Using evolutionary computation to optimize an SVM used in detecting buried objects in FLIR imagery

Alex Paino, Mihail Popescu, James M. Keller, Kevin Stone
atpgh6@missouri.edu, popescum@mail.missouri.edu, kellerj@mail.missouri.edu, kes25c@missouri.edu
Electrical and Computer Engineering Dept.
University of Missouri,
Columbia, Missouri, USA

ABSTRACT

In this paper we describe an approach for optimizing the parameters of a Support Vector Machine (SVM) as part of an algorithm used to detect buried objects in forward looking infrared (FLIR) imagery captured by a camera installed on a moving vehicle. The overall algorithm consists of a spot-finding procedure (to look for potential targets) followed by the extraction of several features from the neighborhood of each spot. The features include local binary pattern (LBP) and histogram of oriented gradients (HOG) as these are good at detecting texture classes. Finally, we project and sum each hit into UTM space along with its confidence value (obtained from the SVM), producing a confidence map for ROC analysis. In this work, we use an Evolutionary Computation Algorithm (ECA) to optimize various parameters involved in the system, such as the combination of features used, parameters on the Canny edge detector, the SVM kernel, and various HOG and LBP parameters. To validate our approach, we compare results obtained from an SVM using parameters obtained through our ECA technique with those previously selected by hand through several iterations of "guess and check".

Keywords*:* forward looking infrared imaging; FLIR; IR; SVM; mathematical morphology; HOG; LBP; evolutionary computation

1 INTRODUCTION

Detection of buried targets continues to represent a challenging problem and multiple sensing modalities such as ground penetrating radar, metal detectors and infrared (IR) have been employed to address it. Among the variety of sensors used to detect buried objects, IR is gaining in popularity due to the recent advances in un-cooled camera technology and multiband sensors.

Infrared cameras have been used in a variety of applications such as ground target recognition [1, 2], flying target tracking [3], remote sensing [4] and landmine detection [5-9]. IR object detection is based on the temperature difference between the target and the surrounding background. More specifically, the detection of buried objects is based on differences in soil texture, spectral composition and temperature between the area above the target and background [5].

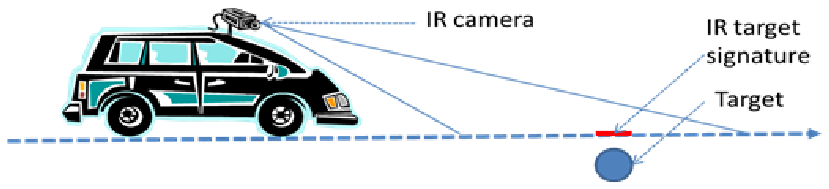
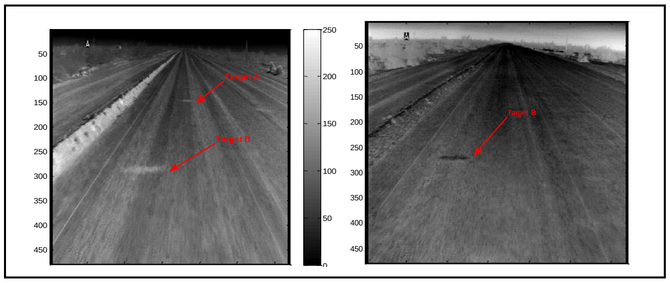


Figure 1. LWIR camera mounted in front of a moving vehicle to detect buried targets based on their IR signature

In this paper we use a long wave IR (LWIR) camera installed on a vehicle (see Fig. 1) to detect objects buried in the road ahead. The vehicle drives on a road that has buried targets which exhibit certain IR signatures (see Fig. 2 for example images). Our goal is to cue the driver for possible buried objects of interest. While our ultimate goal is to cue the driver in real time, in this paper we present only a retrospective (offline) IR video processing methodology.



a) b)

Figure 2. Typical vehicle mounted IR images taken at different time of the day: a) at 11 am, b) at 7 am

In previous papers [7-9] we presented several approaches to solve this problem. Our most recent effort [16] involved implementing a temporal fusion approach that projected hits in the Universal Traverse Mercator (UTM) world coordinates, where the hits are found by a spot-finding procedure on the LWIR image and classified by a Support Vector Machine (SVM). An example of the output from the temporal fusion approach can be seen in Fig. 4. Adding the temporal fusion aspect to our algorithm allowed us to achieve a significant performance increase in terms of the probability of detection of our algorithm. However, up until now we selected parameters to various algorithms via a "guess and check" method, or by settling for the default arguments. Thus, we felt that we may be able to improve our algorithm by optimizing these parameters in a systematic, automatic way. As the algorithm has grown in complexity, so too has the number of parameters it requires. When combined with the complexity of our algorithm and the significant computational power it requires to run, the large number of parameters ruled out using a more traditional parameter optimization approach such as grid search. Thus, we had to develop another method to optimize them.

We decided to use an Evolutionary Computation Algorithm (ECA) for this task, as it is an efficient way to optimize a complex algorithm with many inter-related parameters, such as ours. We break our optimization process up into two parts: one for parameters related to the spot-finding procedure, and one for the parameters related to the Support Vector Machine (SVM) classifier and Local Binary Pattern (LBP) and Histogram of Oriented Gradients (HOG) feature extraction algorithms. Both of these segments utilize the same base algorithm, but with different objective functions.

The rest of the paper is structured as follows: in section 2 we present the available dataset, in section 3 we describe the entire algorithm used for buried target detection, in section 4 we describe the evolutionary computation algorithm used to automatically optimize parameters to our algorithm, in section 5 we discuss results obtained on the available datasets and in section 6 we provide conclusions.

2 AVAILABLE DATASET

The data used in this paper was collected at an arid United States Army Test Site with a LWIR camera installed on a vehicle (see Fig. 1). The 1200 meter lane had 50 buried targets. We collected data from 7 runs on the lane. Typical frames from the collected movies are shown in Fig. 2. We can see two targets in Fig. 2.a and one target in Fig. 2.b. It is interesting to note that, since the two frames were taken at different time of the day, not only the target signatures differ between them but so also does the sky and the left roadside berm appearance.

Each run had a frame level ground truth, that is the (x,y) location in each frame of each of the 50 buried targets. If the detected spot (see next section) in a given frame had a ground truth located in its bounding box, it was declared a hit; otherwise it was declared a false alarm. The UTM location of the 50 buried targets was also known.

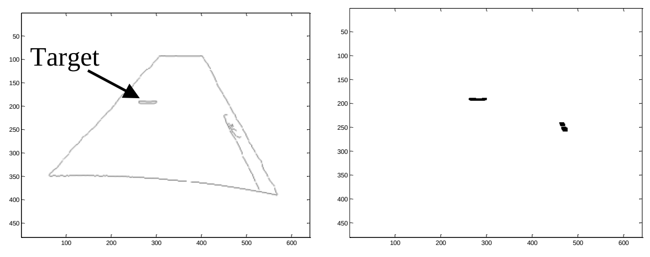
3 BURIED TARGET DETECTION ALGORITHM

The buried target detection algorithm has four parts: anomaly detection, classification, temporal fusion and scoring.

3.1 Prescreener Algorithm

The main steps of the prescreener algorithm are:

1. Find Edges. Apply Canny edge detector to each IR frame. Sample output for this step is shown in Fig. 3.a where a target can be seen. The trapezoidal shape present in Fig. 3.a is an artifact of the fixed road map used for spot search. A dynamic road map based on a road finding algorithm is presented in [12].



a. Canny edge detector b. Closing-Opening

Figure 3. Two steps of the spot finding algorithm

2. Close. Apply morphological closing with a 3×11 structuring element, SE.

Morphological binary image processing has two fundamental operations: erosion and dilation. If we denote by A the set of non-zero pixels of a binary image I, we can define the erosion of image I by structuring element (kernel) SE as the locus of the center of SE when SE itself moves inside A. Essentially, as a result of erosion a binary shape A present in I becomes "skinnier". Conversely, dilation of I by SE is defined as the locus of all point in SE when the center of SE moves inside A. Consequently, as a result of dilation, a binary shape A becomes "fatter". Using these two fundamental operations we can define other morphological operators such as closing, Closing(I, SE)=Erosion(Dilation(I, SE), SE), and opening, Opening(I, SE)=Dilation(Erosion(I, SE), SE). Note that while after erosion and dilation the size of the shape changes, after opening and closing the "desired" objects (targets in our case) should remain unchanged in size. Intuitively, closing has the role of closing holes smaller than SE, while opening has the role of removing objects smaller than SE. In our case, in step 2 we close all contours that might belong to targets. Note in Fig. 3.a how two closed contours were found: one produced by a target and another one by a false alarm.

3. Open. Apply morphological opening with the same SE to remove noise smaller than SE. Sample output is shown in Fig. 3.b. Note how the trapezoidal artifact and all other open contours (lines) were removed by opening.

4. Connected Components. Find the connected components in the image and compute various properties for each of them such as bounding box, centroid, orientation, etc. Of special importance is finding the bounding box that will next be used to compute the anomaly's features. This is an important property of the proposed algorithm and distinguishes this paper from [9] where the window used for feature computation was fixed and determined by trial-and-error.

3.2 Classification

The spot classification procedure has the following steps:

1. Feature extraction. For each alarm we computed local binary pattern (LBP) [13] and histogram of oriented gradients (HOG) feature.

LBP is a texture descriptor defined on P neighbors situated at radius R from a center pixel, $C=(x_c, y_c)$. If the center pixel C has a gray level denoted by g_c, then its $\mathrm{LPB}_{P,R}$ value is:

$$\mathrm{LBP}(C) = \sum_{i=0}^{P-1} s(g_i - g_c)2^i,\ s(x\geq 0)=1,\ s(x<0)=0, \quad (5)$$

where g_i are the gray levels of its P neighbors. In other words, the P neighbors are assigned a value of 0 or 1 depending on whether they are smaller or greater than C, respectively. The resulting bit pattern is then mapped into a number. In our case, we use R=1 and P=8 which results in possible LBP values between 0 and 255. To represent a hit by LBP features we simply compute the histogram of all the LBP values obtained for the pixels in its bounding box related to its centroid, C. If we used all LBP values, the histogram (hence the feature space dimension) will have 256 bins. However, if we only use the uniform LBP patterns (patterns that have at most two 0-1 or 1-0 transitions) from the 256 possible ones, the feature space reduces to dimension 59 (number used in this paper).

HOG is a texture descriptor that computes the occurrences of gradient orientation in a given image region. The gradient of the region is computed by filtering the image with a [-1,0,1] kernel to get the horizontal magnitude of the gradients and with a $[-1,0,1]^t$ for the vertical magnitudes. We begin by dividing the region into $H\times V$ cells and then creating a histogram with B bins for each cell. To compute the histograms, we first calculate the gradient for each pixel in the cell and then increment the bin corresponding to the orientation of the gradient by the magnitude of the gradient. The final HOG descriptor of the region is obtained by the concatenation of the histograms obtained in all cells. In our case we chose

H=3, *V*=3 and *B*=9 resulting in a HOG features vector of size 81 to describe each bounding box. The total number of features was 140.

2. Classifier. We use a leave-one-out procedure to train a support vector machine (SVM) to compute the confidence that a spot i is a target in frame j, $c_{ij} \in [0,1]$ $i \in [1,N_j]$ and $j \in [1,M]$ where N_j is the number of spots found in frame j and M and the total number of frames (between 4000 and 6000). More specifically, we train an SVM with the hits from 6 runs and then test the resulting model on the seventh run. We employed the MATLAB implementation of the LIBSVM library [15] in our experiments.

3.3 Temporal Fusion of the Image (frame) Alarms

After we compute the confidences c_{ij} that spots in frame j are targets, we project each spot together with its confidence in the UTM space.

The projection procedure is detailed in [7] and it is based on finding a mapping between the image space and UTM space. A typical output map of the projection procedure is shown in Fig. 4. The ground truth is marked with (red) circles whereas the bright areas denote aggregated spots. The grid used to produce the UTM image is 0.1m × 0.1m, which means that the image shown in Fig. 4 corresponds to a surface 6 m wide and about 1,200 m long.

3.4 Scoring Procedure

The scoring was performed on UTM maps similar to the one shown in Fig. 4. The main steps of the scoring procedures are:

1. Thresholding. Threshold the UTM map, O, with a value $T \in (\min(O), \max(O))$. For example, for the map shown in Fig. 4, typical thresholds could be $T \in (0, 200)$.

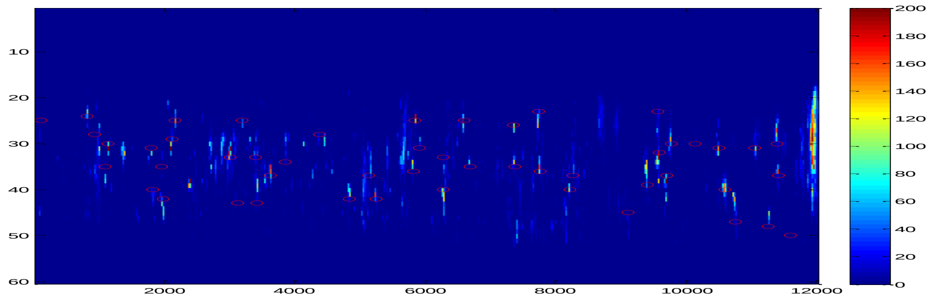


Figure 4. Typical UTM map output of the spot temporal fusion procedure (targets denoted by circles, intensity represents confidence)

2. Closing. Each UTM spot might have multiple local maxima that become disconnected when thresholding is applied. To reconnect the parts of the same spot we apply a closing procedure similar to the one from section III.A.2.

3. Connected Components. Apply a connected components procedure to the thresholded O, say NC UTM spots are detected.

4. Compute PD and FAR. Count hits, d, that are in a radius of HALO from the UTM ground truth location, and compute the probability of detection PD=d/50 and false alarm rate FAR=(NC-d)/Area(O).

5. Compute ROC. Compute a receiver operator characteristic (ROC) curve by choosing various thresholds T and repeating steps 2-4 given above.

4 EVOLUTIONARY COMPUTATION ALGORITHM

The task of optimizing parameters to our algorithm is broken down into two parts: one for optimizing LBP, SVM, and HOG parameters, and one for optimizing the Canny Edge detector parameters. This is so that we can perform the optimization of the parameters for the detection process separately from the optimization of the classification process parameters. The reason for breaking the optimization task down this way is to reduce the number of parameters being optimized by each ECA, allowing us to run each ECA for a lower number of generations before convergence occurs. Additionally, breaking up the optimization task greatly reduces the run time of the fitness function for each ECA, which is important to us because the algorithms were tested on a simple 3.0GHz Intel Core Quad system.

4.1 Structure of Evolutionary Algorithm

The primary steps of the algorithm are:

1. Initialize Population. In this step we create a population of chromosomes, where each chromosome represents a set of parameters. This involves randomly generating each parameter in the chromosome while constraining each parameter to only valid values.

2. Evaluate Fitness. Here we evaluate the performance of each chromosome in our population by applying a fitness function. The fitness function is different for the two stages, but they both produce a fitness value in the interval [0, 1] for each chromosome.

3. Sort Population. We sort the population in descending order in terms of their fitness values.

4. Crossover. We look at each chromosome in the top half of the population and use it as a parent in a crossover operation with probability PROB_X, which is given as a parameter to the algorithm. We tested values of .5, .7 and .9 for PROB_X. We then will choose a second parent through round-robin selection across the entire population of chromosomes. We then create two new chromosomes by blending the parameters of the two parents previously selected. We blend each parameter by first choosing a blend factor from a uniform distribution over [0, 1] and then using the blend factor in the equations:

$$child1.param = blend \times parent1.param + (1 - blend) \times parent2.param$$

$$child2.param = blend \times parent2.param + (1 - blend) \times parent1.param$$

If the parameters are discrete, then rounding must be performed as well. Once the new children have been initialized, they then replace two chromosomes that are in the bottom half of our population.

5. Mutation. Here we iterate over our population of chromosomes and mutate each chromosome with probability PROB_M, which is given as a parameter to the algorithm. We tested values of .01, .1, and .25 for PROB_M. If a chromosome is selected for mutation, we then slightly alter each of its parameters. If the parameter is a discrete variable, this entails incrementing or decrementing its value (both with equal probability). Otherwise, we scale the variable by a random value from [.5, 1.5].

6. Iteration. Repeat steps 2 through 5 for NUM_GENERATIONS, which again is given as a parameter to the algorithm. The value used here was determined based on the run-time of the ECA's.

4.2 LBP, HOG, and SVM Parameter Optimization

In order to evaluate the fitness of a set of parameters to the LBP, HOG, and SVM algorithms efficiently, we built a collection of images cropped to contain the image inside each bounding box produced by the prescreener. This collection consists of 5 times as many false alarms as true targets, and was constricted to containing 1000 images from one data-collection run. Each image is of a distinct cluster on the lane.

Using this collection of images, we are then able to efficiently calculate the effectiveness of the chromosomes' parameters by training an SVM using 70% of the samples and then testing the SVM on the remaining 30%. The fitness value for a chromosome is defined as the percentage of the samples that were correctly classified by the SVM.

The parameters being optimized in this ECA include the SVM kernel type, the degree of the SVM, the number of windows on the X-axis for the HOG, the number of windows on the Y-axis for the HOG, the number of bins for the HOG, the LBP radius, the number of points to sample for the LBP, the mode of the LBP (either histogram or normalized histogram), and the feature flag, which specifies what combination of LBP and HOG features to use. The parameters we were using before optimization are given in Fig. 5.

```
FEATURE_FLAG: 0
  SVM_KERNEL: 2
  SVM_DEGREE: 3
  HOG_NWIN_X: 3
  HOG_NWIN_Y: 3
    HOG_BINS: 9
  LBP_RADIUS: 1
       LBP_N: 8
    LBP_MODE: 'h'
```

Figure 5. Default parameters to LBP, HOG, and SVM algorithms. A feature flag of 0 specifices the usage of both LBP and HOG features, while an SVM kernel of 2 specifies the radial-basis function (RBF) kernel. Also, an LBP mode of 'h' specifies to use a histogram.

4.3 Canny Edge Detector Parameter Optimization

We begin the evaluation of the fitness of the Canny edge detector parameters by running the prescreener through one data-collection run, keeping track of the number of positive detections (bounding boxes with a target in them), negative detections (boxes with no target), and total true targets (incremented every time there is a mine in the view of the frame). We then combined these three values to produce a single fitness value. This combination proved to be difficult, and the different approaches we took are detailed in the results section.

The parameters being optimized in this ECA include the upper and lower thresholds to the Canny edge detector along with the standard deviation of the Gaussian filter used. For reference, the parameters being used before are given in Fig. 6.

```
UPPER_BLOB_THRESHOLD: 0.19
LOWER_BLOB_THRESHOLD: 0.0
               SIGMA: 1.414
```

Figure 6. Default parameters to Canny edge detector. Sigma is the standard deviation being used by the Gaussian filter.

5 Results

Applying the algorithms described in the previous section we obtained the following results, where each ROC curve shown has been produced using the same data run.

5.1 LBP, HOG, and SVM Parameter Optimization Results

We ran our Evolutionary Computation Algorithm for various values of PROB_X and PROB_M and obtained the results for average fitness per generation, Fig. 7, and max fitness per generation, Fig. 8.

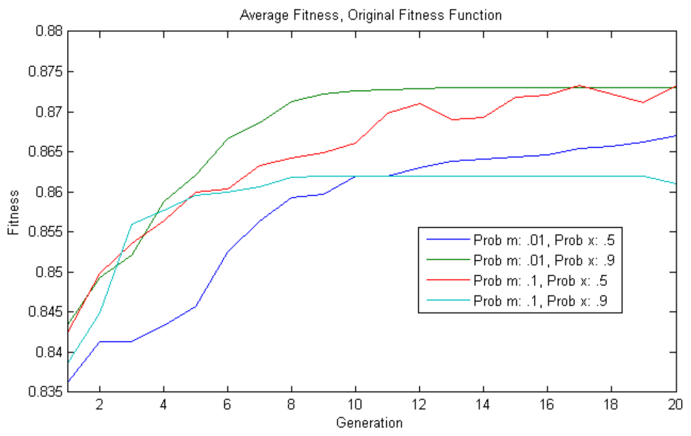


Figure 7. Average fitness of chromosome population per generation of LBP, HOG, and SVM parameter optimization. Several runs of the algorithm are plotted, with varying probabilities of mutation and crossover.

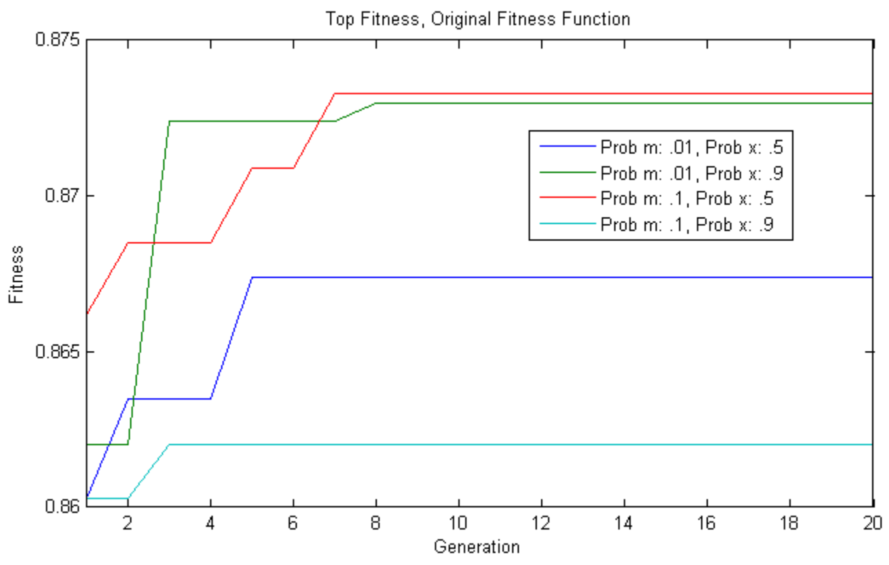


Figure 8. Maximum fitness of chromosome population per generation of LBP, HOG and SVM parameter otpimization Several runs of the algorithm are plotted, with varying probabilities of mutation and crossover.

These graphs demonstrate a few things. First, we can see that the Evolutionary Algorithm is working as expected since the average fitness gradually converges to the top fitness for each run of the algorithm. Additionally, the average fitness curve for the runs with a higher crossover probability (PROB_X=.9) converge more quickly on the maximum fitness attained for that run, while the runs with a higher mutation probability (PROB_M=.1) are less smooth, which is to be expected since we are introducing more randomness in these cases.

```
FEATURE_FLAG: 0    FEATURE_FLAG: 0
  SVM_KERNEL: 2      SVM_KERNEL: 2
  SVM_DEGREE: 2      SVM_DEGREE: 3
  HOG_NWIN_X: 6      HOG_NWIN_X: 3
  HOG_NWIN_Y: 4      HOG_NWIN_Y: 3
    HOG_BINS: 8        HOG_BINS: 9
  LBP_RADIUS: 1      LBP_RADIUS: 1
       LBP_N: 11          LBP_N: 8
    LBP_MODE: 'h'      LBP_MODE: 'h'
```

Figure 9. Evolved parameters (left) vs default parameters (right)

One interesting part of our results for this ECA is that the optimal parameter set (from the run with PROB_M=.1, PROB_X=.5) generated several parameters with the same value as the ones we were already using, as can be seen in Fig. 9. Even more surprising is that the Receiver Operating Characteristic (ROC) curve generated by our algorithm is exactly the same for both of these parameter sets. Thus, the algorithm does not appear to be very sensitive to small changes in its input parameters, particularly to the HOG and LBP algorithms. The only parameters that appear to be crucial, as they were shared across all optimized solutions, were that the SVM kernel be an RBF and that both the LBP and HOG features be used in classification.

5.2 Canny Edge Detector Parameter Optimization Results

Since we had verified the general ECA was functioning properly in the previous section and had found our specific implementation of the ECA to perform the best with PROB_M=.1 and PROB_X=.5, we only tested this ECA using those parameters. We tested our Evolutionary Computation algorithm on this section using several slightly different fitness functions. We began by specifying the fitness of a chromosome as:

$$fitness = \frac{trueHits}{trueHits + falseAlarms} \times \frac{trueHits}{numberOfTargets}$$

Where *trueHits* is the number of bounding boxes identified with a mine in them, *falseAlarms* is the number of bounding boxes identified without a mine in them, and $numberOfTargets$ is the total number of targets present in all frames of the run. This fitness function yielded parameters given in Fig. 10. It is interesting to note that the ratio between the upper and lower threshold values is 2.07, which falls firmly in the range of 2 to 3 that Dr. Canny recommended in his initial paper on the Canny edge detector [17].

```
UPPER_BLOB_THRESHOLD: 0.1548  UPPER_BLOB_THRESHOLD: 0.19
 LOWER_BLOB_THRESHOLD: 0.0748 LOWER_BLOB_THRESHOLD: 0.0
               SIGMA: 2.4646                 SIGMA: 1.414
```

Figure 10. Evolved parameters after first attempt (left) vs default parameters (right)

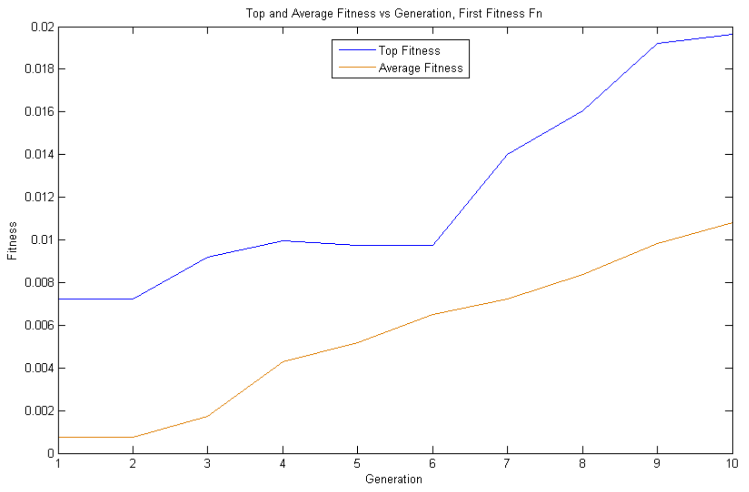


Figure 11. Top and Average Fitness values vs. generation for the first fitness function used in the optimization of parameters for the Canny Edge detector, with PROB_M=.1 and PROB_X=.5.

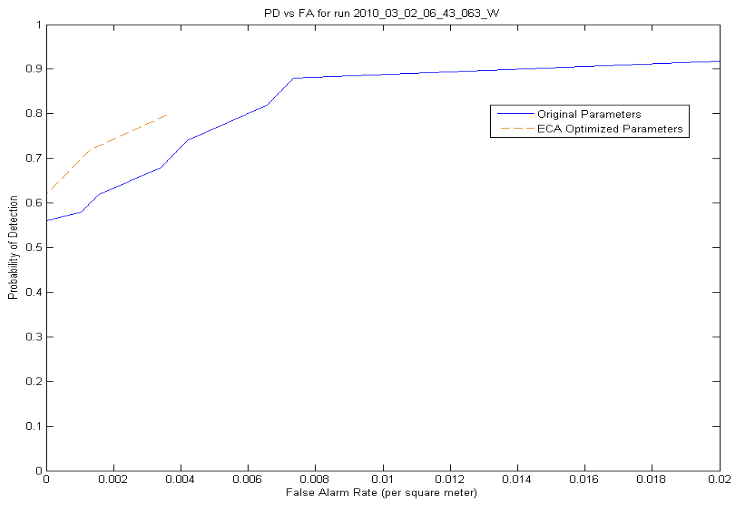


Figure 12. ROC Curve for parameters shown in Figure 10 versus the ROC curve for our original parameters.

We then tested our full algorithm using the parameters in Fig. 10. We did this by training our algorithm on 6 runs of data, and testing on a 7th run. We only ran this ECA for 10 generations due to its fitness function having a longer run time than the previous ECA, but the ECA still managed to double the top fitness achieved as shown in Fig. 11. Our test produced the Receiver Operating Characteristic (ROC) curve shown in Fig. 12. As can be seen in Fig. 12, the parameter set optimized by our first fitness function greatly reduced the number of *falseAlarms*, allowing for greater probabilities of detection at lower false alarm rates. We have observed that a human observer that scored this run achieved a probability of detection of 80% with no false alarms, whereas we achieved a probability of detection of 80% at a false alarm rate of .004 per square meter. Thus, there is still room for improvement, but these latest results prove promising as we have been able to reduce the false alarm rate by 50% from our original parameters (at a probability of detection of 80%).

However, these optimized parameters also reduced the number of *trueHits* recorded, preventing us from achieving a probability of detection greater than .9 regardless of the false alarm rate. Thus, we tried again with the following fitness function, which gives more weight to *trueHits*:

$$fitness = \frac{trueHits}{trueHits + falseAlarms} \times sigmoid((trueHits - 510)/25)$$

Here we used a sigmoid function to sharply penalize any results that yielded an amount of *trueHits* less than 510, which is the number of *trueHits* obtained by our original parameter set. The difference is divided by 25 in order to make the curve slightly smoother, and 25 was chosen so that any result with *trueHits* < 485 (an unacceptable number of *trueHits*) would be reduced by a factor of 0.269, thus more than likely moving it to the bottom half of the population. This fitness function yielded the parameters given in Fig. 13.

```
UPPER_BLOB_THRESHOLD: 0.1380  UPPER_BLOB_THRESHOLD: 0.1548
 LOWER_BLOB_THRESHOLD: 0.1018  LOWER_BLOB_THRESHOLD: 0.0748
               SIGMA: 1.356                      SIGMA: 2.4646
```

Figure 13. Evolved parameters after second attempt (left) vs first attempt (right)

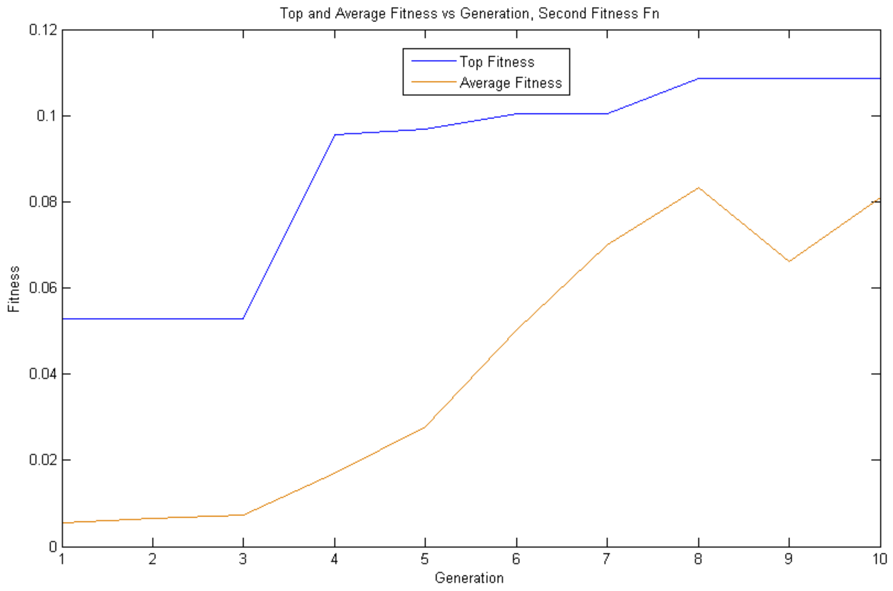


Figure 14. Top and Average Fitness values vs. generation for the second fitness function used in the optimization of parameters for the Canny Edge detector, with PROB_M=.1 and PROB_X=.5.

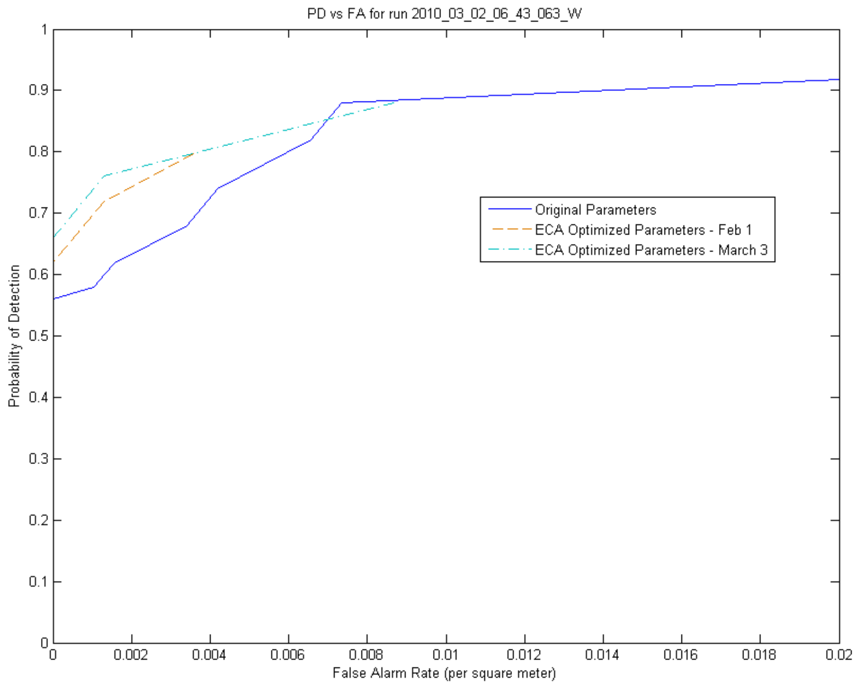


Figure 15. ROC curve comparision across evolved parameter sets.

We then tested our full algorithm with the new set of optimized parameters, and obtained the ROC curve shown in Fig. 15. As can be seen, these parameters yielded our best results yet by both improving probability of detection for low false alarm rates and increasing the maximum probability of detection achieved. Additionally, the system matched the maximum probability of detection achieved by the default parameters (for reasonable false alarm rates).

6 Conclusions

In this paper we outlined a system for detecting buried targets using FLIR imagery and a support vector machine classifier. We then presented a method of automatically optimizing various parameters to our system using an evolutionary computation algorithm. Using this algorithm, we found that we were able to reproduce the same results previously achieved through a "guess and check" method. Additionally, we were able to improve the probability of detection of our system at low false alarm rates (< 1 per 250 square meters). Hence, this optimization approach can be used to determine system parameters under varying scenarios and perform as well (or better) than the arduous task of human trial and error.

7 Acknowledgments

This Work funded by ARO grant 57940-EV to support U. S. Army RDECOM CERDEC NVESD.

References

[1] Li B., Chellapa R., Zheng Q., Der S., Nasrabadi N., Chan L., Wang L., "Experimental Evaluation of FLIR ATR Approaches—A Comparative Study", *Computer Vision and Image Understanding* 84, 5–24 (2001).

[2] Stone, K., Keller, J. M., Popescu, M., Havens, T.C., Ho, K. C., "Forward Looking Anomaly Detection via Fusion of Infrared and Color Imagery", Proc. of SPIE 7664, 2010.

[3] Yu Y., Guo L., "Infrared Small Moving Target Detection Using Facet Model and Particle Filter", Proc of 2008 Congress on Image and Signal Processing, pp. 206-210.

[4] Schmuggea,T., French, A., Ritchie, JC, Rango A., Pelgrum H., "Temperature and emissivity separation from multispectral thermal infrared observations", Remote Sensing of Environment 79 (2002) 189– 198.

[5] Winter,E. M., Fields,D. J., Carter, M. R., Benett, C. L., Lucey,P. G., Hohnson,J. R., Horton,K. A., and Bowman, A. P., "Assessment of Techniques for Airborne Infrared Land Mine Detection", Proc.of the Third International

Airborne Remote Sensing Conference and Exhibition, Copenhagen, Environmental Research Institute of Michigan, Ann Arbor, Vol. II, 1997, pp. 44-51.

[6] Thanh N.T., Sahli, H., Hao D.N., "Infrared Thermography for Buried Landmine Detection: Inverse Problem Setting," *IEEE Transactions on Geoscience and Remote Sensing*, vol.46, no.12, pp.3987-4004, Dec. 2008.

[7] Stone K., Keller M., Popescu M., Spain C.J., “Buried explosive hazard detection using FLIR imagery”, proceedings of SPIE Defense, Orlando, FL, April 24-29, 2011.

[8] Spain C.J., Popescu M., Keller J., Stone K., “Automatic Detection of targets in medium-wave Infrared Imagery using adaptive background mixture models”, proceedings of SPIE Defense, Orlando, FL, April 24-29, 2011.

[9] Popescu M., Stone K., Keller J., "Detection of targets in forward-looking infrared imaging using a multiple instance learning framework", proceedings of SPIE Defense, Orlando, FL, April 24-29, 2011.

[10] Banerji, A., Goutsias, J., "A morphological approach to automatic mine detection problems", IEEE Trans Aero Elect Sys 34:1085–1096, 1998.

[11] Chang’an W., Shouda J., “Automatic Target Detection and tracking in FLIR Image sequences using Morphological connected operator”, IIH-MSP, 2008, pp. 414-417.

[12] Lewis D., Keller J., “Automatic road finding in FLIR”, CISDA 2012, Otawa, Canada, 2012.

[13] Heikkilä, M. and Pietikäinen, M. (2006), “A Texture-Based Method for Modeling the Background and Detecting Moving Objects”, IEEE Trans. Pattern Analysis and Machine Intelligence 28(4):657-662.

[14] Ludwig O., Delgado D., Goncalves V., and Nunes U., “Trainable Classifier-Fusion Schemes: An Application To Pedestrian Detection,” In: 12th International IEEE Conference On Intelligent Transportation Systems, 2009, St. Louis, 2009. V. 1. P. 432-437.

[15] Chih-Chung Chang and Chih-Jen Lin, LIBSVM : a library for support vector machines, 2001, http://www.csie.ntu.edu.tw/~cjlin/libsvm

[16] Popescu, Mihail; Paino, Alex; Stone, Kevin; Keller, James M.; , "Detection of buried objects in FLIR imaging using mathematical morphology and SVM," Computational Intelligence for Security and Defence Applications (CISDA), 2012 IEEE Symposium on , vol., no., pp.1-5, 11-13 July 2012

[17] Canny, John; , "A Computational Approach to Edge Detection," Pattern Analysis and Machine Intelligence, IEEE Transactions on , vol.PAMI-8, no.6, pp.679-698, Nov. 1986