OUTLIER COLOR IDENTIFICATION FOR SEARCH AND RESCUE

Mukund Ramachandran, William Moik

Boston University
Department of Electrical and Computer Engineering
8 Saint Mary's Street
Boston, MA 02215
www.bu.edu/ece

Dec 13, 2013

Technical Report No. ECE-2013-03

Contents

List of Figures

List of Tables

1 Introduction

Search and rescue operations require locating the person of interest in an outdoor environment such as the wilderness prior to extraction. Often times, unmanned aerial vehicles (UAV's) with high resolution cameras are used to expedite the searching process, improve rate of detection, and reduce man-hours. However, when recording video from a high elevation, the object of interest represents a small portion of the image. This presents a major obstacle to the analyst who observes the footage by eye and failure rates are high. In 2011, the US Coast Guard alone failed to save over 170 lives [13]. To further expedite and increase the chance of detection, automated detection methods are used to assist the analyst in locating objects of interest in the footage. Man-made objects of interest such as articles of clothing or large blankets can be useful in locating a missing person. These objects tend to differ greatly in color from the surrounding terrain. By using an automated way of finding these anomalous colors, the missing persons can be located with greater speed and accuracy.

2 Literature Review

The literature for this specific problem is focused on object detection that leaves a characteristic signature in hyper-spectral imagery. These are images that have many spectral bands including bands outside of the visible spectrum. We are concerned with only the color leaving a characteristic signature as compared to the background.

One approach to detecting unusually colored objects is through the use of color histograms. *Rasmussen, Thornton, Morse[1]* transform the images from the RGB color space into the Hue Saturation Value (HSV) color space and identify hues as common or uncommon by partitioning the space of hues into a histogram.

More sophisticated approaches that do not pre-define a partition of hues consist of point-based and region-based segmentation methods. Point-based algorithms classify every pixel in the image as normal or anomalous based on a threshold test. *Reed and Yu [2]* propose a method to

classify a pixel by comparing its features to the background using a Mahalanobis Distance metric. The work by *B. S. Morse, D. Thornton and M. A. Goodrich [3]* implements the Reed and Yu Approach (RX detector) using a dual sliding window approach in a wilderness search and rescue setting. Several variants of the RX algorithm have been proposed that seek to optimize its performance under the assumption of prior information about the anomaly target or image *[4][9]*.

Region-based segmentation methods group pixels based on criteria and are not limited to outlier detection methods in search and rescue settings. The K-means method works to segment the image into distinct clusters *[5][10]*. . The EM algorithm assumes a Gaussian mixture distribution model of membership for a pixel in multiple clusters rather than a deterministic assignment of a pixel to a single cluster *[7][8]*. Density based algorithms assume that normal data occur in regions of high density while outliers are points in regions of low density. The work by *Breunig, Hans-Peter Kriegel et Al [6]* goes one step further by assigning an outlier score to each cluster based on their inter-cluster distances known as the Local Outlier Factor.

3 Solution

We have assumed that we are given an input image taken by the UAV operating in an unknown highly textured natural environment. Therefore no prior knowledge is assumed in our algorithms about the content or color distribution of image.

The anomalies are assumed to be extremely small relative to total image. They also are defined by an outlier color relative to background. In addition this anomalous object is assumed to be of a relatively uniform hue rather than a camouflage pattern. Also the anomalous pixels are assumed to be spatially correlated with neighboring pixels and are not single pixel anomalies.

We will consider two approaches to handle these constraints. The first one is a pixel based classification approach and the second one is a region based segmentation approach with the goal of classifying an object.

3.1 Point-Based RX Detector

In this approach, the assumption is made that a normal pixel is drawn from a Multivariate Gaussian distribution specified by $N(\mu, \Sigma)$

We use an approach similar to [12] to develop a threshold test to classify each pixel under the following hypothesis. Null Hypothesis is that the pixel follows the normal distribution, while the alternate hypothesis is that the pixel follows a uniform distribution.

Null Hypothesis H_0: $X \approx N(\mu, \Sigma)$

Alternative Hypothesis H_1: $X \approx \text{Uniform}(C)$

The log-likelihood function is used to maximize the probability of detecting an anomaly when an anomaly exists for a fixed rate of false alarm which is controlled by the threshold T.

$$f(x) = \log \frac{p(x: H_1)}{p(x: H_0)} \gtrless T$$

Expanding the above equation we get:

$$f(x) = \log p(x: H_1) - \log p(x: H_0) = \log c - \log p(x: H_0) \gtrless T$$

Assuming that x is an n-dimensional vector, the above equation can be written as:

$$f(x) = \log c + \left(\frac{n}{2}\right)\log(2\Pi) + \left(\frac{1}{2}\right)\log(|\Sigma|) + \left(\frac{1}{2}\right)\left((x-\mu)^T\Sigma^{-1}(x-\mu)\right) \gtrless T$$

The constants and multiplicative factor can be grouped into a new threshold S to yield the Mahalanobis distance between the pixel and the background region with the form:

$$(x-\mu)^T(\Sigma)^{-1}(x-\mu) \gtrless S$$

x: Target Feature Vector

μ: Mean of Background Region

Σ: Covariance Matrix of Background Region

S: Threshold

3.2 Region Based: K-Means Approach

The K-Means algorithm partitions the N observations in our feature space into a set of k clusters by minimizing the function F, where the variables c_i is the cluster centroid and x_m is an observation in the selected feature space. There are k clusters with N observtions.

$$F = \sum_{m=1}^{k} \sum_{i=1}^{N} (x_i - c_m)^2$$

Since this is a non-linear minimization problem, an iterative algorithm is used to find the local minimum [11] .The first step is to randomly initialize a set of K cluster centroids out of the N observations. Then each observation x_i is assigned to the closest centroid based on the Euclidean Distance metric. Then an update of the centroids are performed given the new configuration. The process is given by formula, where c is the space pertaining to a specific cluster.

This process is repeated until the cluster centroids no longer move.

3.3 Outlier Centroid Detection

Based on the K-Means segmentation, we proceed to classify each cluster as normal or anomalous based on a metric applied to the cluster centroids. The metric we use to do this is the number of neighboring centroids enclosed in a region defined by radius R for each centroid based on the Euclidean distance. If more than N_{max} neighbors is contained in this region, the centroid and its associated cluster points are classified as normal as this means that the centroid is in a dense region of the data. If less than N_{max} neighbors are contained in this radius, the centroid and its associated cluster points are classified as anomalous as this means the centroid is in a sparse region of the data.

$$\mathrm{N}(C_m) = \sum_{m,i \in C} \mathrm{d}(C_m, C_i) \leq \mathrm{R}$$

The intuition behind the metric is illustrated via a synthetic example below:

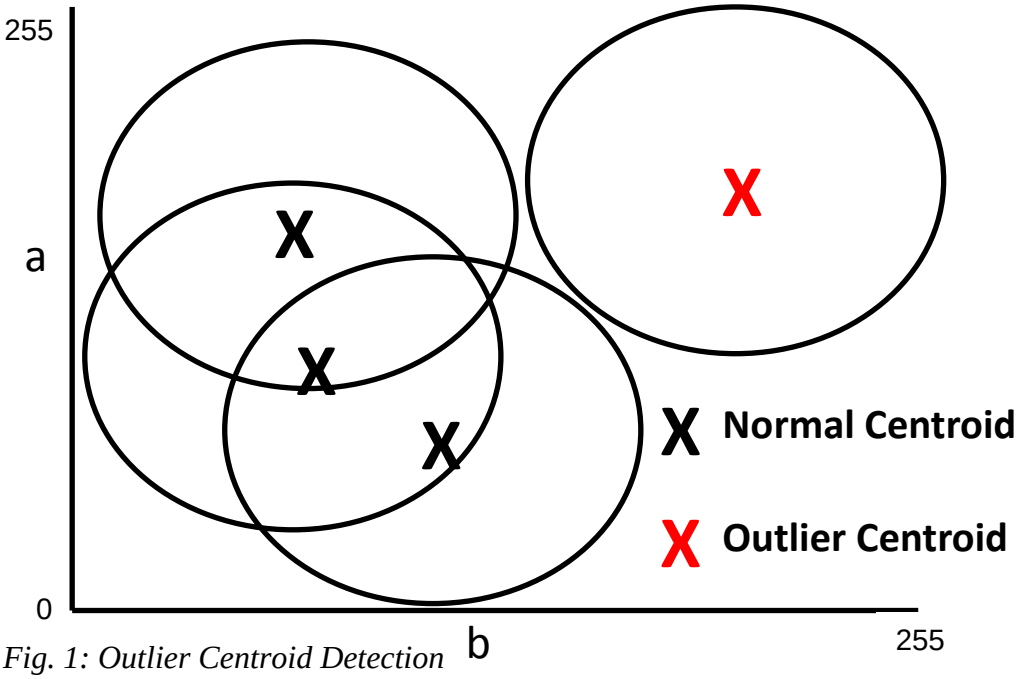

Fig. 1: Outlier Centroid Detection

3.4 Markov Model

To incorporate the knowledge that anomalous pixels are correlated, we apply a Markov Random Field E to the resulting label matrix from the previous step using the approach contained in [11]. If we assume that e, a specific realization of E, is known for all j ≠ k, the task is reduced to deciding a label for e[k]. Define e[k] = 0 as a normal pixel and e[k] = 1 as an anomalous pixel. Define e^N as the label field when e[k] = 0 and e^A be the label field when e[k] = 1. The Hypothesis test then at e[k] is:

$$\frac{P(I = i|e^N)}{P(I = i|e^A)} \underset{N}{\lessgtr} \eta \frac{P(E = e^A)}{P(E = e^N)}$$

The probability $P(I = i|e^N)$ is the joint probability that the Random Field I assumes a realization i, given the label field realization e^N. In [11] it is shown that this can be simplified to:

$$\frac{P(I[k] = i[k]|e^N)}{P(I[k] = i[k]|e^A)} \underset{N}{\lessgtr} \eta \frac{P(E = e^A)}{P(E = e^N)}$$

with the assumption that the components are mutually spatially independent and that the ratio of the left-hand side probabilities only differ at k. Since E is a Markov Random Field, the Hammerson-Clifford theorem allows us to model the right-hand side as a Gibbs distribution, with temperature У and potential function V defined on {k , j}. We use a 3x3 neighborhood with 2 element cliques and a potential function to credit neighbors which are the same and penalize those that are different. The Ising potential can be used as we only have two states, normal or anomalous:

$$V(k,j) = \begin{Bmatrix} 0\ if\ e[k] = e[j] \\ 1\ if\ e[k] \neq e[j] \end{Bmatrix}$$

This can be incorporated into the equation to further simplify our expression to:

$$\frac{P_N(I[k])}{P_A(I[k])} \underset{N}{\lessgtr} \theta\left(\exp\left(\frac{1}{\gamma}\right)(Q_A[k] - Q_N[k])\right)$$

$Q_A[n]$: Number of Anomalous Pixels
$Q_N[n]$: Number of Normal Pixels
ϒ: Natural temperature parameter to control strength of MRF Model
θ: Parameter to control deviation of Gaussian Random Variable
P_N: Gaussian Distribution
P_A: Uniform Distribution

3.5 Data Set

Since we were not able to find an available search and rescue image dataset taken from a UAV, we constructed our own images to depict this scenario. We have used a set of images (800x800) in environments typical of search and rescue settings such as in the wilderness, ocean and woods. We then place several synthetic man-made objects with sizes that are typical sizes in the range of jackets to blankets and whose colors differ significantly from the background and whose luminance blends in naturally into the environment. In the Figure below, Image 1 is a forested mountain, Image 2 is an ocean and Image 3 is a wooded region. The three different synthetic objects are down-sampled to about 12x12 and placed at locations that are typical of search and rescue situations, i.e. not in the sky. We chose the three objects to have colors red, orange and purple as they visually differ significantly from the background. In Image 2, the jacket anomalies were cut in half as to simulate persons floating in water. Once we finished constructing the data set, we manually ground truth the anomalous pixels in a binary label matrix.

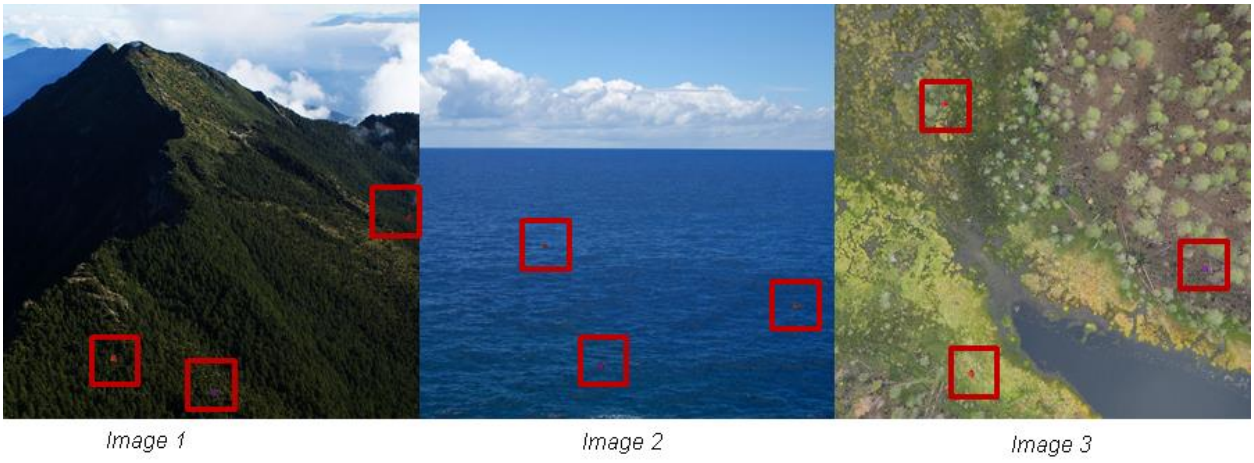

Fig. 2: Sample Images

Fig 3: Image 3 Anomaly Zoom

4 Implementation

We transform the image from the RGB color space to the perceptually uniform LAB color space and discard the L component of this vector as the luminance content in the image varies significantly. The two approaches are implemented as follows:

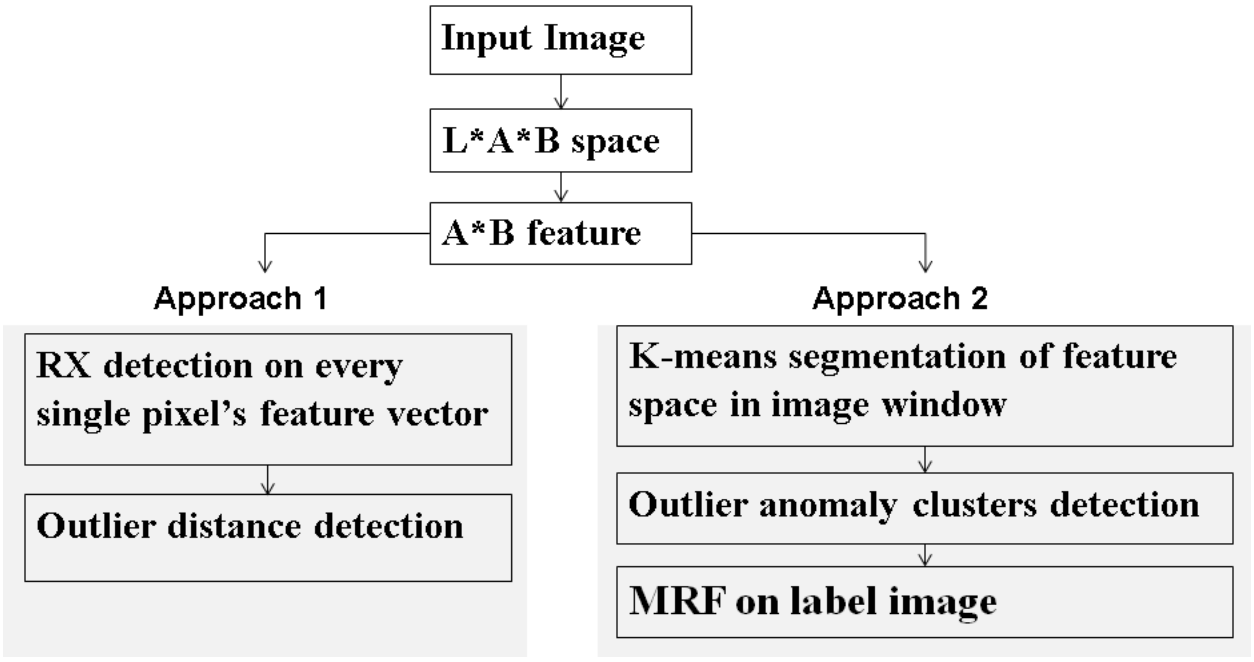

Fig. 4 Implementation Overview

4.1 RX Detector Algorithm

The RX Detector classifies each pixel's features in the image as normal or anomalous. To avoid corrupting the background region statistics with the anomalous values a guard window is placed around the test pixel. We have chosen a 30x30 inner window to capture the anomaly and a 60x60 to capture an effective amount of the background distribution. The diagram below depicts the sliding window method. We implement the Mahalanobis threshold test between each test pixel and the background region to classify the pixel as normal or anomalous. This process is repeated for every pixel in the image. This algorithm as a complexity of $O(N^4)$ because for every pixel in a 2-D image it processes a smaller 2-D subsection of the original image. Therefore the complexity depends on four variables; the height and width of the original image and the height and width of the subsection outer window.

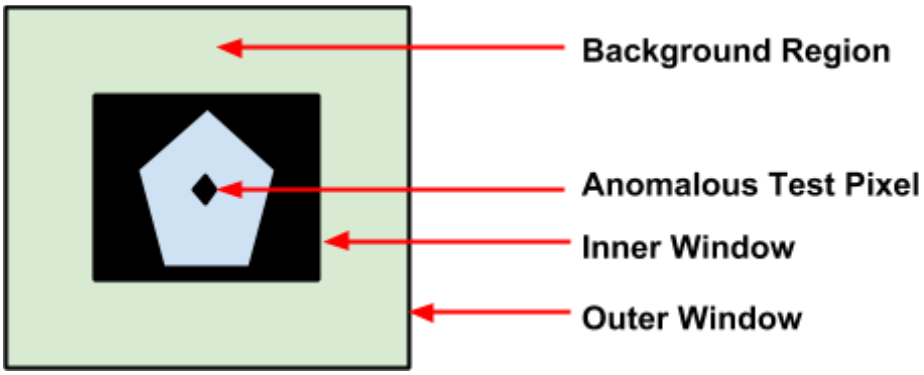

Fig. 5 RX Detector Implementation

4.2 K-Means Algorithm

We divide the image into distinct windows each of size 100x100. We choose K=100 clusters to ensure that we capture the anomaly object in a cluster given a highly varying textured color environment. We set the number of neighbors for each centroid at 2. Therefore, a cluster centroid is considered an anomaly if it has 2 or fewer neighbors within the region defined by the threshold radius R, while it is classified as normal if it has more than 2 neighbors in the region. The R value is varied depending on the windowed image. This is done as each windowed sample may vary in environment from another windowed sample from the same original image. We apply the MRF on the resulting K-Means label image for three iterations with the natural Gibbs temperature set to 1.

5. Experimental Results

5.1 Visual Performance

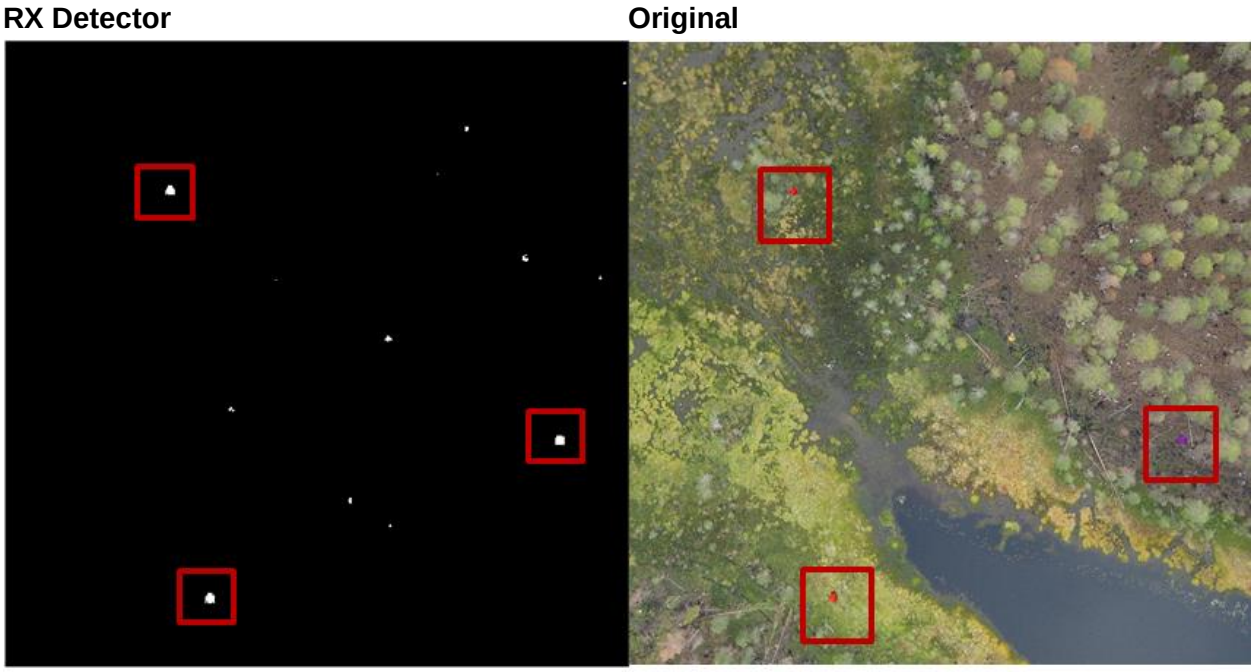

Figure 6: RX Detection results from Image 3

In Figure 6, it can be seen that the three colored anomalies were successfully detected. However, there are also a few false positive readings in the image.

K-Means **Original**

Figure 7: K-Means Detection results from Image 2

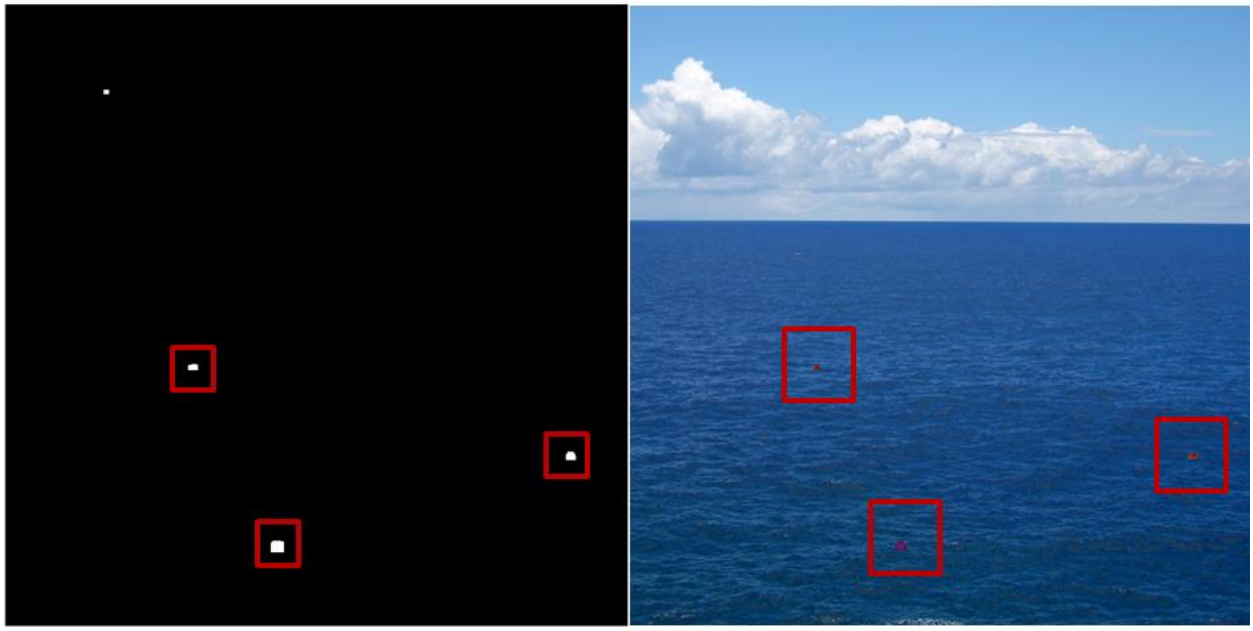

Figure 8: K-Means with MRF Detection applied to Figure 7

Sample detection results prior to and post MRF are shown above. We can see that with the addition of the MRF, the false positives scattered in the image become eliminated. Additionally, the pixels within the right most anomaly are filled because the MRF is designed to enforce object cohesion.

5.2 Quantitative Performance Comparison

We compare the algorithm results objectively with the ground truthed images by plotting Receiver Operating Characteristic (ROC) curves on our data set to capture the trade-off between True Positive Rate and False Positive Rate. True Positive Rate is defined as the TPR = True Anomaly Pixels / Total Anomalous Pixels. False Positive Rate is defined as FPR = False Anomalous Pixels / Total Normal Pixels. For the RX Detector, lowering the Threshold(S) increases the TPR but increases the FPR. In the K-Means approach, increasing the Threshold(Radius of the Neighborhood), results in a higher TPR but with an increase in the FPR. The RX Detector achieves a full 100% detection at close to 5% while the KMeans Algorithm requires a 12% false positive rate to achieve the 100% detection accuracy. The Area under each of the curves (AUC) is calculated as an objective way of comparing the approaches in Table 1. Consistent with our visual observations of the ROC chart, the RX Detector performed better than K-Means in the AUC chart by 0.0369. However with the addition of the MRF, the delta between RX Detector and K-Means becomes only 0.0229 as majority of the false positives are eliminated.

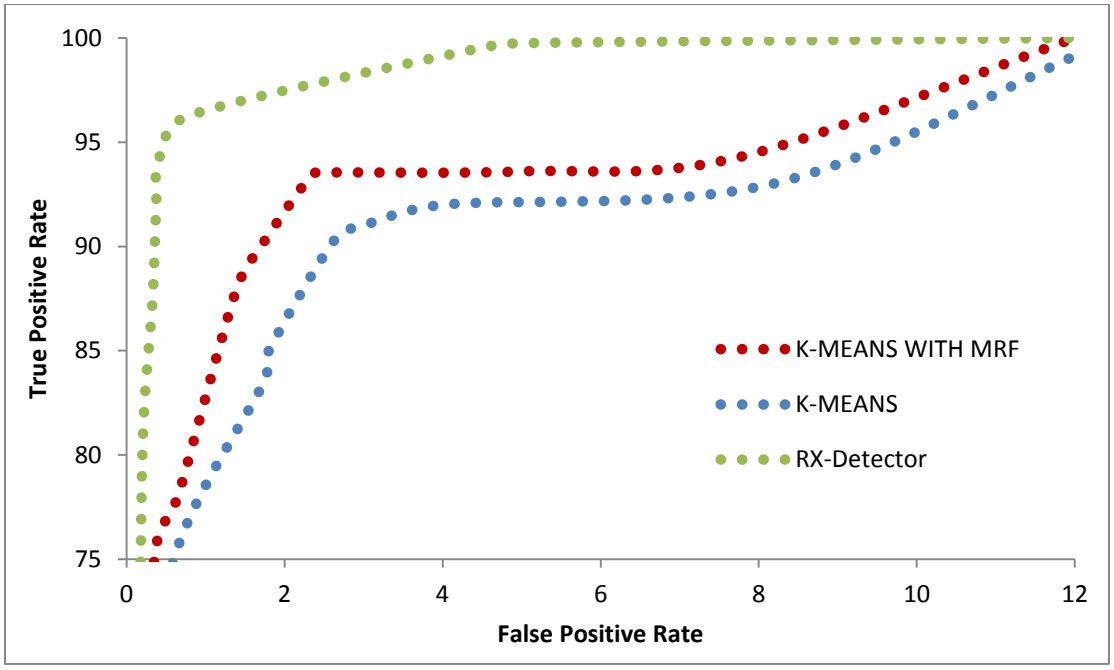

Figure 9 ROC performance

Area Under Curve Comparison

Method	AUC
RX Detector	0.9968
K-MEANS with MRF	0.9738
K-MEANS	0.9598

Table 1

6. Conclusion

The ideal objective in any search and rescue setting should be to get a 100% true positive detection rate. This is because it is a more severe error to miss the detection of a lost person than to detect objects that are not anomalies. However, this may mean that analyst will manually sift through the false positives to cull the results. This approach can be extended to video frames that contain no detected anomalies as the speed at which the analyst views a video can be greatly reduced. Computationally, we noticed a significant improvement using the K-Means approach over the RX Detector. This is due to the fact that RX Detector calculates the covariance matrix of a different neighborhood region at each test pixel, whereas the K-Means approach computes Euclidean distances in non-overlapping windows through the image. This is especially a concern for analyzing time sensitive video sequences. Despite the high FPR using a K-Means approach, we believe it can provide more useful information beyond just outlier detection such as content-aware information (location of sky, grass, water). To improve the accuracy of our K-Means approach, we could also adapt the number of clusters to local window statistics instead of choosing a fixed global number of clusters for each window. Also we have thought about reducing the clusters to a very small amount such as 2 or 3 in a window and testing each point similar to the RX Detector approach. The assumption here would be that the anomalous pixels would be spread apart among the different clusters and they can be determined by measuring the Mahalanobis distance to the background cluster distribution. Lastly we can use a filtered and down sampled image of the original. This will significantly decrease runtime and would be very practical for anomaly objects that are multiple pixels in size. This method will reduce computation time, under the assumption that the hue of the anomaly is highly uniform.

7 References

[1] Nathan D. Rasmussen, Daniel R. Thornton, Bryan S. Morse, “Enhancement of unusual color in aerial video sequences for assisting wilderness search and rescue”

[2] I. Reed and X. Yu, "Adaptive multiple-band CFAR detection of an optical pattern with unknown spectral distribution,"

[3] B. S. Morse, D. Thornton and M. A. Goodrich, "Color Anomaly Detection and Suggestion for Wilderness Search and Rescue,"

[4] Timothy E. Smetek, Kenneth W. Bauer, “Finding Hyperspectral Anomalies Using Multivariate Outlier Detection”

[5] Tapas Kanungo, Senior Member, IEEE, David M. Mount, Member, IEEE, “An Efficient k-Means Clustering Algorithm: Analysis and Implementation”

[6] Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng, Jörg Sander, “LOF: Identifying Density-Based Local Outliers”

[7] James Theiler1 and D. Michael Cai2, “Resampling Approach for Anomaly Detection in Multispectral Images”

[8] V. Ilango et all, “Expected Maximization Methods in Time Series Data”

[9] Chein-I Chang and Shao-Shan Chiang “Anomaly Detection and Classification for Hyperspectral Imagery”

[10] V. Ilango and R. Subramanian, “Outliers Detection for Regression using K-Means and Expected Maximization Methods in Time Series Data”

[11] J. McHugh, J.Konrad, V.Saligrama, and P. –M Jodoin, “Foreground-adaptive background subtraction”

[12] L. R. Bachega, J. Theiler and C. A. Bouman, "Evaluating and improving local hyperspectral anomaly detectors"

[13] United States. US Coast Guard. Office of Search and Rescue. *United States Coast Guard Search and Rescue Summary Statistics 1964 Thru 2011*. By LT T. Gorgol. Web.

[14] T. Bolukbasi, P. Tran, “Outlier Color Identification for Search and Rescue” Boston University 2012

8 Appendix – MATLAB Code

RX Detector

```
function [Maha,Classify] = RXDetector(img,i_size,o_size)
imrgb = im2double(img);
cform = makecform('srgb2lab');
imlab = applycform(imrgb,cform);
imlabxL = imlab(:,:,2:3);
image = imlabxL;
Threshold = 60;
Horizontal = size(image,1);
Vertical = size(image,2);
boundary = floor(o_size/2);
Maha = zeros(Horizontal,Vertical);
guard_window = (o_size - i_size)/2 + 1;
Classify = false(Horizontal, Vertical);

for i=1:1:Horizontal
   for j=1:1:Vertical
        mean_vector = zeros(2,1); point = zeros(2,1);
        outer_block = image(max(i-boundary+1,1):min(i+boundary-1,Horizontal),max(j-
        boundary+1,1):min(j+boundary-1,Vertical),:);
        outer_block(guard_window:guard_window+i_size-
        1,guard_window:guard_window+i_size-1,:) = NaN;
        mean_vector(1) = nanmean(nanmean(outer_block(:,:,1)));mean_vector(2) =
        nanmean(nanmean(outer_block(:,:,2)));
        cov_matrix = nancov(outer_block(:,:,1),outer_block(:,:,2));
        det = 1/(cov_matrix(1)*cov_matrix(4)-cov_matrix(3)*cov_matrix(2));
        invcov_matrix = det.*[cov_matrix(4), -cov_matrix(3); -cov_matrix(2), cov_matrix(1)];
        point(1) = image(i,j,1); point(2) = image(i,j,2);
        Maha(i,j) = (point-mean_vector)'*(invcov_matrix)*(point-mean_vector);
```

```
   end
end

Classify = Maha>Threshold;

figure(1)
imshow(img);

figure(2)
surf(Maha)

figure(3)
imshow(Classify)

end
```

K-Means

```
clear all;
truth_1 = imread('X:\EC 520\Project\true_test_1_1.tiff');
num_clusters = 100;
window_size = 100;

Neighbors = 2;%Isolated
R_Threshold = 3.25;%Radius of each centroid point
%Density = Neighbors / (R_Threshold);

imrgb = imread('X:\EC 520\Project\test1_1.tiff');
imrgb = im2double(imrgb);
cform = makecform('srgb2lab');

imlab = applycform(imrgb,cform);
```

```
imlabxL = imlab(:,:,2:3);

label_matrix = zeros(size(imrgb,1),size(imrgb,2));

for i=1:window_size:size(imlabxL,1)
   for j = 1:window_size:size(imlabxL,2)
      imlabxLTemp =
imlabxL(i:min(i+window_size,size(imlabxL,1)),j:min(j+window_size,size(imlabxL,2)),1:2);
      nrows = size(imlabxLTemp,1);
      ncols = size(imlabxLTemp,2);
      flatImg = double(reshape(imlabxLTemp,nrows*ncols,2));
      [cluster_idx,cluster_center,sumd] =
kmeans(flatImg,num_clusters,'EmptyAction','Singleton','Start','Uniform','Replicates',1);
      pixel_labels = reshape(cluster_idx,nrows,ncols);
      p_dist = squareform(pdist(cluster_center));
      p_dist(logical(eye(size(p_dist)))) = 0;
      for y = 1:length(p_dist)
         num_neighbors(y) = sum(p_dist(y,:)<R_Threshold);
      end
      outlier_cluster = find(num_neighbors<=Neighbors);
      c = ismember(pixel_labels,outlier_cluster);

label_matrix(i:min(i+window_size,size(imlabxL,1)),j:min(j+window_size,size(imlabxL,2)))=c;
   end
end
%}

figure(1)
imshow(pixel_labels,[]), title('image labeled by cluster index');
figure(2)
imshow(imrgb);
```

```
figure(3)
imshow(label_matrix); title('Initial Labels')


%Reapply MRF iteratively (3 times)
final_label_matrix = MRF_MOD(label_matrix);
final_label_matrix = MRF_MOD(final_label_matrix);
final_label_matrix = MRF_MOD(final_label_matrix);

figure(4)
imshow((final_label_matrix)); title('MRF Labels')

post_final_label_matrix = zeros(size(final_label_matrix,1), size(final_label_matrix,2));

for ii = 1:size(final_label_matrix,1)
   for jj = 1:size(final_label_matrix,2)
       if final_label_matrix(ii,jj)<1
          post_final_label_matrix(ii,jj)=0;
       elseif final_label_matrix(ii,jj)>=1
          post_final_label_matrix(ii,jj)=1;
       end
   end
end

figure(5)
imshow((post_final_label_matrix)); title('Binary MRF Labels');
```

MRF Model

```
function [label] = MRF_MOD(label)
T=1;num_neighbors = 8; Ising = 1;
```

```
support = ones(3);support(2,2) = 0 ;
A = -num_neighbors*Ising + 2.*conv2(label,support,'same');
label = 0.00811.*exp(1/T.*A);
end
```

ROC

```
function [ROC] = myroc(C_idx,truth)
%ROC returns a percentage vector of [true_pos false_pos true_neg false_neg]

C_idx = C_idx(1:end,1:end);


A = size(C_idx,1);
B = size(C_idx,2);
true_pos = 0;
false_pos = 0;
true_neg = 0;
false_neg = 0;

for i = 1:A
   for j = 1:B

      if truth(i,j) == 1
         if C_idx(i,j) > 0
            true_pos = true_pos + 1;
         elseif C_idx(i,j) == 0
            false_neg = false_neg + 1;
         end
      elseif truth(i,j) ~= 1
         if C_idx(i,j) > 0
            false_pos = false_pos + 1;
         elseif C_idx(i,j) == 0
            true_neg = true_neg +1;
         end
```

```
      end
   end
end

%percentages
totalgroundpos = sum(sum(truth));
totalgroundneg = A*B - totalgroundpos;
true_pos_percent = 100*true_pos/(true_pos+false_neg);
false_pos_percent = 100*false_pos/(false_pos+true_neg);

ROC = [true_pos_percent false_pos_percent];
```

DWT

```
dwtmode('per');
[LLA,LHA,HLA,HHA] = dwt2(imlab,'db4');
imlabw = LLA;
```