OUTLIER COLOR IDENTIFICATION FOR SEARCH AND RESCUE

Tolga Bolukbasi, Philip Tran

Dec 13, 2012

Boston University

Department of Electrical and Computer Engineering

Technical Report No. ECE-2012-07

BOSTON UNIVERSITY

OUTLIER COLOR IDENTIFICATION FOR SEARCH AND RESCUE

Tolga Bolukbasi, Philip Tran

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

Dec 13, 2012

Technical Report No. ECE-2012-07

Contents

List of Figures

List of Tables

1 Introduction

Search and rescue operations involve finding and extracting stranded persons in distress situations. A tool that is gaining prevalence in search and rescue is the use of high perspective video captured from a camera mounted on aircraft (e.g. unmanned aerial vehicles (UAV)). In order to expedite the search and rescue operation, cameras require a large field of view to survey as much area as possible. As a result, the size of missing persons or objects of interest becomes small relative to the field of view, and high resolution cameras are necessary for person/terrain differentiation to be possible. With these conditions in mind, relying on a human observer alone becomes impractical. Displaying high resolution video without downscaling is often technically infeasible, and even if it was possible, the inability of the human visual system to concentrate on different spatial locations at the same time prevents reliable object tracking in high resolution video. Consequently, some automated methods for locating objects of interest should be implemented in order to assist the observer.

Automatic target detection could be used to approach this problem; however, the objects of interest are not well-defined for most search and rescue operations. Directly finding the person in distress would be ideal, but in some scenarios, scattered objects such as a shirt or a blanket may provide some useful information. ATR systems usually assume a pre-known model for the object of interest. Therefore, anomaly detection would provide a more general approach to the issue. Anomalies are defined as unusual patterns that do not fit with "normal" data, and in search and rescue, it is appropriate to assume the color of the objects of interest differs substantially from its immediate surroundings. This project aims to exploit this difference by applying local anomaly detection techniques to detect objects of interest.

2 Literature Review

While there is much research in the field of anomaly detection for videos and hyperspectral images, only recently has there been a publication for the use of color anomaly detection in a search and rescue setting. The field of video anomaly detection is usually concerned with behavioral anomalies; therefore, research of that field often involves the use of temporal content as features [1]. Since this project is more focused on non-temporal anomalies, we will mainly be concerned with the findings from hyperspectral anomaly detection research.

Common techniques used for anomaly detection in hyperspectral imagery include cluster-based techniques, Reed-Xiaoli (RX) detector and its variants [2], [3], [4], and dual window eigen-separation transform (DWEST) algorithm [5]. A clustering technique called cluster-based anomaly detection (CBAD) assumes the distribution of normal pixel vectors within clusters of an image can be modeled as Gaussian mixture models (GMM) [6]. Anomalies in this case are vectors that deviate from the estimated model by considerable amount. The RX detector is considered the benchmark for anomaly detection in hyperspectral images [3]. The detector assumes feature vec-

tors in a processing window are Gaussian-distributed. It then estimates the mean and variance of the distribution and calculates the Mahalanobis distance between features of the center pixel and the mean feature vector. These distances are used to determine whether a pixel is anomalous or not. DWEST is an eigen-based technique that utilizes two windows, an inner and outer window, to maximize the distances between target and background to extract anomalies. Assumptions underlying DWEST are that the statistics between background and the area of interest are sufficiently different and that data distributions are Gaussian, for mathematical simplicity.

A recent publication by Morse et al adapted some of the previously discussed hyperspectral image techniques to color anomaly detection for search and rescue [7]. They compare the performances of the following approaches: RX algorithm [2], vector quantization (CBAD) [6], k-means clustering, and EM algorithm [8] on several aerial *RGB* images. They found that the RX algorithm outperformed every other method, given the proper parameter selection. Therefore, the RX algorithm will serve as a reliable comparison to the methods that will be proposed for this project.

All of the anomaly detection methods mentioned thus far have focused only on point anomalies [9]. However, anomalous pixels of an image typically have some spatial correlation. Modeling images as Markov random fields (MRF) is a commonly used approach to incorporate spatial correlation between pixels, and some recent papers have applied this approach in order to improve their detection results [10] [11]. We will also use MRFs to improve detection results.

3 Solution

In order to properly present the methods for anomaly detection, concepts and notation that will be used in this report must be introduced.

The main assumption about the object of interest's characteristics is that it has a distinct color from its surroundings. As a result, our detection algorithm was built to depend heavily on distances between the color features, and images should be converted in a manner in which the algorithm could perceive it similar to how a human observer would. The initial stage of the project involves transforming the image from *RGB* to *L*a*b** color space. It is not guaranteed that the outlier will have a distinct brightness level from its surroundings. Moreover, it is seen that the variance of luminance in highly textured portions of the image is significantly large. Based on these observations, the L^* component was excluded from our feature vector, since it only adds to the distances between the normal points. Let $\vec{\mathcal{I}}[\boldsymbol{x}]$ be a random field and its realization be an a^* and b^* component image represented by $\vec{I}[\boldsymbol{x}] = [A[\boldsymbol{x}], B[\boldsymbol{x}]]^T$, where $\boldsymbol{x} = [x_1, x_2]^T$represents spatial coordinates. $\vec{\mathcal{I}}$ and $\vec{I}$ without spatial indices denote the entire vector random field and its realization, respectively.

We propose a two-stage approach to color anomaly detection. In the first stage, the pixels will be assumed to be spatially independent, which provides some simplification of formulae. However, since objects of interest will always be comprised of more than a single pixel, accounting for spatial dependence will have a positive effect on detection

performance. In the second stage, we will incorporate the spatial correlation between neighboring pixels and formulate the problem accordingly.

3.1 Binary Hypothesis Testing for Anomaly Detection

Color anomaly detection can be considered as a binary classification task. There are assumed to be two classes: "normal" and "anomalous", and binary hypothesis testing can be used to classify each pixel to one of the two classes. When pixels are assumed to be spatially independent, the hypothesis test can be written as:

$$\frac{P(\vec{\mathcal{I}}[\boldsymbol{x}] = \vec{I}[\boldsymbol{x}]|\eta)}{P(\vec{\mathcal{I}}[\boldsymbol{x}] = \vec{I}[\boldsymbol{x}]|\xi)} \underset{\xi}{\overset{\eta}{\gtrless}} c \cdot \frac{P(\xi)}{P(\eta)} \tag{3.1.1}$$

Symbols η and ξ denote the classes "normal" and "anomalous", respectively. The terms on the left side of equation (3.1.1) represent the probability of observing realization $\vec{I}[\boldsymbol{x}]$, given that the pixel at $\boldsymbol{x}$ belongs to an "anomalous" region or "normal" region. The probabilities on the right side of equation (3.1.1) represent the a priori probability of belonging to the "anomalous" class or "normal" class. Due to the unknown nature of the anomaly and the large gamut of possible colors that can be realized by an "anomalous" pixel, it is logical to assume $P(\vec{\mathcal{I}}[\boldsymbol{x}] = \vec{I}[\boldsymbol{x}]|\xi)$ to be constant. In addition, when the a priori probability of a pixel being "anomalous" or "normal" is assumed to be the same at all positions of the image, the equation (3.1.1) simplifies to:

$$P(\vec{\mathcal{I}}[\boldsymbol{x}] = \vec{I}[\boldsymbol{x}]|\eta) \underset{\xi}{\overset{\eta}{\gtrless}} \alpha \tag{3.1.2}$$

As a result, the classification, and hence, color anomaly detection, reduces to estimating $P(\vec{\mathcal{I}}[\boldsymbol{x}] = \vec{I}[\boldsymbol{x}]|\eta)$ and selecting an appropriate threshold.

3.2 Probability Distribution Estimation

We describe the use of k_n-nearest neighbor (kNN) algorithm in order to estimate $P(\vec{\mathcal{I}}[\boldsymbol{x}] = \vec{I}[\boldsymbol{x}]|\eta)$. It is explained in [12] that the density of a scatter plot may be used to estimate a distribution. The kNN density estimate of a feature vector $\vec{I}[\boldsymbol{x}]$ is given by:

$$p_n(\vec{I}[\boldsymbol{x}]) = \frac{k_n}{nV_n} \tag{3.2.1}$$

where k_n samples are the k_n nearest neighbors of $\boldsymbol{x}$, V_n is the volume of the smallest sphere containing all those samples in the feature space, and n is the total number of samples in the processing window for this case. It is stated that for $k_n \to \infty$ as $n \to \infty$ and $\frac{k_n}{n} \to 0$ as $n \to \infty$ the estimate converges to the real distribution of $\vec{I}[\boldsymbol{x}]$. The problem of applying this approach is that a training set, where all data are "normal", does not exist. Therefore, additional constraints are needed to make

this work with data mixed with both classes. We assume that the maximum number of anomalous samples in a given processing window does not exceed N. Therefore, by selecting $k_n > N - 1$, the effect of anomalous data can be avoided, ensuring that $p_n(\vec{I}[\boldsymbol{x}])$ will be a reliable estimate of $P(\vec{\mathcal{I}}[\boldsymbol{x}] = \vec{I}[\boldsymbol{x}]|\eta)$.

In order to illustrate the intuition behind this method, we synthetically generated an example with two classes: class 1 and class 2; which represent the "normal" and "anomalous" class, respectively. Since "anomalous" points should be different from "normal" points, class 1 and 2 were constructed to be distinct from each other in feature space. Figure 1 shows a scatter plot of feature samples from both classes. The number of feature samples is 5 for class 2, corresponding to $N = 5$, and 100 for class 1, which can be interpreted as feature points from a 10×10 processing window. Each circle represents the smallest circle centered at a feature point of interest that contains k_n samples. Note that the radii of these circles are inversely proportional to the probability distribution values of the feature points of interest. For $k_n = 4(= N - 1)$, both circles have similar radii, i.e. the estimated probability distribution is affected by the samples of class 2 and has a peak in the region of the "anomalous" features. However, when $k_n = 5(> N - 1)$, the radius of the circle around the pink ("anomalous") point of interest increases dramatically. Therefore, the estimated probability distribution will have low values in that region, resulting in the estimate of $p_n(\vec{I}[\boldsymbol{x}])$ to be close to the estimate of $P(\vec{\mathcal{I}}[\boldsymbol{x}] = \vec{I}[\boldsymbol{x}]|\eta)$.

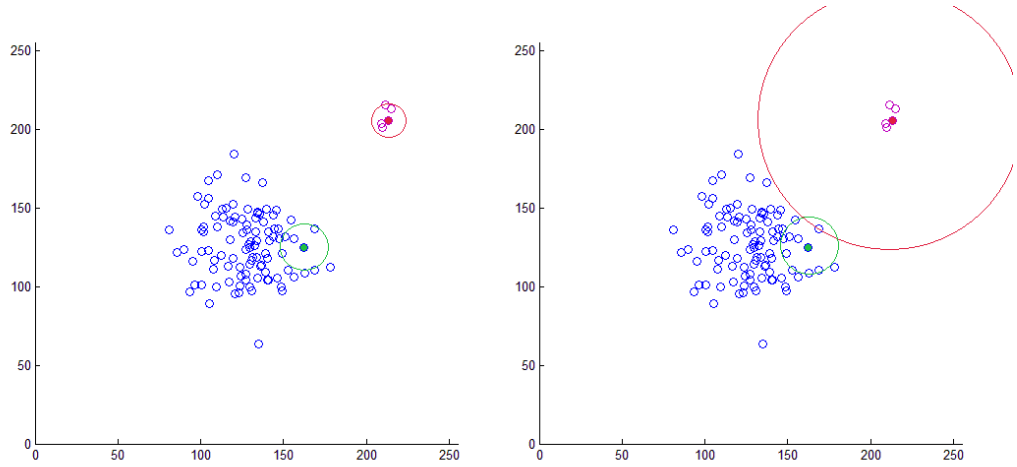

Figure 1: Scatter plots of synthetically generated features from 2 classes. Class 1 and 2 are represented by blue and pink dots, respectively. Solid dots represent feature points of interest.

3.3 Model with Spatial Correlation

In section 3.1, a binary hypothesis testing method is derived for color anomaly detection in the case of spatially independent pixels. For the second stage of our approach,

reformulation of the binary hypothesis testing is required in order to take spatial correlation between the pixels into account. Start by defining a label field E with following possible realizations: $e[\boldsymbol{x}] = 0$ if a pixel at $\boldsymbol{x}$ is classified "normal"and $e[\boldsymbol{x}] = 1$ if it is classified "anomalous". This label field will be modeled as a MRF, which is similar to the approach used in [11]. Assuming that the label field is known for every pixel except for the pixel of interest, and e_η is the label field with $e[\boldsymbol{x}] = 0$ and e_ξ is the label field with $e[\boldsymbol{x}] = 1$, the hypothesis test stated in equation (3.1.1) may be restated in the following way:

$$\frac{P(\vec{\mathcal{I}} = \vec{I}|e_\eta)}{P(\vec{\mathcal{I}} = \vec{I}|e_\xi)} \underset{\xi}{\overset{\eta}{\gtrless}} c \cdot \frac{P(E = e_\xi)}{P(E = e_\eta)} \tag{3.3.1}$$

If we assume that color components at each location are conditionally independent given the label field, and the label field realizations $e[\boldsymbol{y}]$ are known for all $\boldsymbol{y} \neq \boldsymbol{x}$, then equation (3.3.1) can be rewritten as (3.3.2). By eliminating common terms and assuming $P(\vec{\mathcal{I}}[\boldsymbol{x}] = \vec{I}[\boldsymbol{x}]|\xi)$ to be constant, the expression becomes (3.3.3).

$$\frac{\prod_{\boldsymbol{y}\neq\boldsymbol{x}} P(\vec{\mathcal{I}}[\boldsymbol{y}] = \vec{I}[\boldsymbol{y}]|e[\boldsymbol{y}])}{\prod_{\boldsymbol{y}\neq\boldsymbol{x}} P(\vec{\mathcal{I}}[\boldsymbol{y}] = \vec{I}[\boldsymbol{y}]|e[\boldsymbol{y}])} \cdot \frac{P(\vec{\mathcal{I}}[\boldsymbol{x}] = \vec{I}[\boldsymbol{x}]|\eta)}{P(\vec{\mathcal{I}}[\boldsymbol{x}] = \vec{I}[\boldsymbol{x}]|\xi)} \underset{\xi}{\overset{\eta}{\gtrless}} c \cdot \frac{P(E = e_\xi)}{P(E = e_\eta)} \tag{3.3.2}$$

$$P(\vec{\mathcal{I}}[\boldsymbol{x}] = \vec{I}[\boldsymbol{x}]|\eta) \underset{\xi}{\overset{\eta}{\gtrless}} \gamma \cdot \frac{P(E = e_\xi)}{P(E = e_\eta)} \tag{3.3.3}$$

The Hammersley-Clifford theorem states that a MRF can be represented as a Gibbs random field. Since the label for each location $\boldsymbol{x}$ is assumed to be drawn from a MRF, the $P(E = e)$ terms in equation (3.3.1) can be written as:

$$P(E = e) = \frac{1}{Z} exp\{-\frac{1}{T} Q(e)\} \tag{3.3.4}$$

which is a Gibbs distribution with temperature T, normalizing constant Z, and energy function $Q(e)$. The energy function is defined as the summation of potential function V_c over cliques c belonging to the set of all cliques C ($Q(e) = \sum_{c\in C} V_c(e)$). We use 2-element cliques in a second-order neighborhood (8 immediate neighbors), which is commonly used in image processing applications. One could increase the neighborhood size or use cliques with more than 2 elements at the cost of computational complexity. The potential function was chosen in order to penalize the pixels whose labels do not match to their neighbors'. The label field E is binary; therefore, we decided to use the Ising potential stated in equation (3.3.5).

$$V_{ce}(\boldsymbol{x}, \boldsymbol{y}) = \begin{cases} 0 & e[\boldsymbol{x}] = e[\boldsymbol{y}] \\ 1 & otherwise \end{cases} \tag{3.3.5}$$

Since e_ξ and e_η differ only at location $\boldsymbol{x}$, the potential function evaluated on all cliques not containing the pixel at $\boldsymbol{x}$ are be the same, and the ratio of a priori probabilities can be written as:

$$\frac{P(E=e_\xi)}{P(E=e_\eta)} = \frac{\frac{1}{Z}exp\{-\frac{1}{T}\sum_{\boldsymbol{y}=\boldsymbol{x}or\boldsymbol{z}=\boldsymbol{x}} V_{ce_\xi}(\boldsymbol{y},\boldsymbol{z})\}}{\frac{1}{Z}exp\{-\frac{1}{T}\sum_{\boldsymbol{y}=\boldsymbol{x}or\boldsymbol{z}=\boldsymbol{x}} V_{ce_\eta}(\boldsymbol{y},\boldsymbol{z})\}} \tag{3.3.6}$$

After adding the Ising potential and cancelling the normalizing constants, the equation becomes:

$$\frac{P(E=e_\xi)}{P(E=e_\eta)} = \frac{exp\{\frac{1}{T}G(\xi)\}}{exp\{\frac{1}{T}G(\eta)\}} \tag{3.3.7}$$

where G(ξ) is the number of "anomalous" pixels and G(η) is the number of "normal" pixels in a neighborhood. Finally, substituting this term to equation (3.3.3) produces the following:

$$P(\vec{\mathcal{I}}[\boldsymbol{x}] = \vec{I}[\boldsymbol{x}]|\eta) \underset{\xi}{\overset{\eta}{\gtrless}} \gamma \cdot exp\{\frac{1}{T}(G(\xi) - G(\eta)\} \tag{3.3.8}$$

Expression (3.3.8) represents the final decision rule used for classification.

3.4 RX Detector

Thus far, a promising method for color anomaly detection has been developed. In this section, the RX detector, an established anomaly detector that will be used for comparison, will be explained.

The formulation of the measure calculated by the RX detector follows the same assumptions mentioned in section 3.1. Rather than estimating the probability distribution, the method assumes that $P(\vec{\mathcal{I}}[\boldsymbol{x}] = \vec{I}[\boldsymbol{x}]|\eta)$ is a multivariate Gaussian distribution with mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{C}$. From this, equation (3.1.2) can be simplified by taking the logarithm of the entire expression.

$$\begin{aligned}\log\{P(\vec{\mathcal{I}}[\boldsymbol{x}] = \vec{I}[\boldsymbol{x}]|\eta)\} = \\ -\frac{m}{2}\log(2\pi) - \frac{1}{2}\log|\boldsymbol{C}| - \frac{1}{2}(\vec{I}[\boldsymbol{x}] - \boldsymbol{\mu})^T\boldsymbol{C}^{-1}(\vec{I}[\boldsymbol{x}] - \boldsymbol{\mu}) \underset{\xi}{\overset{\eta}{\gtrless}} \log(\alpha)\end{aligned} \tag{3.4.1}$$

By grouping all the constants into a single threshold term, equation (3.4.1) becomes:

$$(\vec{I}[\boldsymbol{x}] - \boldsymbol{\mu})^T\boldsymbol{C}^{-1}(\vec{I}[\boldsymbol{x}] - \boldsymbol{\mu}) \underset{\eta}{\overset{\xi}{\lessgtr}} \varphi \tag{3.4.2}$$

A rigorous derivation of this equation can be found in [13]. The term on the left side of equation (3.4.2) can be interpreted as the Mahalanobis distance between realization $\vec{I}[\boldsymbol{x}]$ and mean vector $\boldsymbol{\mu}$. The RX algorithm computes this term at every $\boldsymbol{x}$. A caveat with this approach is that the true values of second-order statistics $\boldsymbol{\mu}$ and

$\boldsymbol{C}$ are unknown; therefore, these parameters must be estimated from samples. The estimates of the mean and covariance are denoted by $\hat{\boldsymbol{\mu}}$ and $\hat{\boldsymbol{C}}$, respectively.

For local anomaly detection, a sliding window is used to compute $\hat{\boldsymbol{\mu}}$ and $\hat{\boldsymbol{C}}$ from nearby pixels. The design of the window is shown in Figure 2. The estimated statistics for a pixel of interest are calculated from pixels within the outer window excluding those contained within the guard window. The guard window is incorporated in order to avoid inclusion of pixels that belong to "anomalous" regions. Therefore, the size of the guard window should exceed the area of "anomalous" pixels. Moreover, the outer window must be large enough to produce estimates that represent the true mean and covariance.

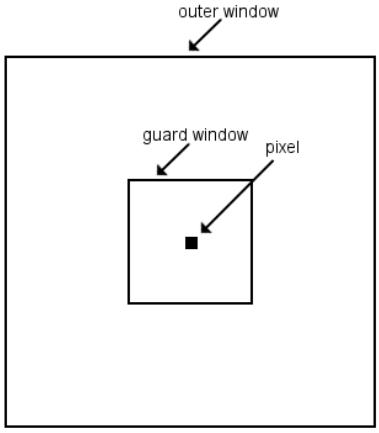

Figure 2: Sliding window used by the RX detector

4 Implementation

4.1 Data Set Construction

There is no publicly available dataset for search and rescue images; therefore, it was necessary to construct a set of images that simulate a search and rescue scenario. These images consist of a large scenic background with smaller foreground objects.

The background images were selected considering their spatial resolution and angle: for a fixed pixel resolution, spatial resolution should be enough to represent a human-sized object with an appropriate number of pixels; on the other hand, this resolution should not be too great such that objects become large and detection by a human observer is trivial. One thing to mention is that despite our selectivity in which background images to use, there is still high variance in altitude/spatial resolution and viewing angle in the data set. This is acceptable, since the resolution of video and altitude of the vehicles vary for different search and rescue operations.

Inclusion of the foreground images is performed as follows: three jacket images

were cropped and placed at random location on the background image. The saturation and brightness values of these cropped images were modified such that they blend in with a more natural look. Since both anomaly detection methods are sensitive to the size of the anomalies, cropped images were scaled down to a size of $\approx 10 \times 10$. Ground truth images were constructed where all foreground objects are white and background is black.

4.2 Classification

Our detection algorithm can be divided into two parts: probability distribution estimation and thresholding. First, the image is divided into 50×50 non-overlapping blocks. For each block, the probability estimate described by equation (3.2.1) is calculated for every pixel within the block. Since the maximum anomaly size is assumed to be $\approx 10 \times 10$, using k_n of 100 satisfies the criterion, $k_n > N-1$ described in section 3.2.

Once probability distribution values are available for each pixel, equation (3.1.2) is used for a fixed threshold to obtain the label result for the first stage of the method. For the second stage of the method, equation (3.3.8) is implemented iteratively. The label field is initialized with the resulting label image of the first stage, then, at each step, the threshold term of each pixel is calculated. The next label image is found by thresholding each pixel by their corresponding thresholds. We found that 6 iterations were enough for convergence in our dataset.

For the RX detector, a sliding window with 51×51 outer window and 15×15 guard window was used.

All algorithms were implemented in MATLAB.

5 Results

Objective evaluation of detection performance was performed by calculating a receiver operating curves (ROC) for each detector and using the area under ROC curve (AUC) as a metric. The ROC curve is a plot of the statistical measures true positive rate (TPR) versus false positive rate (FPR). The TPR is the proportion of anomalous pixels classified as anomalous to the total number of anomalous pixels. The FPR is the proportion of normal pixels classified as anomalous to the total number of normal pixels.

Detector	Area Under Curve
kNN without MRF	0.9865
RX	0.9727
kNN with MRF	0.9909

Table 1: Area Under Curve Comparison

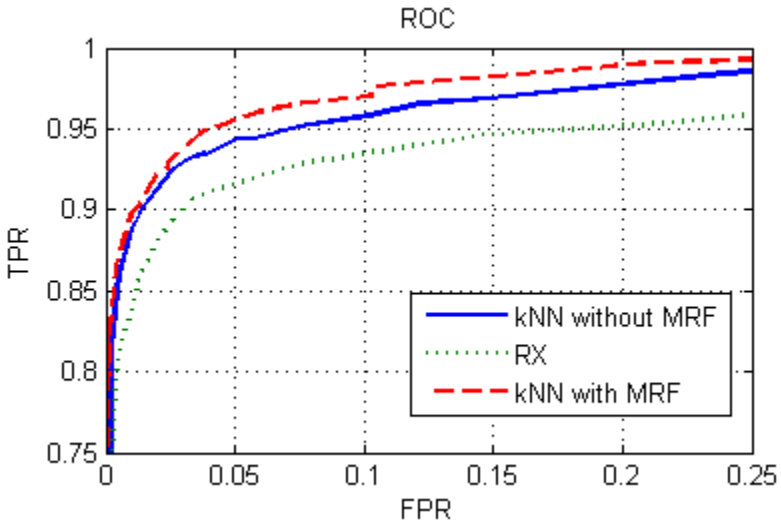

Figure 3: ROC curves for tested detection algorithms

Table 1 and Figure 3 show the results from our evaluation. Our detector obtains a higher TPR than the RX detector for any fixed FPR. In addition, the AUC for our detector with MRF is 0.0182 higher than the RX detector. Thresholds of 2×10^{-4} and 20-30 were found to be visually the best for our method and the RX detector, respectively.

Figure 4 illustrates the benefit of incorporating the MRF modeling into our method. The resulting label image without MRF modeling has many false positives. After the addition of MRF modeling, majority of these false positives were removed while retaining the true positives. Although the performance improvement was not identical for every image, we have not observed any images where addition of MRF decreased the performance. This beneficial effect is also apparent in the ROC curves, where there is an increase in the AUC of 44×10^{-4}

Some outputs of the detectors are given in Figures 5 and 6. It is apparent that the false positive rate of the RX algorithm is higher on the average. The RX algorithm tends to produce a large number of false positives with small areas. This stems from the fact that it is a point anomaly detector. Since our method takes spatial correlation into account, the resulting false alarms are more concentrated. Having false positives concentrated is more favorable for automated target suggestion since having too many scattered suggestions could prove to distract, rather than assist, some human operator. In terms of speed, our algorithm is much faster due to its implementation using non-overlapping blocks. The RX detector algorithm relies on a sliding window approach, which requires computation at every pixel.

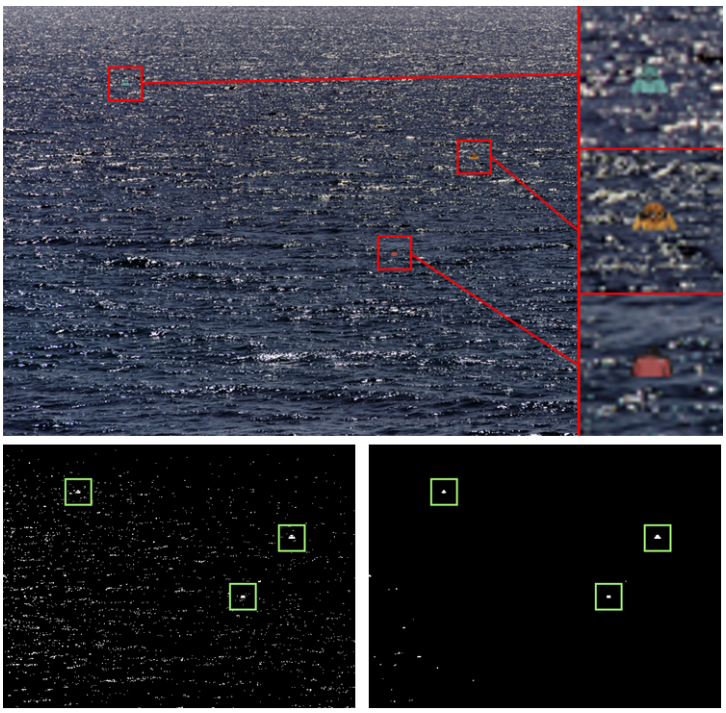

Figure 4: Effect of MRF Modeling. (Top) Original image: "anomalous" objects are denoted by red squares (Bottom) Output image with our detector without (left) and with (right) MRF modeling

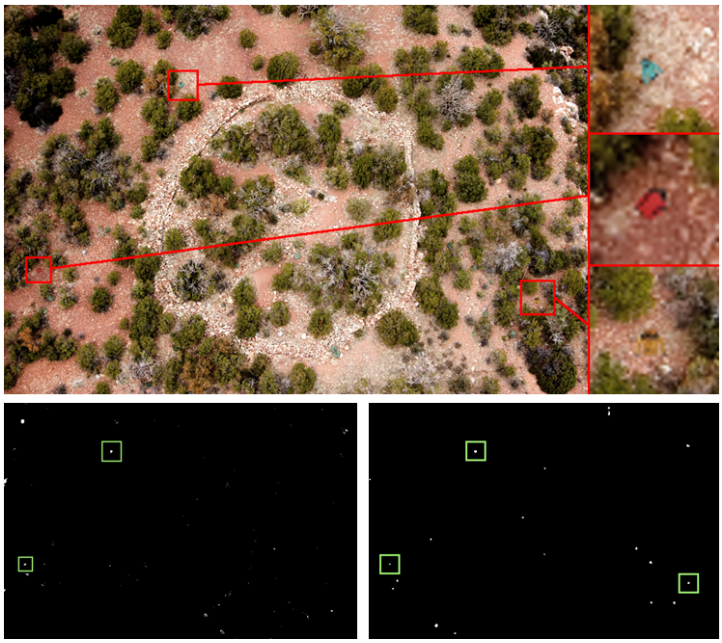

Figure 5: Detection Comparison I.(Top) Original image: "anomalous" objects are denoted by red squares (Bottom) Output image with the RX detector (left) and our detector (right)

After comparing our detector on a synthetically generated dataset, we tested it on an image taken from a real search and rescue operation. As can be seen in Figure 7, the detector is able to find the object of interest, which could easly be misinterpreted as a wave.

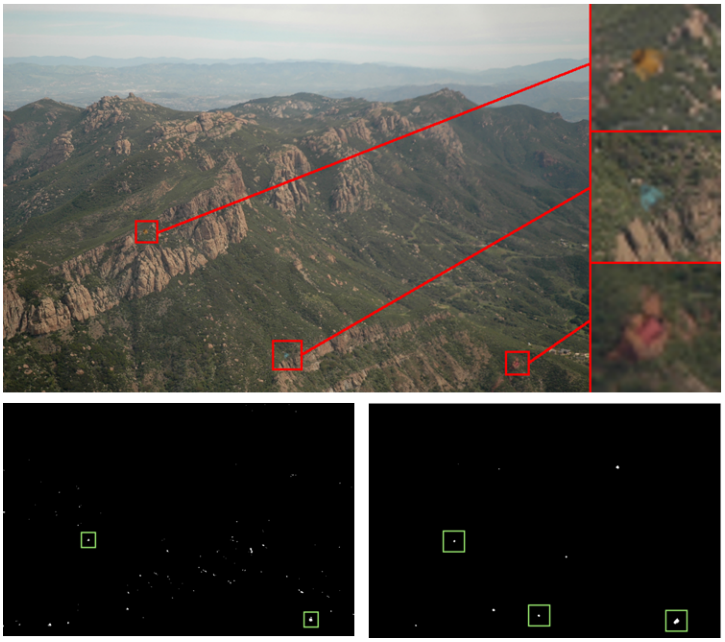

Figure 6: Detection Comparison II.(Top) Original image: "anomalous" objects are denoted by red squares (Bottom) Output image with the RX detector (left) and our detector (right)

Figure 7: Detection on a real search and rescue image. (Left) Original image (Right) Output image with the our detector

6 Conclusions and Future Work

We have proposed a new color anomaly detector in this report, and it has performed well based on objective and subjective measures and time complexity compared to the RX method. However, there is still room for improvements. First, the variance of "normal" data had an effect on threshold selection and, hence, detection efficiency. One could account for this by implementing an automated method for threshold adjustment. We also observed that the detection algorithm had difficulty detecting white or black outliers. The most probable reason of this is the exclusion of the $L*$ component from the feature vector. One could modify the methodology to account for these extremas in luminance. In addition, we used non-overlapping blocks, which leads to false positives when there is a sharp change in the background (like sea/land border), since the block fails to capture enough samples from one part of the background. To account for this using overlapping blocks and keeping the anomalies that occur on every block for overlapping regions may be considered.

References

[1] A. Mecocci, M. Pannozzo and A. Fumarola, "Automatic detection of anomalous behavioural events," *Symposium on Computational Intelligence for Measurement Systems and Applications* , pp. 187-192, July 2003.
[2] I. Reed and X. Yu, "Adaptive multiple-band CFAR detection of an optical pattern with unknown spectral distribution," *IEEE Transactions on Acoustic, Speech and Signal Processing*, vol. 38, no. 10, pp. 1760-1770, 1990.
[3] T. Smetek and K. Bauer, "Finding Hyperspectral Anomalies Using Multivariate Outlier Detection," *IEEE Aerospace Conference*, pp. 1-24, 2007.
[4] D. Borghys, V. Achard, S. R. Rotman, N. Gorelik, C. Perneel and E. Schweicher, "Hyperspectral anomaly detection: A comparative evaluation of methods," *General Assembly and Scientific Symposium, 2011 XXXth URSI*, pp. 1-4, 2011.
[5] H. Kwon, S. Z. Der and N. M. Nasrabadi, "Adaptive anomaly detection using subspace separation for hyperspectral imagery," *Optical Engineering*, vol. 42, no. 11, pp. 3342-3351, 2003.
[6] M. Carlotto, "A cluster-based approach for detecting man-made objects and changes in imagery," *IEEE Transaction Geoscience and Remote Sensing*, vol. 43, no. 2, pp. 374-387, 2005.
[7] B. S. Morse, D. Thornton and M. A. Goodrich, "Color Anomaly Detection and Suggestion for Wilderness Search and Rescue," *in 7th ACM/IEEE International Conference on Human-Robot Interaction*, Boston, MA, 2012.
[8] A. P. Dempster, N. M. Laird and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the Royal Statistical Society. Series B (Methological)*, vol. 39, no. 1, pp. 1-38, 1977.
[9] V. Chandola, A. Banerjee and V. Kumar, "Anomaly detection: A survey," *ACM Computing Surveys (CSUR)*, vol. 41, no. 3, 2009.
[10] G. G. Hazel, "Multivariate Gaussian MRF for multispectral scene segmentation and anomaly detection," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 38, no. 3, pp. 1199-1211, 2000.
[11] J. McHugh, J. Konrad, V. Saligrama and P. -M. Jodoin, "Foreground-adaptive background subtraction," *IEEE Signal Processing Letters*, vol. 16, pp. 390-393, 2009.
[12] J. Tohka, "k-Nearest Neighbors Classifier," in Introduction to Pattern Recognition, Tampere University of Technology. Institute of Signal Processing, 2006-2008, pp. 49-50.
[13] L. R. Bachega, J. Theiler and C. A. Bouman, "Evaluating and improving local hyperspectral anomaly detectors," *AIPR*, pp. 1-8, 2011.

7 Appendix: MATLAB Code

```
%init
clear all

%thresholds and parameters
thr_nomrf=1*10^-4;
thr_mrf=1*10^-4;
g_temp = 1;

%variables
label = cell(1,7);
%read image
imrgb=imread('sample_17.tif');
%remove alpha channel if exists
if size(imrgb,3)>3
    imrgb=imrgb(:,:,1:3);
end
%convert it to Lab space
imLab=rgb2lab(imrgb);
%estimate likelihoods (currently using a and b only)
fun=@knnpdf;
B = blockproc(imLab,[50 50],fun, 'UsePparallel', 1 , '
    PadPartialBlocks', 1,'PadMethod', 'symmetric');
%label image anomalies=1
size_im = size(imrgb);
im_prob = B(1:size_im(1),1:size_im(2));
label{1}=im_prob<thr_nomrf;
%MRF model
n_mask = ones(3);
n_mask(2,2) = 0;
for i = 1:6
    anom_normal_cnt = -8 + 2.*conv2(double(label{i}),n_mask,'
        same');   %Qf- Qb
    label{i+1}=im_prob<thr_mrf.*exp(1/g_temp.*anom_normal_cnt
        );
end
%
```

```
function imLab = rgb2lab(imrgb)
 cform = makecform('srgb2lab');
 imLab = applycform(imrgb, cform);
end
```

```
function P = knnpdf(block)

size = block.blockSize;

%variables
kn = 100;
n = size(1)*size(2);
%
features = zeros( n, 3);
for i = 2:3
temp = block.data(:,:,i);
features(:,i) = double(temp(:));
end
X = features;
Y = features;
[IDX,D] = knnsearch(X,Y, 'K', kn); %k=25 for this case
P = kn./(n.*(pi.*(D(:,kn)+1).^2));
%P = reshape(D(:,25),size(1), size(2));
P = reshape(P, size(1), size(2));
end
```

```
function rx = rxdetectblock(block)

r = 7;                              %guard window/# of points
    away from the center
cent = block.border+1;              %center of block
data = double(block.data);
x = data(cent(1),cent(2),:);        %features of center pixel

% Remove center region/window
data(cent(1)-r:cent(1)+r,cent(2)-r:cent(2)+r,:) = NaN;
f = size(data,3);

% Find statistics from region out of inner window
temp = data(~isnan(data));
featureval = zeros(1,length(temp)/f,f);

% Save features of non-inner window pixels
for ii = 1:f
    featureval(:,:,ii) = temp(1+(length(temp)/f)*(ii-1):(
        length(temp)/f)*ii);
end

% Calculate second-order statistics
```

```
mu = mean(featureval);
C = cov(squeeze(featureval));

% Find measurement
rx = (squeeze(x-mu)'/C)*squeeze(x-mu);
if isnan(rx)
    rx = 0;
end
```

```
function label = MRFupd(im_prob, label, ht_th, total_iter,
    gibbs_tmp)

n_mask = ones(3);
n_mask(2,2) = 0;
label = double(label);

for i = 1:total_iter
    anom_normal_cnt = -8 + 2.*conv2(label,n_mask,'same');    %
        Qf- Qb
    label=double(im_prob<ht_th.*exp(1/gibbs_tmp.*
        anom_normal_cnt));
end

end
```

```
%init
clear all
%thresholds and parameters
TH = [0, 10^-4:0.5*10^-4:100*10^-4, Inf];
mrf_temp = 1;
num_MRF_iter = 6;
blk_sz = 50;

%read images and ground truths to workspace
namesGT = dir('*GT.tif');
imagesrgb = cell(length(namesGT),1);
imagesLab = cell(length(namesGT),1);
imageGTs = cell(length(namesGT),1);
imagePrs = cell(length(namesGT),1);
for i = 1:length(namesGT)
    imageGT_name = namesGT(i).name;
    image_name = [imageGT_name(1:end-6),'.tif'];
    try
        imagesrgb{i} = imread(image_name);
```

```
20      catch
21          sprintf('Some images do not exist in the folder')
22          return
23      end
24      %remove alpha channel if exists
25      im_temp = imagesrgb{i};
26      imagesrgb{i} = im_temp(:,:,1:3);
27      %convert to Lab space
28      imagesLab{i} = rgb2lab(imagesrgb{i});
29      imageGTs{i} = imread(imageGT_name);
30      im_temp = imageGTs{i};
31      imageGTs{i} = im_temp(:,:,1);
32  end
33  for i = 1: length(imagesLab)
34
35      fun=@knnpdf;
36      B = blockproc(imagesLab{i},[blk_sz blk_sz],fun, '
            UseParallel', 1 , 'PadPartialBlocks', 1,'PadMethod', '
            symmetric');
37      size_im = size(imagesLab{i});
38      imagePrs{i} = B(1:size_im(1),1:size_im(2));
39  end
40
41  %for every threshold calculate ROC point
42  ROC = zeros(length(TH),2);
43  PR = zeros(length(TH),2);
44  for j = 1: length(TH)
45      [ROC(j,:),PR(j,:)] = perf_iterv2(imagesLab,imageGTs,
            imagePrs,[TH(j),mrf_temp,num_MRF_iter],'pixelwise');
46  %%%%%%%%%%%%%%%%%%ignore ROC if set to centroid
        %%%%%%%%%%%%%%%%%%%%%%
47  end
48  figure,plot(ROC(:,2),ROC(:,1));
49  area_under_ROC=trapz(ROC(:,2),ROC(:,1));
```

```
1  %init
2  clear all
3  %thresholds and parameters
4  TH = [0:1:1000];
5  outer_blk_size = 25;
6
7  %read images and ground truths to workspace
8  namesGT = dir('*GT.tif');
9  imagesrgb = cell(length(namesGT),1);
```

```
imagesLab = cell(length(namesGT),1);
imageGTs = cell(length(namesGT),1);
imagePrs = cell(length(namesGT),1);
for i = 1:length(namesGT)
    imageGT_name = namesGT(i).name;
    image_name = [imageGT_name(1:end-6),'.tif'];
    try
        imagesrgb{i} = imread(image_name);
    catch
        sprintf('Some images do not exist in the folder')
        return
    end
    %remove alpha channel if exists
    im_temp = imagesrgb{i};
    imagesrgb{i} = im_temp(:,:,1:3);
    %convert to Lab space
    imagesLab{i} = rgb2lab(imagesrgb{i});
    imageGTs{i} = imread(imageGT_name);
    im_temp = imageGTs{i};
    imageGTs{i} = im_temp(:,:,1);
end
for i = 1: length(imagesLab)

    fun=@rxdetectblock;
    B = blockproc(imagesLab{i},[1 1],fun ,'BorderSize',[
        outer_blk_size, outer_blk_size],...
    'PadPartialBlocks', 1,'PadMethod', 'symmetric','
        TrimBorder',false);
    size_im = size(imagesLab{i});
    imagePrs{i} = B(1:size_im(1),1:size_im(2));
end
%for every threshold calculate ROC point
ROC = zeros(length(TH),2);
PR = zeros(length(TH),2);
for j = 1: length(TH)
    [ROC(j,:),PR(j,:)] = perf_iter_RX(imagesLab,imageGTs,
        imagePrs,[TH(j)],'pixelwise');
%%%%%%%%%%%%%%%%%%ignore ROC if set to centroid
    %%%%%%%%%%%%%%%%%%%%%%%
end
figure,plot(ROC(:,2),ROC(:,1));
```

```
function [ROC,PR] = perf_iterv2(imagesLab,imageGTs,imagePrs,
    parameters,type)
```

```
%Inputs: imagesLab cell containing Lab images
%        imageGTs cell containing corresponding ground truth results
%        parameters: vector [Hypothesis test threshold, Gibbs temperature,
%                            Total number of MRF iterations, block size]
%Output: Receiver operating characteristics [sensitivity, specificity]
PR = zeros(1,2);
ROC = zeros(1,2);
num_of_images =length(imagesLab);
ht_th = parameters(1);
gibbs_tmp = parameters(2);
total_iter = parameters(3);

TP = zeros(num_of_images,1);
TN = zeros(num_of_images,1);
FP = zeros(num_of_images,1);
FN = zeros(num_of_images,1);

for i = 1: num_of_images
    %initial label field
    label_init=imagePrs{i} < ht_th;
    %MRF updates
    label = MRFupd(imagePrs{i}, label_init, ht_th, total_iter, gibbs_tmp);
    [TP(i) TN(i) FP(i) FN(i)] = perf_meas(imageGTs{i},label, type);
end
PR(1) = sum(TP) / (sum(TP) + sum(FP));
PR(2) = sum(TP) / (sum(TP) + sum(FN));
ROC(1) = sum(TP) / (sum(TP) + sum(FN));
ROC(2) = 1 - (sum(TN) / (sum(TN) + sum(FP)));
end
```

```
function [TP TN FP FN] = perf_meas(GT,R,type)
%Inputs:
%GT: Ground truth binary image with 1s corresponding to class1 and
%0s corresponding to class2
%R : Result binary image with 1s corresponding to class1 and
%0s corresponding to class2
```

```
%Outputs True positive True Negative False Positive False Negative
min_area=9; %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
max_area= 5000;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
GT = logical(GT);
R = logical(R);
    switch type
        case 'pixelwise'
            GT = GT(:);
            R = R(:);
            TP = sum(GT & R);
            TN = sum(~(GT | R));
            FP = sum(~GT & R);
            FN = sum(GT & ~R);
        case 'centroid'
            TN = 1;
            TP = 0;
            FN = 0;
            connR = bwconncomp(R);
            statsR = regionprops(connR, 'Centroid','Area');
            for i = 1:length(statsR)
                if(statsR(i).Area>min_area && statsR(i).Area<max_area)
                    addr = round(statsR(i).Centroid);
                    TP = TP + GT(addr(2),addr(1));
                end
            end
            FP = length(statsR) - TP;
            connGT = bwconncomp(GT);
            statsGT = regionprops(connGT, 'Centroid');
            for i = 1:length(statsGT)
                addr = round(statsGT(i).Centroid);
            FN = FN + (1-R(addr(2),addr(1)));
            end

       otherwise
            sprintf('Invalid function type')
    end
end
```