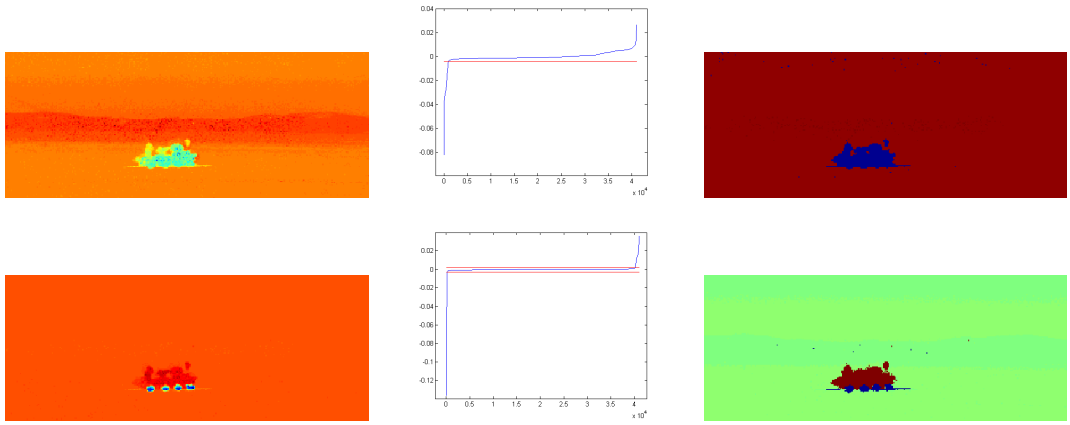


FIGURE 35. Two spectral clustering eigenvectors of gas plume.

Strengths of spectral clustering

Gives very accurate results and is able to locate many of the different features in hyperspectral images.

Forms an accurate rendering of the gas plume.

Is able to locate areas of gas plume that are slightly different from the rest.

Weaknesses of spectral clustering

As with other fuzzy clustering methods, the results need to be analyzed in order to determine what information is actually yielded.

Analysis and determining the eigenvector that is desired cannot be done automatically, at this time.

Very time consuming, upwards of 30 minutes for one frame and 50 eigenpairs.

Nyström Extension

The Nyström Extension [**?**] is a fast approximation to the eigenpairs of the graph Laplacian. The runtime for one frame is around a minute, compared to the 30+ minutes that it takes for spectral clustering. The results are also comparable,

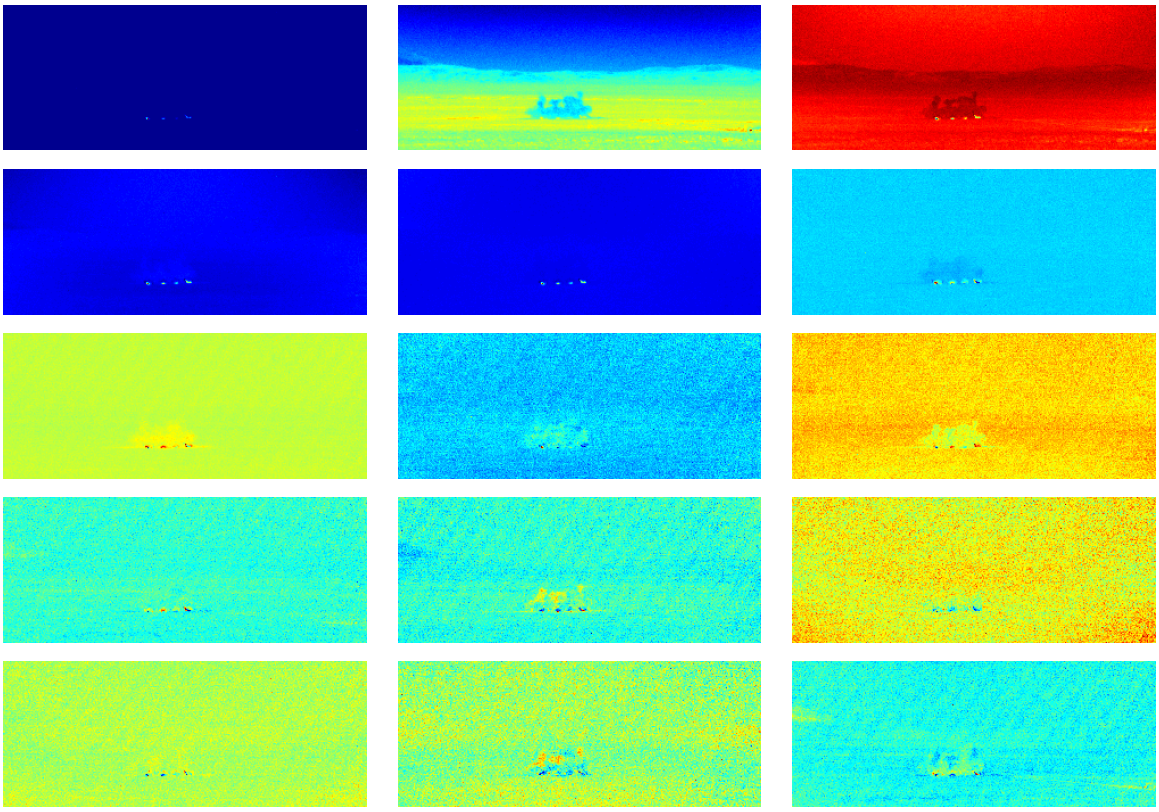


FIGURE 36. Nyström extension results from a gas plume frame.

for example compare Figure 33 to those seen in Figure 36. The four circles can be seen, as well as the gas plume itself, however since it is an approximation, it does not isolate the gas plume quite as well as in spectral clustering. In addition, the eigenvectors degrade to noise a lot faster than those of spectral clustering, again this is caused by the fact that it is an approximation. However, the speed at which the eigenpairs are calculated with Nyström extension, allow for the inaccuracies to be overlooked.

The short speed and limited memory requirements allow for usage of multiple hyperspectral video frames. Using more than one frame allow for connections to be made across time or even across a completely different video

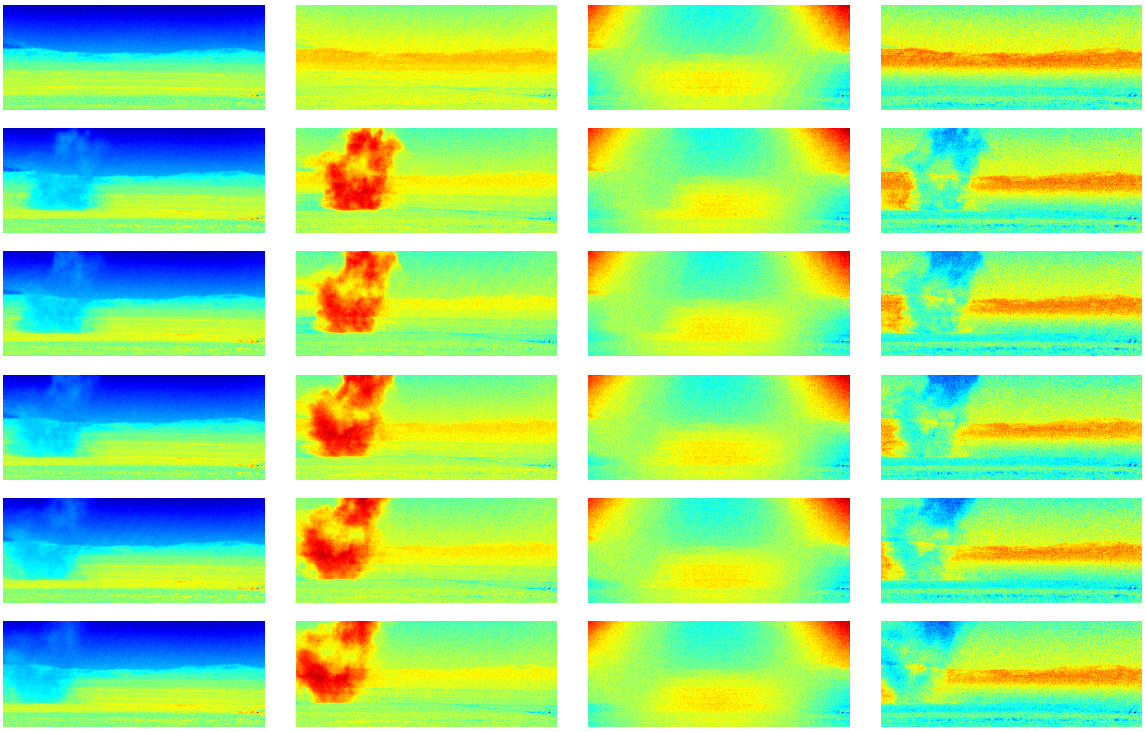


FIGURE 37. First four eigenvectors of multiframe Nyström extension.

frame. For example, Figure 37 show the first four eigenvectors of the graph Laplacian, using five sequential frames and one background frame. This multiframe Nyström extension allows for graph connections to be made across these video frames. Each column in Figure 37 represents one eigenvector, just reformatted so that it is easier to view. The first eigenvector differentiates the mountain and foreground from the sky in every single frame. In the second eigenvector, the gas plume in all 5 frames are together. This means that there is a continuity of the signal in all of the gas plume frames, and for that matter, the rest of the background signatures must also have the same continuity. The background frame has no trace of the gas plume, which is good. The third eigenvector gets the same signature in the same location over all of the frames, since this is a background signature, it is correctly behaving.

Strength of the Nyström extension

The Nyström extension is very fast, taking one minute when spectral clustering would take 30 minutes.

Has very little memory usage, allowing for a multiframe Nyström alternative to be implemented. Multiframe Nyström extension forms clusters across multiple frames, allowing for an incorporation of a temporal component.

Weaknesses of the Nyströpm extension

Is an approximation, therefore the results are not as good as spectral clustering.

Has the same weaknesses as spectral clustering, is not automated and needs human interpretation.

CHAPTER 5

SUMMARY AND CONCLUSIONS

Summary

This thesis goes over the algorithm of K-means, spectral clustering, and Nyström extension and applies them on long wavelength infrared hyperspectral images. The standard for trying to isolate gas plume signatures within a hyperspectral image relies on forming a probability map by using a set of known gas signature. Our results show that it is possible for clustering methods to isolate gas plumes within the hyperspectral images. K-means is a fast algorithm, that forms clusters around centroids. It is able to isolate the gas plume signature, but only in ideal situations due to the limitations of the algorithm. On the other hand, spectral clustering is a very slow algorithm that is able to accurately cluster the image. However it requires additional processing in order to arrive with usable results. The Nyström extension is an approximation to the spectral clustering results, taking a fraction of the time and producing good results.

Amongst the literature of hyperspectral data sets, the direction was to use single pixels and perform a statistical test to see if a pixel has a certain target spectral signature. There is no problem with this idea, however there is an abundance of information that is not being looked at. By performing clustering techniques, one does not limit the scope of information to a single pixel, but to the entire graph structure. Spectral clustering and the Nyström extension all look at the underlying structure of the data, and are able to separate the graph into the various elements that make up the image, such as isolating the mountain,

foreground, gasplume, and sky. This thesis has shown that clustering algorithms are an alternative to the probabilistic detection algorithms, because clustering is able to effectively isolate gas plumes in hyperspectral video sequences.

APPENDIX

MATLAB CODES

K-means

```
function [ newgroup ] = KMean( Data,k ,dist , kcenter)
% KMean rewritten to allow own distance metric.
% Currently only has random starting centers.
%
% Function call is:
%   [ newgroup ] = KMean( Data , k , dist )
% Data        is a matrix of row major data points.
% k           is the number of centers.
% dist        is the distance metric chosen amongst pdist2 metrics.
%                 If none specified, defaults to Euclidean.
% newgroup    is a vector containing the new group for each row of Data.
%
%
% Sample call:
%
%     randgroup = [rand(50,2)/2 ; rand(50,2)/2+.5];
%     %This is 100 points in the xy plane
%     figure(1);
%     plot(randgroup(:,1),randgroup(:,2),'.')
%     newgroup = KMean(randgroup,2,'euclidean');
%     figure(2); hold on;
%     plot(randgroup(newgroup==1,1),randgroup(newgroup==1,2),'r.')
%     plot(randgroup(newgroup==2,1),randgroup(newgroup==2,2),'b.')


[ r c ] = size(Data);


%set the start points if not defined
if nargin == 2
dist = 'euclidean';
end
if nargin < 4
kcenter=ceil(r*rand(k,1));
end
kcenter=Data(kcenter,:);

newgroup=zeros(r,1);

% datamat=zeros(r*c,k);
conv = 1;
```

```
newavg=zeros(k,c);
counter = 0;
oldgroup=zeros(r,1);
while (conv > 10^-3) && (counter < 100)
counter = counter+1;
datamat = pdist2(Data,kcenter,dist);
[~, newgroup]   = min(datamat,[],2);

for i = 1:k
newavg(i,:) = mean(Data(newgroup==i,:));
end
conv = max(newgroup~=oldgroup);
oldgroup=newgroup;
kcenter = newavg;
end
if counter==100
warning('counter hit 100'); %#ok<WNTAG>
end
```

Spectral Clustering

```
% SpectralClustering(remis,KNN,distmet,sigma,neigs)
%
% KNN is number of nearest neighbors
% distmet is distance matrix.  Do /help similarity2 for more info
% sigma is the sigma value
% neigs is the number of eigenvalues/vectors


function [test4 test5] = SpectralClustering(remis,KNN,distmet,sigma,neigs)

test = Similarity2(remis,KNN,distmet,sigma);
d = diag(sum(test,2));
dinv = d^-1;
dsqinv = sqrt(dinv);
test3 = dsqinv * test * dsqinv;
[test4 test5] = eigs(test3,neigs);
```

Similarity function

```
%Justin Sunu
function [ similarity ] = Similarity2(Data,k,dist,sigma) %, epsilon)
```

```
[ r c ] = size(Data);

k = min(r,k);

if nargin == 2
dist = 'euc';
end

if nargin < 4
sigma = 1;
end

switch dist
case 'selfc'
[cindex values ] = knnsearch(Data, Data, 'k', k , 'distance','cosine');
cindex = cindex(:);
t1val = values(:,k);
t1val = repmat(t1val,k,1);
t2val = t1val(cindex);
values = (values(:).^2)./(t1val.*t2val);
case 'cos'
[cindex values ] = knnsearch(Data, Data, 'k', k , 'distance','cosine');
cindex = cindex(:);
values = values/sigma;
case 'selfu'
[cindex values ] = knnsearch(Data, Data, 'k', k);
cindex = cindex(:);
t1val = values(:,k);
t1val = repmat(t1val,k,1);
t2val = t1val(cindex);
values = (values(:).^2)./(t1val.*t2val);
otherwise
[cindex values ] = knnsearch(Data, Data, 'k', k);
cindex = cindex(:);
values = values/sigma;
end

values = exp(-values(:));

rindex = transpose(1:r);
rindex = repmat(rindex,k,1);
```

```
index = [rindex cindex values ; cindex rindex values];

[~,index1] = unique(index,'rows','first');
repInd = setdiff(1:size(index,1),index1);
index(repInd,3)=0;

similarity = sparse(index(:,1),index(:,2),index(:,3),r,r);
```

Nyström Extension

```
function [V L] = nystrom_colo2_v2(data, kernel, num_samples, sigma,
                                  varargin) %#ok<INUSL>

if ~isempty(varargin)
flag = varargin{1};
else
flag = 1;
end

% randomly select samples
num_rows = size(data, 1);
permed_index = randperm(num_rows);
sample_data = data(permed_index(1:num_samples), :, :);
other_data = data(permed_index(num_samples+1:num_rows), :, :);
clear data;
% calculate the distance between samples themselves
disp('Calculating A');

if flag == 1
A = pdist2(sample_data, sample_data,'cosine');
A = exp(-A/sigma);
elseif flag == 2
A = SimilarityHaus(sample_data,sample_data,num_samples);
else
A = DistFuncCosFV9(sample_data, sample_data);
A = exp(-A/sigma);
end

% calculate the distance between samples and other points
disp('Calculating B');
other_points = num_rows - num_samples;

if flag == 1
```

```
B = pdist2(sample_data, other_data,'cosine');
B = exp(-B/sigma);
elseif flag == 2
B = SimilarityHaus(sample_data,other_data,other_points);
else
B = DistFuncCosFV9(sample_data, other_data);
B = exp(-B/sigma);
end

% clear sample_data other_data;

% Normalize A and B using row sums of W, where W = [A B; B' B'*A^-1*B].
% Let d1 = [A B]*1, d2 = [B' B'*A^-1*B]*1, dhat = sqrt(1./[d1; d2]).

disp('Normalizing A and B for Laplacian...');
B_T = B';
d1 = sum(A, 2) + sum(B, 2);
d2 = sum(B_T, 2) + B_T*(pinv(A)*sum(B, 2));
dhat = sqrt(1./[d1; d2]);
A = A .* (dhat(1:num_samples)*dhat(1:num_samples)');
B1 = dhat(1:num_samples)*dhat(num_samples+(1:other_points))';
B = B .* B1;
% clear d1 d2 B1 dhat;

% Do orthogalization and eigendecomposition
disp('Orthogalizing and eigendecomposition...');
Asi = sqrtm(pinv(A));
B_T = B';
BBT = B*B_T;
W = single(zeros(size(A, 1)+size(B_T, 1), size(A, 2)));
W(1:size(A, 1), :) = A;
W(size(A, 1)+1:size(W, 1), :) = B_T;
% clear B B_T;
% Calculate R = A + A^-1/2*B*B'*A^-1/2
R = A + Asi*BBT*Asi;
R = (R + R')/2; % Make sure R is symmetric, sometimes R
                %can be non-symmetric because of numerical inaccuracy
[U L] = eigs(R,num_samples);
[~, ind] = sort(diag(L), 'descend');
U = U(:, ind); % in decreasing order
L = L(ind, ind); % in decreasing order
clear A R BBT;
```

```
W = W*Asi;
V = W*U(:, 1:num_samples)*pinv(sqrt(L(1:num_samples, 1:num_samples)));

V(permed_index,:) = V;
V = real(V);
L = 1-diag(L);
```

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] Dimitris Manolakis. "Standoff hyperspectral chemical agent detection: phenomenology, sensors, data, and algorithms," presented at Defense Threat Reduction Agency Algorithms conference 2010.

[2] Joshua B. Broadwater, Diane Limsui, and Alison K. Carr. "A primer for chemical plume detection using LWIR sensors," Technical Paper, National Security Technology Department, Las Vegas, NV, 2011.

[3] Ulrike von Luxburg. "A tutorial on spectral clustering." *Statistics and Computing*, vol. 17, pp. 1-32, 2007

[4] Fan R. K. Chung. *Spectral graph theory*. Washington, DC: American Mathematical Society, 1997.

[5] Andrew Y. Ng, Michael I. Jordan, and Yair Weiss. "On spectral clustering: analysis and an algorithm." *Advances in Neural Information Processing Systems*, vol 14, pp. 849-856, 2001.

[6] Charless Fowlkes, Serge Belongie, Fan Chung, and Jitendra Malik. "Spectral grouping using the Nyström method." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, pp. 214-225, Feb 2004.

[7] Justin Sunu, Jen-Mei Chang, and Andrea L. Bertozzi. "Simultaneous spectral analysis of multiple video sequence data for LWIR gas plumes," in *SPIE Conference on Defense Security and Sensing*, 2014, preprint.

[8] Ekaterina Murkerjev, Justin Sunu, and Andrea L. Bertozzi. "Graph MBO method for multiclass segmentation of hyperspectral stand-off detection video," in *IEEE International Conference on Image Processing*, 2014, preprint.

[9] Torin Gerhart, Justin Sunu, Lauren Lieu, Ekaterina Merkurjev, Jen-Mei Chang, Jèrôme Gilles, and Andrea L. Bertozzi. "Detection and tracking of gas plumes in LWIR hyperspectral video sequence data," in *SPIE Conference on Defense Security and Sensing*, vol. 8743, (May 2013), doi:10.1117/12.2015155.