2006

Optical-based ATR algorithms for applications in swarmed UAVs

Vasavi Rangammagari
West Virginia University

Optical-Based ATR Algorithms for Applications in Swarmed UAVs

by

Vasavi Rangammagari

Thesis submitted to the
College of Engineering and Mineral Resources
at West Virginia University
in partial fulfillment of the requirements
for the degree of

Master of Science
in
Electrical Engineering

Matthew C.Valenti, Ph.D., Chair
Xin Li, Ph.D.
Natalia A. Schmid, Ph.D.

Lane Department of Computer Science and Electrical Engineering

Morgantown, West Virginia
2006

Keywords: ATR, Sensor Fusion, Image Processing

Abstract

Optical-Based ATR Algorithms for Applications in Swarmed UAVs

by

Vasavi Rangammagari

Swarmed Unmanned Aerial Vehicles (UAVs) with Automatic Target Recognition (ATR) technology are becoming an important element of electronic warfare. The advantages of UAVs over piloted vehicles have increased the need to develop robust and reliable ATR algorithms. Various issues like changing weather, camouflage, low contrast and resolution, clutter, inadequate databases place a limit on the performance capabilities of a typical ATR algorithm. In an effort to deal with these issues, a correlation-based algorithm is proposed in this thesis. This algorithm calculates the correlation between the input image and a target template which is created by projecting a 3-D model from the perspective of the UAV. The locations of correlation peaks are then declared to be the locations of the targets. We apply this algorithm to images with one object of a known class and move on to the more general case of images with an unknown number of targets from one or more classes. We provide an analysis of the performance of this correlation-based algorithm.

We compare the performance of the proposed correlation-based approach with that of a training-based approach. To provide a concrete example of an "off-the-shelf" training-based ATR algorithm, the open source IntelCV library was used. In the training-based method, a sample set is created and trained (Haar-like features are used for training) to produce results for comparison purpose. We further develop and analyze a method of correlating across multiple frames that have been preprocessed using the correlation-based approach. This method is shown to be useful in detecting true targets and suppressing false alarms in cases where a single image is not sufficient for classification.

Acknowledgments

I take this opportunity to thank Dr. Matthew Valenti who stood by me through out the entire ordeal of organizing the idea and conceiving it. “This would not have been possible without him”, itself is an understatement. I would also like to thank Dr. Natalia Schmid who provided resourceful ideas which aided in the completion of this thesis. I also thank Dr. Xin Li who agreed to be on my committee. He has been a source of inspiration to me. I would also like to thank Xiaohan Chen, who lent more than just a helping hand in the completion of my thesis. Last but not the least, I would like to thank my family and friends who inspired me through out.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation and Research Objectives

UAVs (Unmanned, Unpiloted, or Unattended Aerial Vehicles) are rapidly becoming an important tool in modern military applications. Because UAV's are used primarily for surveillance, there is an increasing necessity to develop reliable Automatic Target Recognition (ATR) systems for use in UAVs. Automatic target recognition is the computer processing of image data obtained from image sensors (example: optical, radar, infrared etc.) to acquire (or detect), identify (or recognize) and track image locations of objects of interest [3, 4]. ATR systems find applications in national and civil defence, medical, industrial and commercial fields. Detection is the process of distinguishing between target and non-target objects. Recognition involves identifying the target being detected [5, 6]. ATR systems identify and classify targets based on the features specific to a target. Depending on the application, there is a trade-off between the probability of false alarm and the probability of missed detection. This trade-off can be quantified in terms of a Receiver Operating Characteristic (ROC) curve, which plots the achievable probability of detection as a function of the probability of false alarm. Most ATR systems use either pixel, feature or decision level fusion of information [7, 8, 9]. Some systems may use a combination of the above mentioned criteria. ATR algorithms may or may not use collateral information (like maps, meteorological data, altimeter etc.). ATR systems must be robust and adaptive. Robustness is a measure of a system's insensitivity to the variations in input conditions. An ATR algorithm

can perform well and would not fail under any test conditions if it can learn and adapt to varying operating conditions.

1.2 Swarmed UAVs

Advances in sensor technology, wireless communication and other engineering areas have paved a path for the development of UAVs with good target recognition capabilities. UAVs today can search, detect, suppress and take appropriate actions against enemy targets. UAVs offer many advantages: they are economical, useful in high danger scenarios and thus reduce risk to life, and have an edge over manned vehicles in certain types of missions. The most common applications of UAVs are search for rescue/destroy operations, disaster response and exploration. Efforts are being made to equip UAVs with a high level of intelligent autonomy to carry out various tasks with little or no supervision and to quickly adapt to new environments [10].

Swarms[1] of UAVs can be used in tasks where a single UAV would not serve the purpose. Under such conditions, each UAV works as an autonomous agent by sensing the environment, analyzing threats and sending out notifications and responses to other UAVs within the swarm. Swarms of UAVs configure themselves and coordinate their actions and have inherent advantages like workload partitioning (i.e., dividing up the targets to be tracked or attacked) and workload coordination, resulting in cooperative classification of enemy targets and refined imagery. A mission's likelihood of success is enhanced by the use of such multiple UAVs [10].

Coordination of multiple UAVs is achieved by spatial, temporal and team coordination [12, 13] among the UAVs. Spatial coordination is accomplished by distributing the UAVs in the region of interest, coming up with a nonconflicting air space for UAVs, determining the acceptable distance between vehicles and determining the acceptable number of revisits per vehicle. Temporal coordination is obtained by making sure that UAVs act accordingly at the right moment so that the team works efficiently. Team coordination is about the

[1]Swarm intelligence is inspired from the behavior of social insects. Swarms of biological organisms offer good examples of emergent self-organizing coordination mechanisms that develops from the local interactions [11].

optimization of the tasks assigned to each UAV in accordance with its preferences, capabilities and constraints, managing the formation and maintenance of the UAVs [12]. Swarms of UAVs generally satisfy three requirements: speed (fast search), coverage (the entire region of interest should be covered) and efficiency (deficient resources should be conserved) [11, 14]. The requirements are balanced by the coordination policy to achieve an acceptable trade-off.

1.3 Common Problems Associated with ATR

A good ATR algorithm should be able to detect objects in a way that it is invariant to intra-class variations and at the same time, distinguish objects of different classes [15, 16]. Variables that affect the ATR performance include changing weather, seasons, terrain, viewing angles, target orientation, target operating state, camouflage, low contrast and resolution, deception and clutter. Even objects belonging to the same category have a wide range of variation in appearance due to varying viewing conditions (like illumination changes, different viewpoints, different sensors, noise etc.). The primary challenge is coming up with models that include the essence of the object class and are flexible enough to include variations in the object. Effortless learning and adaption to new objects and environment is also a key issue in the design of ATR algorithms.

The success of an ATR algorithm is characterized by a controlled operational environment, a sensor that meets all requirements, a clear set of objectives and a good testing and training database of images. The above mentioned conditions are very rarely available for an ATR algorithm developer [3, 17]. The ATR algorithm developer does not have control over the operational conditions, or access to a complete database which is representative of all possible operating conditions (such data collections have prohibitively high costs and are estimated to contain at least a billion images). Even images collected over the years would not solve the problem of insufficient databases of images as the image quality improves over the years, thus rendering older images obsolete. Also, target definitions change over the years. Sensor simulators that can produce several images by simulating the operating conditions and the sensors, can be used as a means to alleviate this problem. The images thus generated should represent real world imagery since the capabilities of any ATR algorithm are

demonstrated only when used in the field. A large bias on performance can be introduced by the use of such synthetic images. Hence, caution should be used when substituting synthetic images for real imagery. However, the use of synthetic images is very useful in the design of robust ATR algorithms for new applications such as swarmed UAVs, where actual imagery is limited. Therefore, synthetic images (produced using the ATR 3D Training tool produced by Augusta Systems) will be used throughout this thesis.

ATR algorithms are iterative. The results of one iteration are used to improve the results of the next iteration by, for instance, decreasing the number of missed targets and false alarms while maintaining the detected targets [3, 18]. Such kind of iterative development of the ATR algorithm can lead to very good performance of the algorithm when applied to the existing database of images but can fail completely when applied to a new set of images taken in different environment. One way to deal with this problem is to divide the available database into two sets: training and testing sets. The test imagery determines the accuracy and reliability of the ATR algorithm in the field. However, this would still not alleviate the problem of an inadequate database.

There is a chance of ATR algorithms becoming outdated due to several reasons: the target definitions can change unexpectedly due to unseen events; the existing model of the target may get replaced by a new model which may have entirely different image domain features; the background environment can change; and the enemy can use intelligent deceptive methods. Hence its advisable to have ATR algorithms that are economical to create and modify (to accommodate the various circumstances that render the algorithm obsolete) which is not the case with the ATR algorithms of today. They all are expensive and are developed slowly and thus have high disposing costs [3].

Sometimes the objective for the ATR algorithm development is not specified in image domain terms especially in the case where the objects to be recognized are functionally identical but do not have anything in common as far as image domain is concerned. For example there is no way we can train an ATR algorithm to recognize a pistol (a hand held weapon) given the fact that the algorithm can recognize a sword (an object of the same class i.e., hand held weapon). Such kind of objectives cannot be translated into image domain terms [19].

The computational throughput is a function of the incoming data bandwidth, the image processing functions and their complexity. Also the speed of the algorithm is dependent on the capabilities of the available hardware. Today commercial hardware exists to perform many of the component modules in the ATR algorithm. These get the computational time down to a few seconds to process a large number of images [20].

Thus, there exists a need for developing ATR algorithms that perform efficiently in dynamically changing environmental conditions and adapts easily to changes in ATR objectives.

1.4 Thesis Outline

This thesis mainly focuses on the development and analysis of algorithms for correlation based ATR and compares those algorithms against existing training based ATR algorithms. The training and test images for these algorithms were mainly generated using the ATR 3D training tool developed by Augusta Systems.

Chapter 2 exposes the notation, definitions and image operators used throughout the thesis. Chapter 3 introduces algorithms for correlation based ATR. The basic idea behind the correlation based approach is template matching i.e., finding the correlation between the template and the image and looking for correlation peaks to find the location of the target. This chapter concludes with a performance evaluation of the correlation based approach.

The training-based approach to ATR is discussed in Chapter 4. This method consists of three basic steps which are executed using Intel OpenCV Library [2]: creating a good training sample set consisting of positive and negative samples, training this set using Haar training (which utilizes Haar-like features in the training set) and testing the performance of the approach (using the same set of images that were used to test the performance of correlation based ATR).

Chapter 5 shows how correlation is used for detection and recognition of targets across multiple frames. The multiple frames could be images by the same UAV, or they could be from images taken by multiple UAVs. Such information fusion across multiple frames leads to reliable and robust detection of targets which are not easy to detect using single frame. Chapter 6 concludes with some suggestions for future work.

Chapter 2

A Mathematical Framework for Correlation-Based ATR

This chapter presents the notation, definitions and operators used in the following chapters. Section 2.1 deals with the basic notation used in describing the images and the image operators. Sections 2.2 and 2.3 describe the unary and binary image operators respectively.

2.1 Basic Notation

Let $\mathbf{X}$ be a grayscale image (taken from a UAV) represented by a matrix of size[1] M by N. The notation $\mathbf{X} = [x_{(m,n)}]$ is used to define $\mathbf{X}$ as the matrix whose $(m,n)^{th}$ element is $x_{(m,n)}$. Pixel $x_{(m,n)}$ takes on a value between 0 and 1[2], where $x_{(m,n)}$ is the $(m,n)^{th}$ element of $\mathbf{X}$, $1 \leq m \leq M, 1 \leq n \leq N$. Fig. 2.1 shows a typical reference image where the object of interest is the box.

The target of interest is specified in terms of a template. Each target type is represented as a 3-D polygon, and the polygon representation is stored in memory. Let $\mathbf{\Omega} = \{\theta, \phi, \mathrm{r}\}$, where θ and ϕ are the azimuth rotation angle and elevation angle respectively, relative to the UAV and r is the range (or distance) from UAV to target. Define $\mathbf{T}(\mathbf{\Omega})$ to be a grayscale 2-D projection of the target template (of size J by K), according to the view from the UAV. Fig. 2.2 shows the silhouette of the object of interest (box) as would be seen from the perspective

[1] Matrix indices start from 1.
[2] 0 is black and 1 is white.

Figure 2.1: Image showing the object of interest (box).

of the UAV. We assume that rotation in the azimuth plane (θ) is unknown but the elevation angle (ϕ) and distance to target (r) are known. The elevation angle can be calculated by simple trigonometry when the location of the UAV and the distance from the UAV to the target, based on the terrain on which the target is located, are known (Fig. 2.3 shows a typical scenario being described here). We assume that the target is on flat ground, so its absolute roll and pitch angles are both zero. For ease of exposition, let us assume the target is white and the background is black i.e. $\mathbf{T}(\boldsymbol{\Omega})$ is all ones inside the target, and all zeroes outside. A more sophisticated version of this algorithm could take into account the actual shading of the target and a priori knowledge of the average intensity of the background, and an even more sophisticated could operate on color images.

Figure 2.2: Target template of the object of interest in Fig. 2.1.

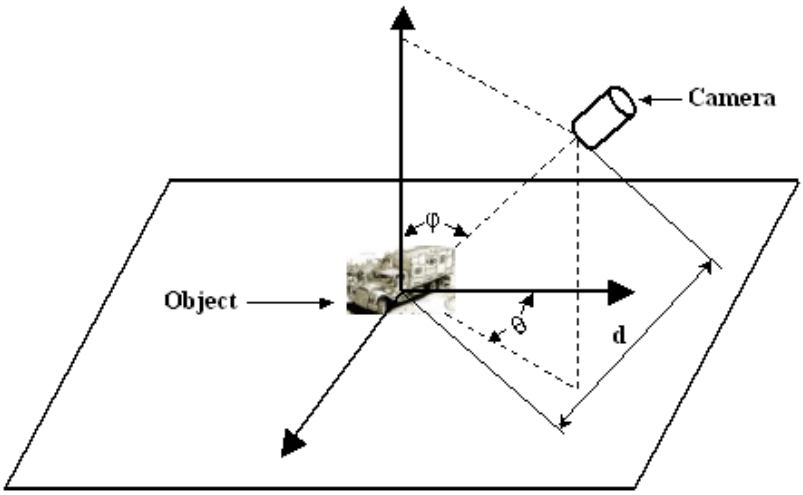

Figure 2.3: A typical scenario showing the camera parameters.

2.2 Mathematical Operators

This section presents the image operators used throughout this thesis. These image operators are used mainly in the description and development of correlation-based ATR algorithm. There are two types of operators described here: unary and binary operators. Unary operators work on a single image while the binary operators operate on two images.

2.2.1 Unary Operators

Inversion: $\mathbf{X}' = \mathbf{1} - \mathbf{X}$. This operation is called *inversion* [21]. Applying this operation to an image makes the bright parts of the image dark and vice versa.

Grayscale Expansion: $\hat{\mathbf{X}} = 2\mathbf{X} - \mathbf{1}_{M,N}$, where $\mathbf{1}_{M,N}$ is a matrix of all ones. Assuming that the values of $\mathbf{X}$ are between zero and one, this operation converts each value of $\mathbf{X}$ to a corresponding negative or positive number depending on whether it is less than or greater than 0.5. Any pixel value equal to 0 becomes -1. This operation is called grayscale expansion as it expands the range so that instead of going from 0 to 1, it

now goes from -1 to +1.

Thresholding: $\bar{\mathbf{X}}(T)=[\bar{x}_{(m,n)}]$, where

$$\bar{x}_{(m,n)} = \begin{cases} 0, & \text{if } x_{(m,n)} < T; \\ 1, & \text{otherwise.} \end{cases}$$

Here $x_{(m,n)}$ is the *(m,n)*th element of $\mathbf{X}$ and T is threshold. This operation is called *thresholding* [21]. This causes $\bar{\mathbf{X}}$ to be a binary image where all the pixels are either 0's (black) or 1's (white).

Windowing: $\mathbf{X}^{(i,q)}$, is a J by K submatrix of $\mathbf{X}$, centered at the coordinate (i,q), $1 \leq i \leq M, 1 \leq q \leq N$. More specifically, $\mathbf{X}^{(i,q)}=[x^{(i,q)}_{(j,k)}]$, where $x^{(i,q)}_{(j,k)}$ is the $(j,k)^{th}$ element, $1 \leq j \leq J, 1 \leq k \leq K$, and is related to $\mathbf{X}$ by $x^{(i,q)}_{(j,k)}=x_{(m,n)}$, where $m = j + i - \lceil J/2 \rceil$ and $n = k + q - \lceil K/2 \rceil$. This is *windowing*. If the elements of the windowed segment fall out of range of dimensions of $\mathbf{X}$ then each of those elements is set to a pixel value that is an average of all the pixel values of the image $\mathbf{X}$.

2D DFT: $\mathbf{A} = \mathcal{F}(\mathbf{X})$, returns the two dimensional discrete fourier transform (DFT) [22] of $\mathbf{X}$, could be computed with a fast Fourier transform (FFT) algorithm [3]. The result $\mathbf{A}$ is the same size as $\mathbf{X}$. $\mathcal{F}(\mathbf{X})$ can be computed by first finding the one-dimensional DFT of each column of $\mathbf{X}$ (i.e., M point DFT, where M is the number of rows of $\mathbf{X}$) and then of each row of the result (i.e., N point DFT, where N is the number of columns of $\mathbf{X}$). The one-dimensional DFT of a vector $\mathbf{Y}$ of length N is calculated by:

$$\mathbf{Y}(k) = \sum_{n=1}^{N} y(n)\omega_N^{(n-1)(k-1)}, \tag{2.1}$$

where $\omega_N = e^{(-j2\pi)/N}$ is the N^{th} root of unity.

$\mathbf{A} = \mathcal{F}(\mathbf{X}, m, n)$ truncates $\mathbf{X}$, or pads $\mathbf{X}$ with zeros to create a m by n array before the transform. The result is m by n.

[3] FFT requires that N be a power of a small prime number. When N is a power of 2, FFT algorithm reduces the number of computations required for N points from $2N^2$ to $2N\log_2 N$.

2.2.2 Binary Operators

Correlation: $C(\mathbf{X}, \mathbf{Y}) = [c_{(i,q)}]$, is a *correlation matrix*. Correlation is commutative, so $C(\mathbf{X}, \mathbf{Y}) = C(\mathbf{Y}, \mathbf{X})$. The correlation matrix has the same size as the larger of the two matrices in its argument. Without loss of generality, assume that $\mathbf{X}$ is the larger of the two matrices and has a size M by N, while $\mathbf{Y}$ is of the size J by K, $J \leq M, K \leq N$. Element $c_{(i,q)}$ of the correlation matrix is the correlation of the J by K windowed segment $\mathbf{X}^{(i,q)}$ centered at (i, j) with $\mathbf{Y}$. More specifically, the value is calculated as:

$$c_{(i,q)} = \sum_{j=1}^{J} \sum_{k=1}^{K} x_{(j,k)}^{(i,q)} y_{(j,k)}. \tag{2.2}$$

Correlation can also be performed in the frequency domain. Correlation of an image $\mathbf{Y}$ with an image $\mathbf{X}$ (of size m by n) in the frequency domain is given by

$$\mathbf{C}_F(\mathbf{X}, \mathbf{Y}) = Re(\mathcal{F}^{-1}[\mathcal{F}(\mathbf{X})\mathcal{F}(\mathbf{Y}_r, m, n)]), \tag{2.3}$$

where $\mathbf{C}_F$ is the correlation in frequency domain, $\mathcal{F}^{-1}$ is the inverse two dimensional DFT and $\mathbf{Y}_r$ is $\mathbf{Y}$ rotated by 180^0.

Decision Statistic: $S(\mathbf{X}, \mathbf{Y}) = C(\hat{\mathbf{X}}, \hat{\mathbf{Y}})$, where S is the decision statistic. The value of $S(\mathbf{X}, \mathbf{Y})$ ranges from -JK to +JK.

Dual Complimentary Decision Statistic:

$$Z_d(\mathbf{X}, \mathbf{Y}, a) = aZ(\mathbf{X}', \mathbf{Y}) + (1 - a)Z(\bar{\mathbf{X}}, \mathbf{Y}'), \tag{2.4}$$

Here a is a weight chosen between 0 and 1 and

$$Z(\mathbf{X}, \mathbf{Y}) = JK\mathbf{1}_{M,N} - C(\mathbf{X}, \mathbf{Y}), \tag{2.5}$$

where $\mathbf{1}_{M,N}$ is a M by N matrix of all ones and J, K are the dimensions of the smaller of the two matrices $\mathbf{X}$ and $\mathbf{Y}$. The range of $Z(\mathbf{X}, \mathbf{Y})$ is 0 to JK.

Chapter 3

Algorithms for Correlation-Based ATR

This chapter describes a correlation-based approach to finding an object within a optical image taken from a UAV-based camera. The location and the orientation of the targets are unknown, but that of the UAV are known. Some knowledge of the underlying terrain is assumed so that even though the orientation of the target in the azimuth direction is unknown, the elevation angle and range from target to sensor can be determined. Each target type is described by a 3-D polygon, and a processor projects the 3-D representation to a 2-D plane according to the estimated relative orientation between UAV and target.

We first considered the use of a single decision criteria $S(\mathbf{X}, \mathbf{T}(\mathbf{\Omega})) = C(\hat{\mathbf{X}}, \hat{\mathbf{T}}(\mathbf{\Omega}))$, which did produce some good results, but there were a lot of false alarms (see Fig. 3.1), which would not get suppressed with the use of a single decision criteria (A ROC[1] curve showing the performance of this algorithm is shown in Fig. 3.2). This algorithm works better in the cases where the background is entirely black, which is not the case with most of the input images used for ATR. We then tried setting the template background value equal to inverse of average intensity of the input image. This modification produced slightly better results but still would not entirely suppress false alarms. Because of these problems we proposed

[1]ROC stands for Receiver Operating Characteristic. ROC is a measure of detection performance. The values for the ROC curve were calculated as follows: A set of test images were taken and ATR algorithm was applied. The threshold for detection was varied uniformly and the number of true detections and false alarms were recorded for each of the test images as the threshold was varied. Then the detection rate vs. the false alarm rate was plotted for each of the threshold values.

an improved method which uses two decision criteria to detect the objects of interest while successfully eliminating the false alarms. This improved method, discussed below, uses two decision thresholds to get good results. The first threshold picks up all objects of interest, and also may pickup few false alarms. The second threshold makes sure that the false alarms get suppressed. The sections below discuss the development and analysis of this improved algorithm.

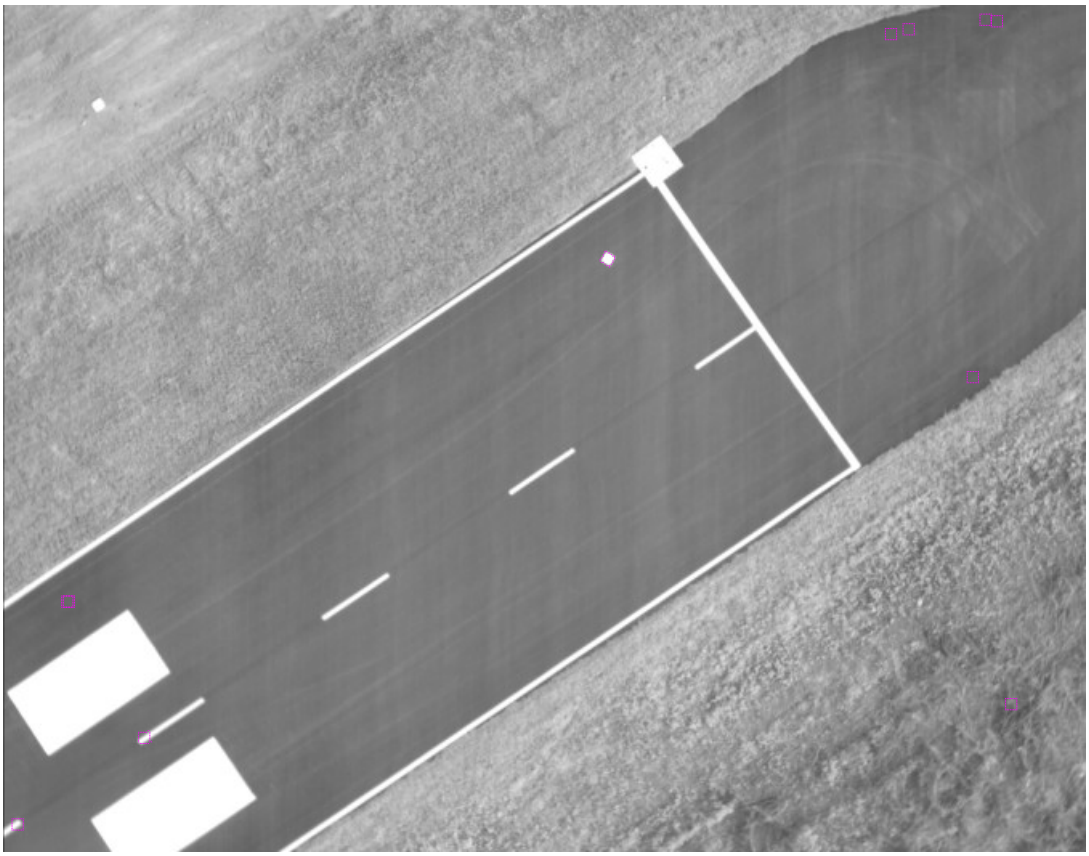

Figure 3.1: Image showing the results of the ATR algorithm using a single decision criterion. We can see that there are lot of false alarms.

We first consider the case where there is only one target type present in the image[2] and then we move on to the general case of an unknown number of targets and multiple target

[2]Most of the images used for the development of the algorithm were synthetic images generated using the 3D modeling tool provided by Augusta Systems.

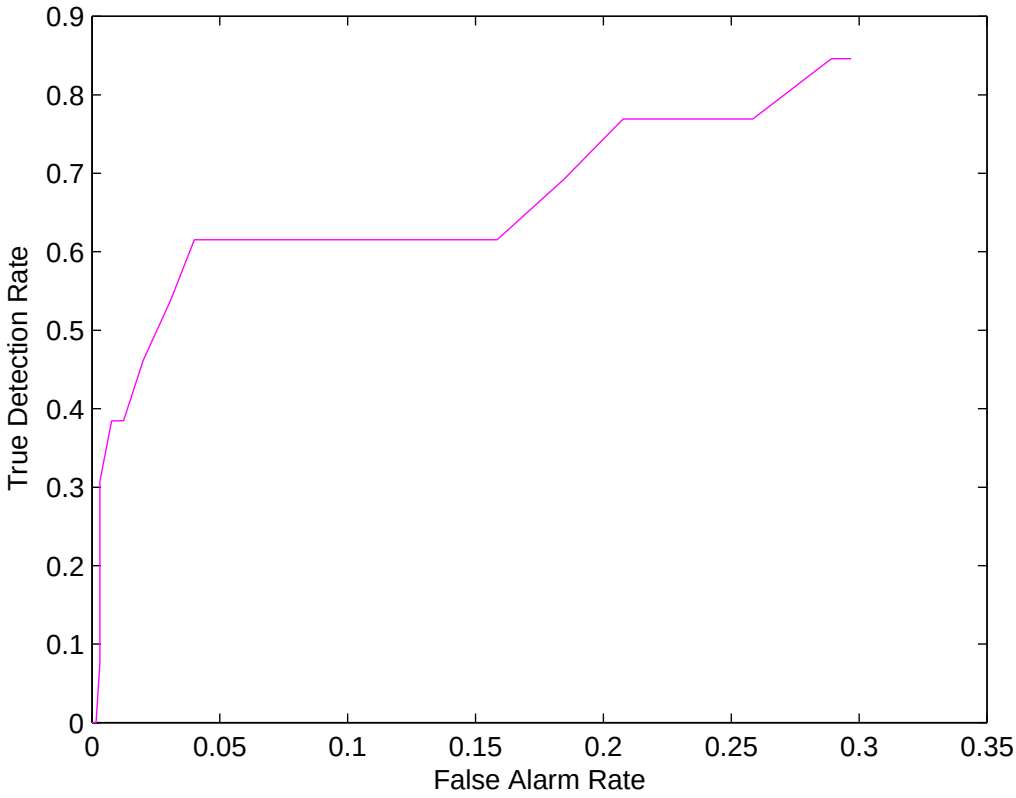

Figure 3.2: ROC curve showing the performance of the ATR algorithm using a single decision metric.

types. Rather than computing hard decisions, the decision metric is *soft* in the sense that it conveys a measure of reliability and hence can be combined with observations from other UAVs.

3.1 Case 1: All the Targets are of the Same Type

Let $\mathbf{\Omega} = \{\theta, \phi, \mathrm{r}\}$ (here θ and ϕ are the azimuth rotation angle and elevation angle respectively, relative to the UAV and r is the range (or distance) from UAV to target) indicate an estimate of the target orientation and $\mathbf{T}(\mathbf{\Omega})$ the template's 2D projection according to this orientation, as described in chapter 2. The ATR algorithm computes two correlation metrics. First it computes $\mathrm{C}(\mathbf{X}', \mathbf{T}(\mathbf{\Omega}))$ which gives the amount of black inside the target object (see Fig. 3.3) and $\mathrm{C}(\bar{\mathbf{X}}, \mathbf{T}'(\mathbf{\Omega}))$ which gives the amount of white outside the projected

target template (see Fig. 3.4).

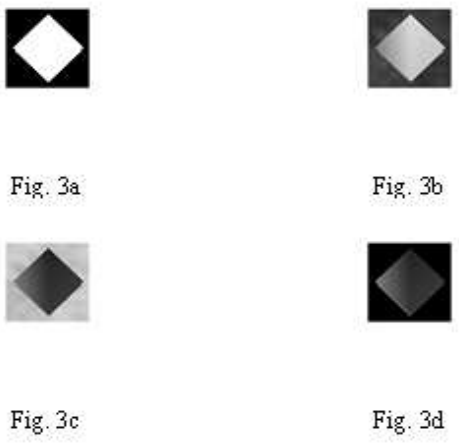

Figure 3.3: Images showing the computation of the first metric $C(\mathbf{X}', \mathbf{T}(\mathbf{\Omega}))$. Fig. 3a shows the template, $\mathbf{T}(\mathbf{\Omega})$, Fig. 3b shows $\mathbf{X}$, Fig. 3c shows $\mathbf{X}'$ and Fig. 3d shows the product of $\mathbf{X}'$ and $\mathbf{T}(\mathbf{\Omega})$. The numerical value of this correlation is 258.2.

3.1.1 Detecting a Single Target

Suppose the target appears once in image $\mathbf{X}$. Then the most likely location and orientation of the target is the argument of:

$$\max_{i,q,\mathbf{\Omega}} \{aZ(\mathbf{X}', \mathbf{T}(\mathbf{\Omega})) + (1-a)Z(\bar{\mathbf{X}}, \mathbf{T}'(\mathbf{\Omega}))\}, \qquad \forall i, q \text{ and } \mathbf{\Omega}. \tag{3.1}$$

Here a and 1-a are the weights for the two decision metrics and a lies between 0 and 1 and Z is as defined in equation (2.5). An empirical analysis suggests picking a to be between 0.35 and 0.7 in order to achieve a reasonable trade-off between probability of detection and false alarm. The $(i,q)^{th}$ element of $\mathbf{X}$ which satisfies the above condition gives the location of the target and $\mathbf{\Omega}$ is its estimated orientation. This information, along with the value of the weighted metric is computed for each target type and stored in an array for fusion with information obtained by processing other frames. Fig. 3.5 is an example where a single target (box) is detected. Fig. 3.6 shows its corresponding correlation peak.

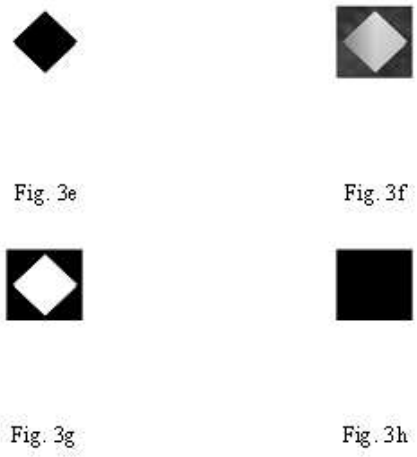

Figure 3.4: Images showing the computation of the second metric C($\bar{\mathbf{X}}, \mathbf{T}'(\mathbf{\Omega})$). Fig. 3e shows $\mathbf{T}'(\mathbf{\Omega})$, Fig. 3f shows $\mathbf{X}$, Fig. 3g shows $\bar{\mathbf{X}}$ and Fig. 3h shows the product of $\bar{\mathbf{X}}$ and $\mathbf{T}'(\mathbf{\Omega})$. The numerical value of this correlation is 0.

3.1.2 Detecting Unknown Number of Targets

In this case the number of targets is not known a priori. We first calculate Z($\mathbf{X}'$,$\mathbf{T}(\mathbf{\Omega})$) (the *soft* decision metric) for all coordinates (i,q) and compare it to a threshold T_1. We declare a match when the threshold is exceeded. This results in a set $\mathcal{S}_1$ where

$$\mathcal{S}_1 \quad = \quad \{(e,f) : z_{(e,f)} \text{ where } \mathbf{Z} = Z(\mathbf{X}', \mathbf{T}(\mathbf{\Omega})) > T_1\}, \tag{3.2}$$

that is (e, f) is the set of all indices of $Z(\mathbf{X}', \mathbf{T}(\mathbf{\Omega}))$ that exceed threshold T_1. This set may contain false alarms. Also, this set may contain coordinate values that may be very close to each other that the targets may appear to be overlapping. Such coordinate values are grouped into clusters, such that the distance between any two coordinates in a cluster is less than the sum of J and K (the dimensions of the template). Then from each of these clusters only the coordinate value which has the maximum value of Z($\mathbf{X}'$,$\mathbf{T}(\mathbf{\Omega})$) is considered and the rest of the coordinate values are discarded. This leaves us with a set of non-overlapping targets. Fig. 3.7 shows the result after the application of first threshold to an image where the object of interest is the box. The figure shows the two boxes getting detected along with

one false alarm due to the big white stripe.

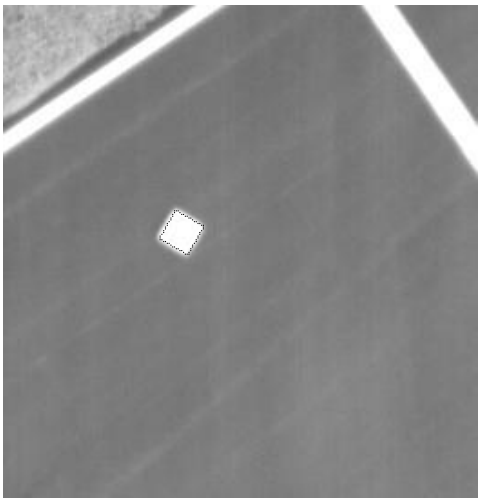

Figure 3.5: Image showing a single target (box) which is detected.

Now the second metric[3] Z($\bar{\mathbf{X}}$,$\mathbf{T}$'($\mathbf{\Omega}$)) is applied to this set of targets i.e. only those targets whose correlation value is greater than a pre-assigned threshold T_2 are considered as true targets. This results in a set

$$\mathcal{S}_2 \quad = \quad \{(c,d) \in \mathcal{S}_1 : z_{(c,d)} \text{ where } \mathbf{Z} = Z(\bar{\mathbf{X}}, \mathbf{T}'(\mathbf{\Omega}) > T_2\}, \tag{3.3}$$

This second metric has the advantage that most of the background in the image is black and potential targets are white. So many of the false alarms (like a light background with a high grayscale value, white stripes which are a part of the airport runway which was found in many images taken from the UAV, objects which are larger than the template,etc.) get filtered here. The second metric does not significantly increase the complexity of the algorithm as it is done only for the targets that have exceeded the threshold T_1. Fig. 3.8 shows the final result after the application of the second threshold to Fig. 3.7. We can see that only true targets get detected with the elimination of the false alarm.

[3]Depending on the contrast between the background and the target, the order in which the metrics are applied can vary. The order of the metrics is chosen in such a way that few number of hits are obtained using the first metric.

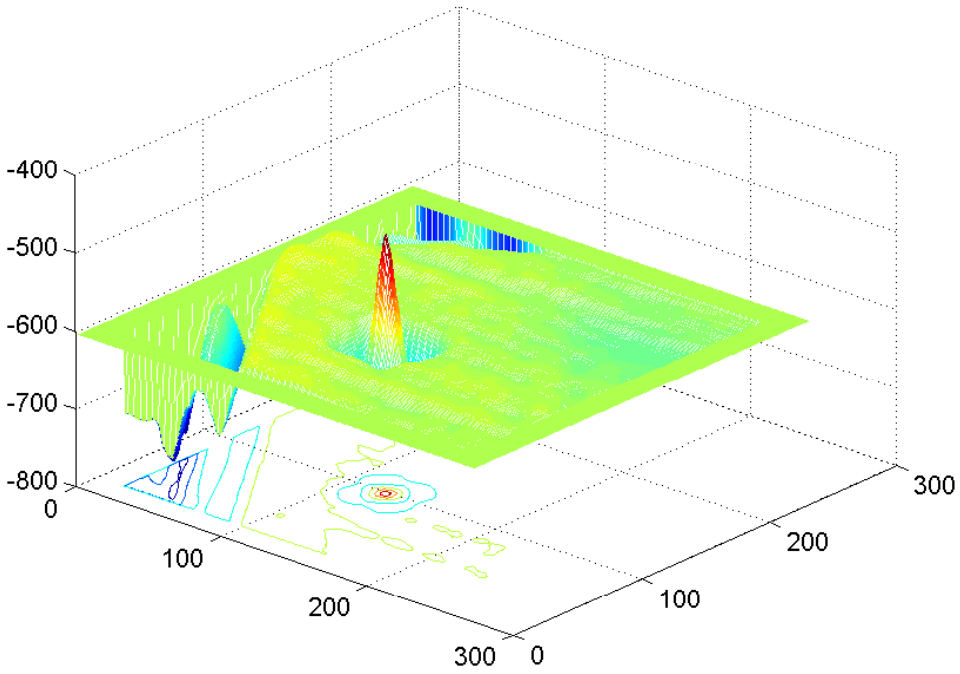

Figure 3.6: Image showing the correlation peak for the target shown in Fig. 3.5.

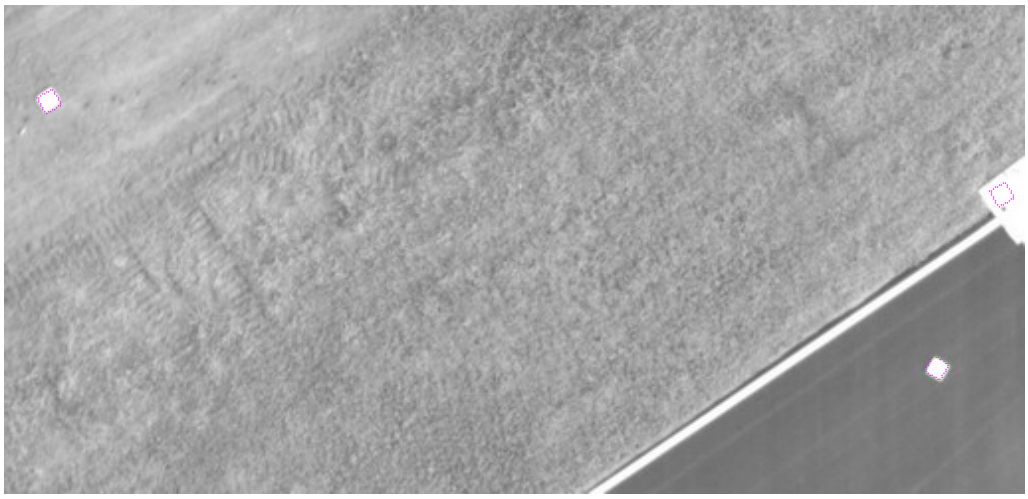

Figure 3.7: Image showing the result after application of the first threshold.

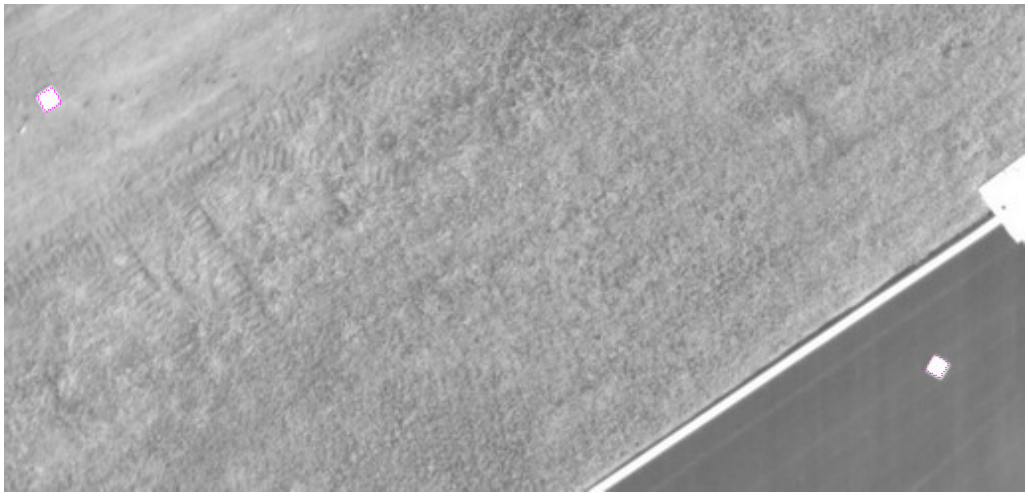

Figure 3.8: Image showing the result after application of the second threshold to Fig. 3.7.

To optimize the performance of the correlation-based ATR system the correlator was written in c, so that it can be called as a c-mex function from Matlab, which improved the execution time considerably.

3.2 Case 2: Targets are of multiple types

We calculate Z($\mathbf{X}'$,$\mathbf{T}(\mathbf{\Omega})$) and Z($\bar{\mathbf{X}}$,$\mathbf{T}'(\mathbf{\Omega})$) for each different target type. Each of the target types have a unique set of threshold ranges for the two metrics. A target is declared to be present at a particular location if the value of the decision statistic at that location exceeds a threshold for that target type.

Fig. 3.9 shows one such case where there are two different target types identified. We see that target type 1 (sphere) appears thrice and target type 2 (truck) appears once in the image. This information, along with the value of the weighted metric is computed for each target type and stored in an array for fusion with information obtained by processing other frames.

Figure 3.9: Image showing the target type 1 (sphere) and target type 2 (truck).

3.3 Complexity Reduction Using Frequency Domain Convolution

In order to find the regions where there is a possibility of finding the targets, we can perform correlation in the frequency domain using equation (2.3) and the result $\mathbf{C}_F(\mathbf{X}, \mathbf{T}(\mathbf{\Omega}))$ is thresholded to get the regions of interest. This operation has the advantage that we will be left with a few regions of interest instead of searching the entire image for target, thus producing a reduction in complexity of the algorithm. This method is translation invariant, but is not rotation invariant. So rotated templates still have to be used to find objects which are rotated. The ATR algorithm described above is then applied only to those regions of interest instead of the entire image. For example, let us consider an image where the target being searched for, is a truck. Fig. 3.10 shows the image being considered, with the truck as the target. Fig. 3.11 shows the result of correlating the image with the target template.

The brighter region is the region of interest. This region is picked up by thresholding the correlation result by an appropriate value.

Figure 3.10: Image showing the target (truck).

3.4 Performance Evaluation

The following are some sample results (Fig. 3.13, Fig. 3.14, Fig. 3.15) obtained using the correlation-based ATR algorithms.

The performance of the correlation-based ATR algorithm is shown in the ROC curves (Fig. 3.16). The values for the ROC curve were calculated as follows: A set of test images (see Appendix A) were taken and correlation-based ATR algorithm was applied. One of the two thresholds was kept constant while the other threshold was varied uniformly and the number of true detections and false alarms were recorded for each of the test images as the threshold was varied. As we have a minimum spacing between the detected objects, there is a limit to the maximum number of false alarms. False alarm rate was then calculated as the number of false alarms divided by the maximum number of false alarms. Then the detection rate vs. the false alarm rate was plotted for each of the varied threshold values.

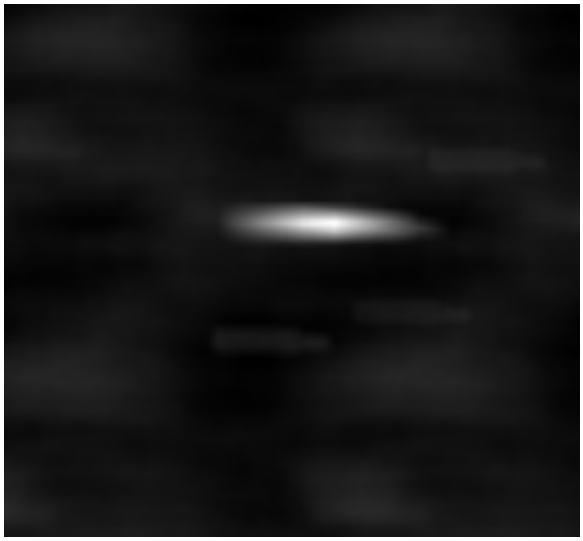

Figure 3.11: Image showing the regions of interest in white. These regions are obtained by first performing the correlation against the truck as the template in frequency domain and thresholding the resulting image.

Figure 3.12: Final image where target (truck) is correctly identified.

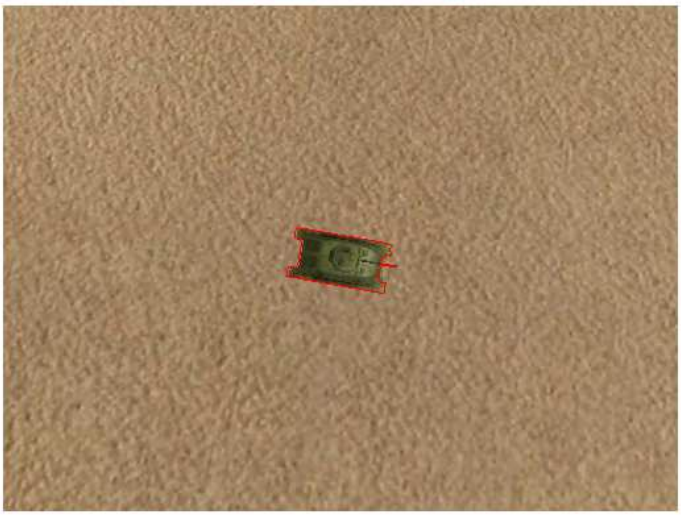

Figure 3.13: Image showing the detected target (tank).

Figure 3.14: One more image showing the detected target (tank).

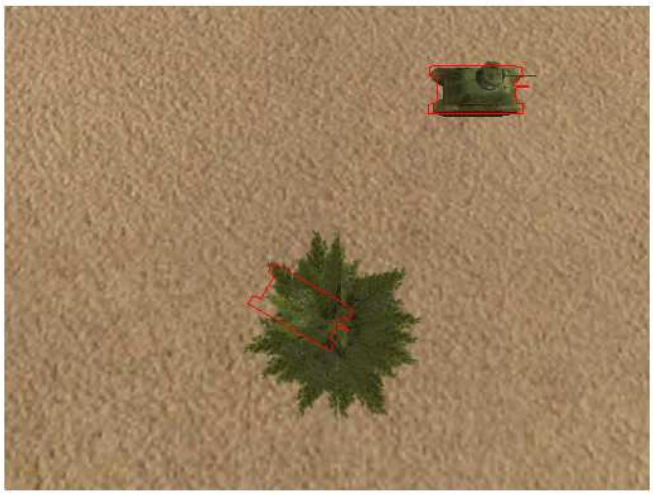

Figure 3.15: Final image where target (tank) is correctly identified along with a false alarm.

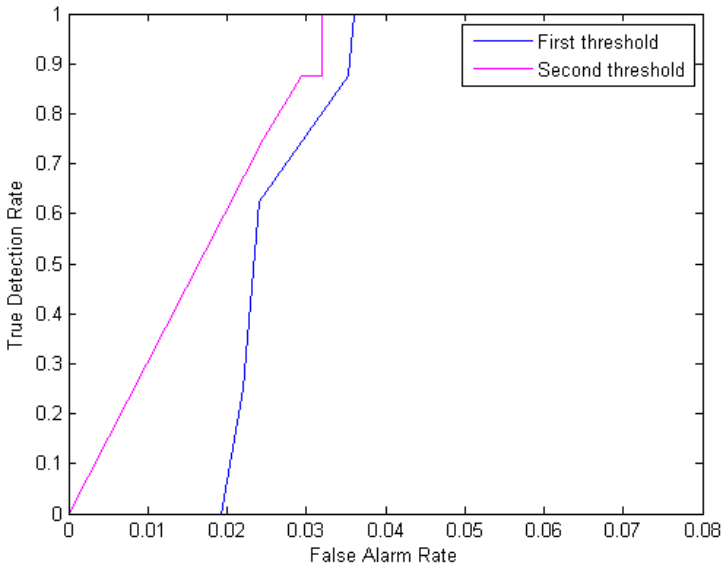

Figure 3.16: ROC curve showing the improvement in performance of the ATR algorithm by the inclusion of the second metric.

Image	No. of targets	Threshold 1	Threshold 2
Image 1	0	267.8	N/A
Image 2	0	283.5	N/A
Image 3	0	272.1	N/A
Image 4	0	288.32	N/A
Image 5	0	288.75	N/A
Image 6	0	288.75	N/A
Image 7	1	278 - 284.1	243 - 276
Image 8	1	275.6 - 281.6	234 - 288
Image 9	1	276.1 - 282.5	264 - 288
Image 10	1	280 - 282	168 - 282
Image 11	1	279 - 281.2	118 - 283
Image 12	2	277 - 279.3	249 - 285
Image 13	3	275.8 - 277.7	268 - 287

Table 3.1: Threshold values for images with and without objects of interest.

This ATR algorithm was tested for its robustness by testing it over a set of test images (see Appendix A), with and without targets. The minimum and maximum range of the thresholds for which all the targets are detected with no false alarms, were determined (see Table 3.1).

Chapter 4

Training-Based ATR

In the training based approach, object detection is done by training a statistical model (classifier). Statistical model-based training requires positive samples i.e., images containing the object class of interest and multiple negative samples i.e., images that do not contain object class of interest [2]. Thus, the training set is made up of positive and negative samples. The training process consists of extracting different features from the training samples and selecting distinctive features that can be used to classify the object. This information is put into the statistical model parameters. If an object is not detected or the training classifier gives a false alarm, adjustment is made easily by adding identical positive or negative samples to the training set.

After a survey of available tools, the Intel OpenCV (Open Source Computer Vision) library was selected for use in training based ATR. The OpenCV is a free, open source collection of computer vision routines geared mainly towards human-computer interactions, robotics, security, and other vision applications [2]. The Intel OpenCV library contains low-level and high-level APIs (Application Programming Interfaces) for object detection. This training based method uses simple Haar-like features[1] and a cascade of boosted tree classifiers [23, 24, 25] as a statistical model for object detection. The method followed for object detection is described in the following sections (for command line arguments see Appendix B).

[1]They are called so because they are reminiscent of Haar basis functions (named after a Hungarian mathematician, Alfred Haar) and are computed the same way as the Haar wavelet transforms.

4.1 Samples Creation

To train the classifier, a set of training samples are required. Training samples are of two types: negative samples that do not contain the object of interest and positive samples that contain object of interest. Typically the positive samples set consists of a few hundreds of sample views of the object of interest which are scaled to the same size. Negative samples set consists of arbitrary images of the same size. These images can be any non-object of interest images (example: images obtained from the web that do not contain the object representation.). The richer the training set, the better the performance of the classifier will be.

Createsamples utility is used for creating positive samples. They may be created from single object image (In this case, a large set of positive samples is obtained from the given object image by arbitrarily rotating, changing the color of object and placing the object on different backgrounds) or from collection of previously marked up images.

4.2 Training

After samples creation, the next step is training of classifier. This is done by using the *haartraining* utility. Haar-like features and a large set of very simple weak classifiers are principal to this approach [23, 26]. These weak classifiers use a single distinctive feature to classify the object as object of interest or non-object of interest. Each feature is represented by (i) a template describing shape of the feature, (ii) corresponding position in the region of interest, (iii) the feature size (scale factor). The Haar-like features used for this approach are shown in Fig. 4.1

A Haar feature consists of two or three joined black and white rectangles of appropriate weights[2] of opposite signs. The Haar feature's value is the weighted sum of two values: The sum of the pixel values in the black rectangular region and the sum over the whole feature area (both black and white rectangles). The opposite signs of these two weights are normalized by the fact that their absolute values are inversely proportional to the areas of

[2]Black rectangles have negative weights and white rectangles, positive weights.

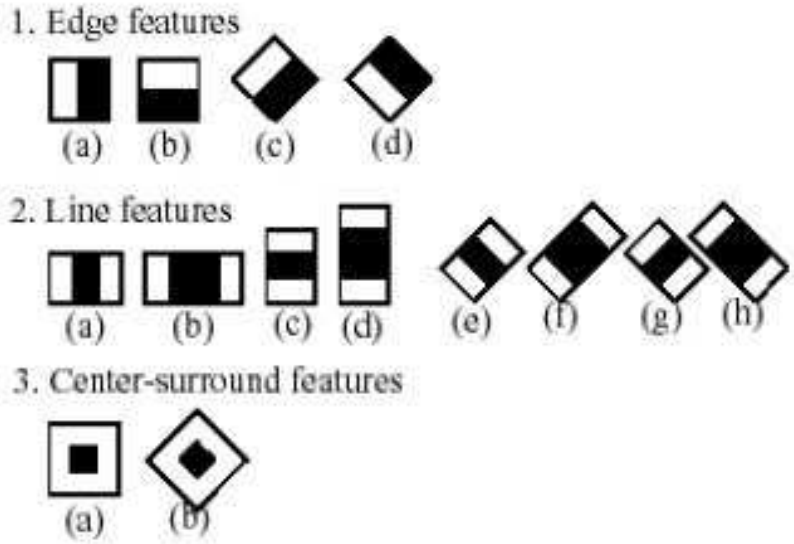

Figure 4.1: Extended set of Haar-like features (figure from [2]).

the rectangles.

Each week classifier cannot detect the desired object but rather it reacts to some simple feature in the image that may relate to the object. A complex and robust cascade (Cascade here means that the classifier is built using several weak classifiers (or stages). These weak classifiers are applied to a region of interest until a stage is reached where the object being considered is rejected or it passes all the stages)classifier is built out of multiple weak classifiers using a procedure called boosting (By boosting we mean that the classifiers at each stage are complex and are built out of basic classifiers using the following boosting methods: Gentle Adaboost, Discrete Adaboost, Real Adaboost and Logitboost) [23].

4.3 Performance Evaluation

The performance of the classifier can be evaluated using the *performance* utility. This function finds rectangular regions in the given image that are likely to contain objects of interest,and outputs those regions as a set of rectangles. It takes a collection of marked up test images, applies the classifier and outputs the performance, i.e. number of found objects of interest, number of missed objects of interest, number of false alarms etc. Detection is

done by sliding a search window across the whole image and checking whether an image region at a certain location looks like the object of interest. The classifier is designed in such a way that it can find objects of interest at different scales. Fig. 4.2, Fig. 4.3 and Fig. 4.4 are a few sample images showing the detected object (tank).

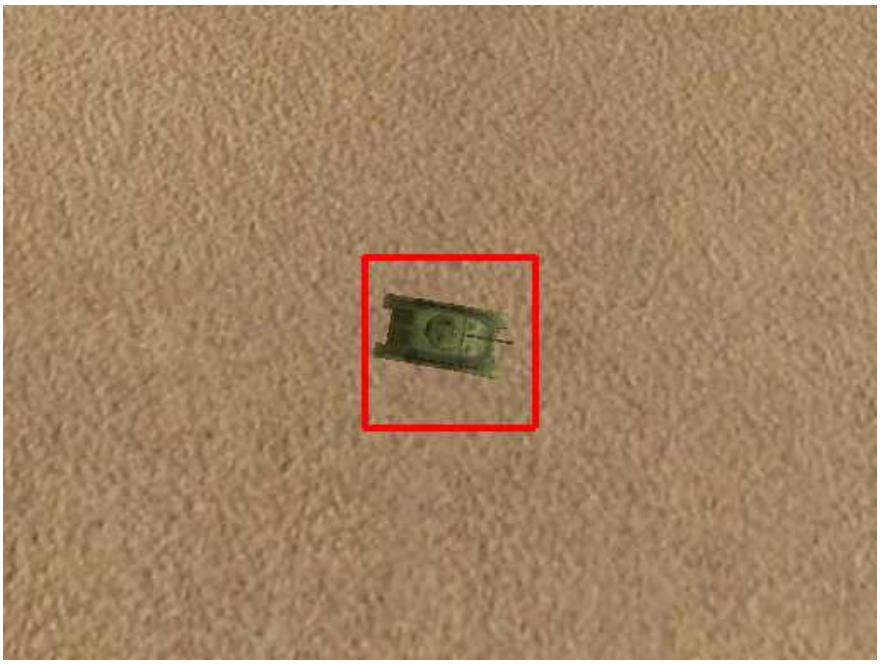

Figure 4.2: Image where the desired object (tank) is detected.

Fig. 4.5 show the ROC curve [1] for the scenario 1 described in Appendix C.

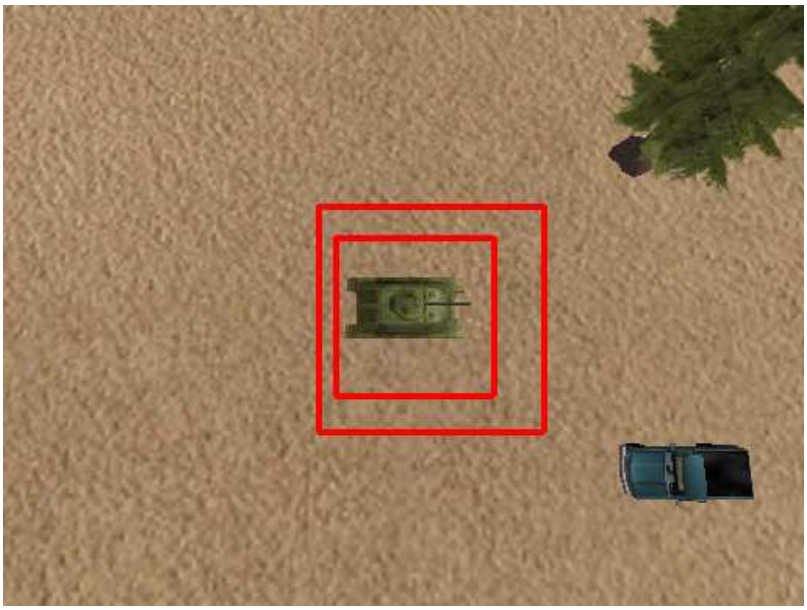

Figure 4.3: One more sample image where the desired object (tank) is detected.

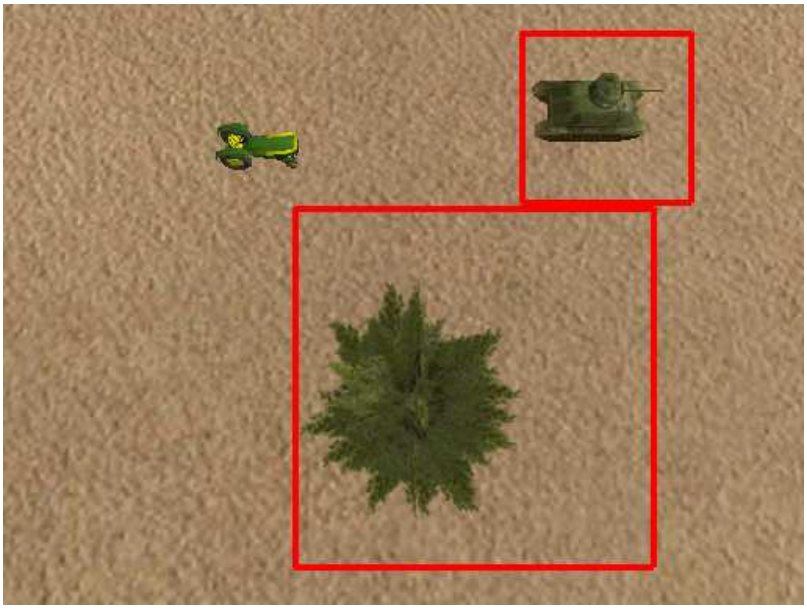

Figure 4.4: Image where the desired object (tank) is detected along with the tree (false alarm)

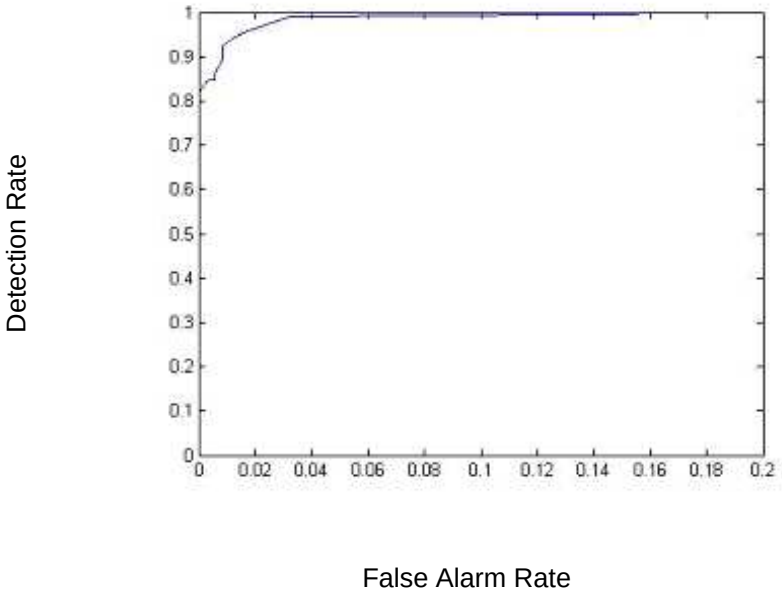

Figure 4.5: ROC curve showing the detection capabilities under the training Scenario 1 described in Appendix C.

Chapter 5

Correlation using Multiple Images

Correlation across multiple frames is an useful tool for detection and recognition of targets which are difficult to detect using a single image. Multiple images used for this purpose can be taken from the same UAV or from multiple UAVs. Because of the latter possibility, it is important that the amount of data shared across frames be minimized, given that this data must ultimately be communicated over a low bandwidth wireless link. In this thesis, two methods for correlating such multiple images are discussed. Here we are assuming that the images considered for this purpose are taken from a single UAV flying from left to right at a constant velocity so that each image is displaced by the same number of columns. Also the UAV is assumed to be flying at a constant altitude.

5.1 Method 1

This method is explained using an example. Here we are considering four different images of the object of interest (sphere). Each image is displaced from its next image by D units (here D is one-fourth the width of the image) in the x-direction. There are three instances of the object of interest (sphere) and one non-object of interest (a box) which is similar to the object of interest.

The correlation-based ATR algorithm described in chapter 3 is performed over the first image. The result of the algorithm is shown in Fig 5.1. We can see from the figure that the box which gets detected along with the spheres has correlation scores almost as high as the

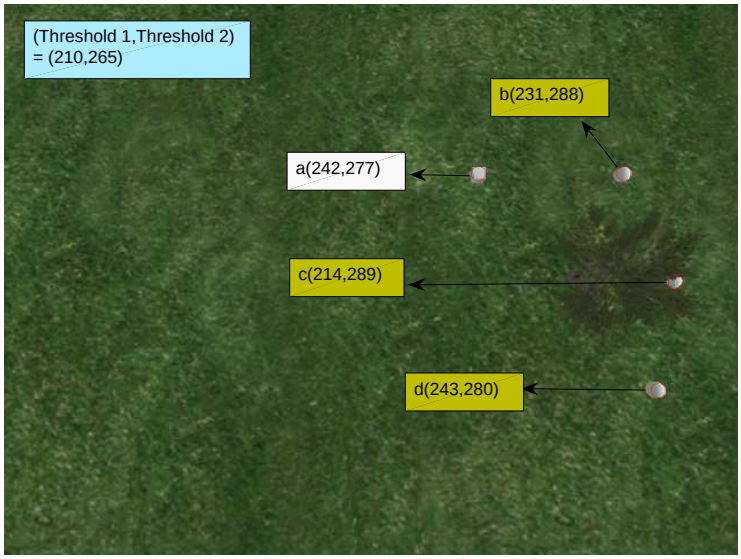

Figure 5.1: First image showing the identified objects. We can see from the figure that there is a false alarm (object a: box.)

spheres b and d while the sphere c (which is partially obscured by the tree) has a smaller first correlation score than the box.

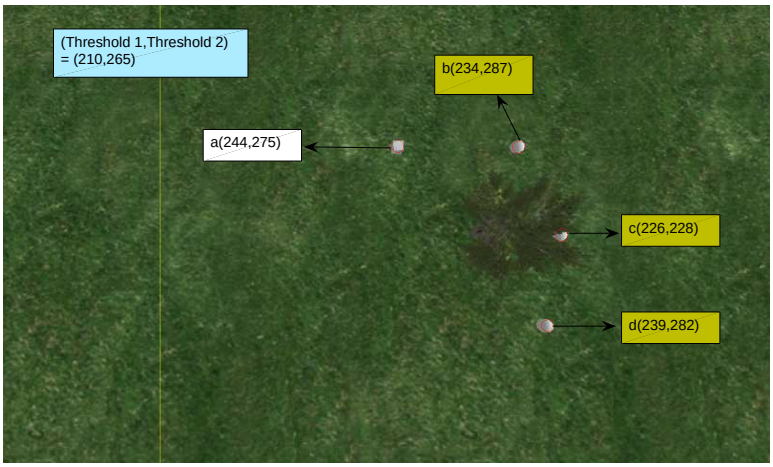

Figure 5.2: Image showing the correlation result after the merging of information from first and second image.

Now correlation is performed on the second image (which is displaced from the first image

by D units), but instead of performing correlation operation for the entire image, we do it only for the new region and for those points in the second image which correspond to the four objects that exceeded the thresholds in the first image. Once we get the correlation scores for the second image, the correlation scores from the first and second images are averaged over the two images and compared against a preset threshold. For this example a threshold pair (210,265) was chosen because it results in false alarms when a single image is used, yet allows false alarms to be suppressed when information is fused across multiple images. Only those objects which satisfy the threshold condition are considered for correlation with the third image. The result of correlation across the first two images is shown in Fig 5.2. This process is repeated for the next two images the results of which are shown in Fig. 5.3 and Fig. 5.4. It can be seen from the figures that as the number of images for information fusion increases, the correlation scores for the objects of interest keep increasing while the non-object of interest scores keep decreasing and reach a point where they no more satisfy the threshold condition (see Fig. 5.3). This leads to the successful detection of only the objects of interest.

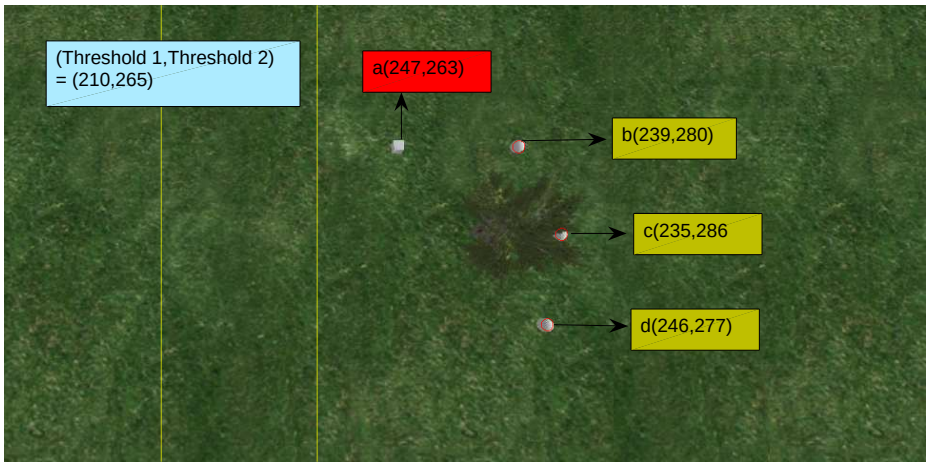

Figure 5.3: Image showing the correlation result after the merging of information from first, second and third image. Here the false alarm is fixed.

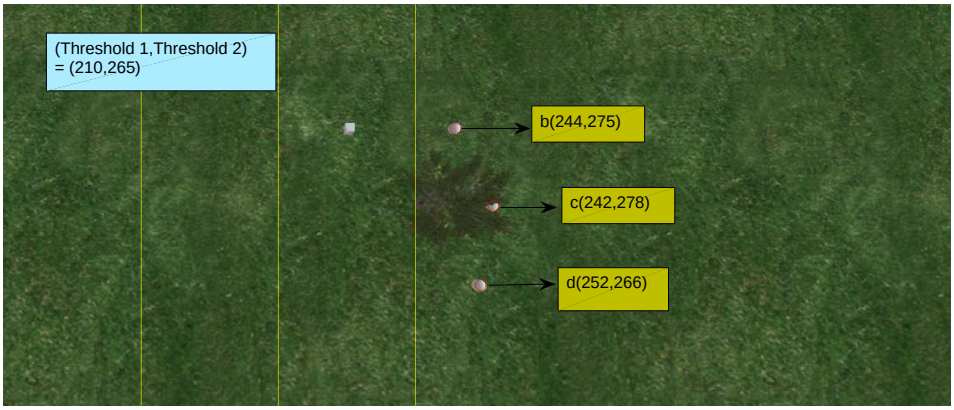

Figure 5.4: Image showing the correlation result after the merging of information from first, second, third and fourth image.

5.2 Method 2

The geometry of the 3D training tool provided by Augusta Systems was not as expected, which proved to be a major drawback in the performance of method 1. Also we needed to keep track of the position and location of each object in the previous method, which is not the case with method 2 proposed here. In this method 2, the position of each object relative to the other objects, is considered. This method is valid as long as the objects maintain their relative positions. This method is explained using four images from four different views consisting of three instances of object of interest (sphere) and one non-object of interest (box). Correlation-based ATR algorithm is performed on the first image, which results in the detection of the three spheres and the box (false alarm) as seen in Fig. 5.5. These four objects are labeled in the order they appear if the image is scanned from top to bottom (i.e., based on their x coordinate values). If more than one object has the same x coordinate value then their y coordinates are compared, so the one with lesser y coordinate value will have the priority in labeling. Thus the object which first appears in the image while scanning the image from top to bottom will be labeled object *1* and the next object is labeled object *2* and so on. Let this set of objects from first image be called *set a*.

Then ATR is performed on the second image. Once we have the (x,y) coordinate values

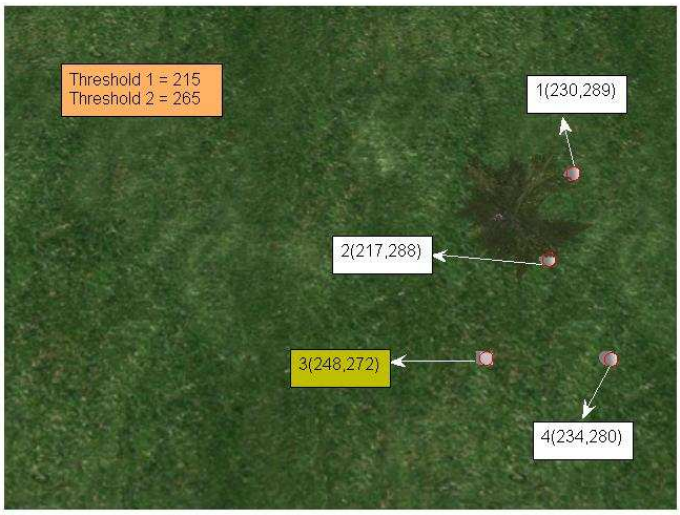

Figure 5.5: First image showing the identified objects with labeling. We can see from the figure that there is a false alarm (object *3*: box.)

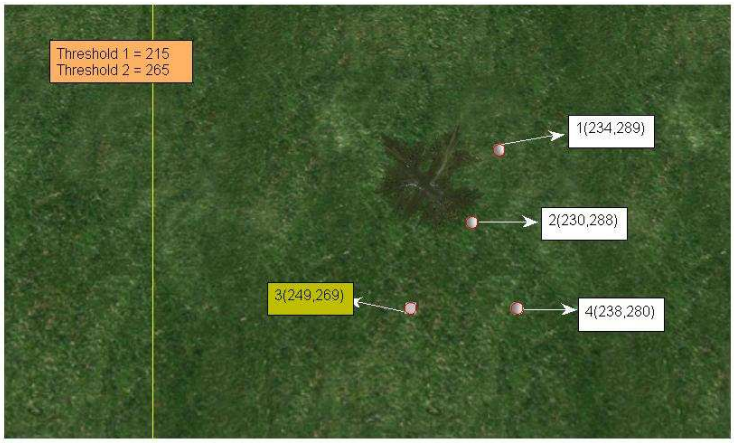

Figure 5.6: Image showing the correlation result and labeling after the merging of information from first and second image.

of the objects from second image (lets call it *set 2*), we compare the x coordinate values of the *set 2* to the x coordinate values of *set 1* (as the camera is assumed to be moving from

left to right, the x coordinate values will remain the same but the y coordinate values of the objects will change). The ones that match are given the same label as the ones in the set one. Also the correlation values of this matching objects are averaged (see Fig. 5.6). The same process is done on the third and fourth image. The results are shown in Fig. 5.7 and Fig. 5.8.

As seen from Fig. 5.6 and Fig. 5.7, object *3* appears in Fig. 5.6 but does not get detected in Fig. 5.7 as it does not satisfy the threshold condition. So the label object *3* is dropped in the subsequent images on which ATR is performed. This method too leads to the successful detection of the objects of interest which may not be obvious in a single frame.

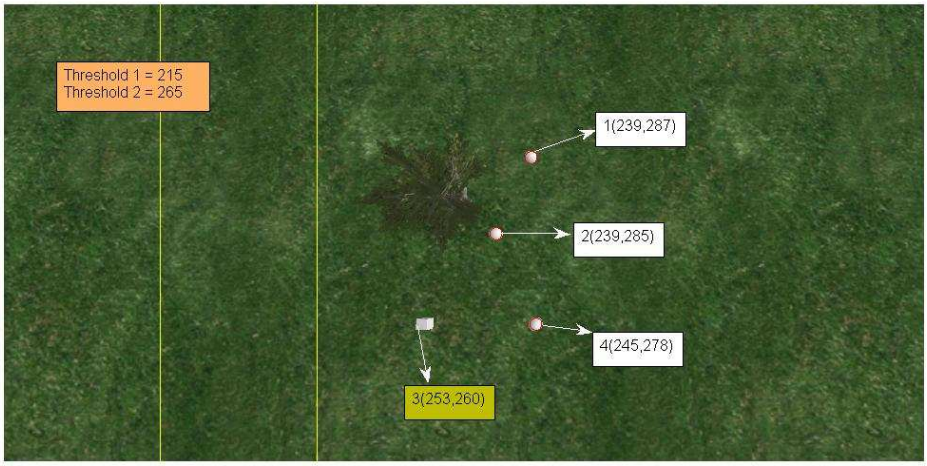

Figure 5.7: Image showing the correlation result and labeling after the merging of information from first, second and third image. Here the false alarm is removed.

5.3 Comparison of the Two Methods

As discussed in Section 5.2, method 1 is pose (or geometry) dependent. It takes into account the position and location of the objects with respect to the image coordinates. In this aspect, method 2 has an advantage over method 1 in the sense that as long the objects

Figure 5.8: Image showing the final correlation result and labeling after the merging of information from first, second, third and fourth image.

maintain their relative positions, method 2 need not keep track of the locations of the objects with respect to the image coordinates i.e., method 2 is geometry-independent. But method 1 provides reduction in complexity since correlation has to be performed only on the new regions and at the locations which correspond to potential targets in the previous image. In method 2 correlation has to be performed on the entire image. This considerably increases the computational time. For the examples considered in this chapter, method 1 took 5 sec for processing the data while method 2 took 8 sec. In spite of these drawbacks, both the methods are found to be good in suppressing false alarms.

Chapter 6

Conclusions

6.1 Summary and Conclusions

In this thesis, we looked at the development and performance of the correlation-based ATR algorithm. We dealt with the objectives, applications and the problems associated with ATR in the chapter 1. The notation and the image operators (unary and binary) used through out the thesis were introduced. We then considered the simple case of one object of interest and then extended the algorithm to a more general case of unknown number of object of interest and multiple classes of objects of interest. The algorithm finds the correlation between the input image and the template and declares a target where ever there are correlation peak. Then we looked at an existing training-based ATR algorithm which utilizes Haar-like features for training. Intel OpenCV library was used for this purpose. The results of the training-based ATR approach were compared with the results of correlation-based ATR algorithm. We also looked at correlation across multiple images which is considered to be useful when the objects of interest are not very obvious using a single frame.

It was observed that the overall performance of the correlation-based approach matched that of the training-based approach but the former method performed better in the case of simpler objects (i.e., with less number of features) and the latter method performed better when targets richer in features were considered. Correlation-based method worked well in idealized conditions. One drawback of this method is that it relies on high contrast between the target and the background. The training-based method has better computational

efficiency than the correlation-based method.

6.2 Future Work

In this thesis we focused mainly on lighter objects on darker background. The correlation-based ATR algorithm has problems detecting objects of darker shade and objects with low contrast. Modifications could be made to the algorithms to accommodate all types of targets and backgrounds. Also the algorithm could be modified to have only one decision criteria instead of two and better computational efficiency (speed). The correlation-based algorithm could also be modified to work well even when the assumptions made in this thesis are relaxed. Combining the two algorithms (correlation-based and training-based) efficiently could lead to a better algorithm. In this thesis we considered grayscale images. The algorithm can be extended to explore color images. Also one of the assumptions of this thesis, is that the targets are considered to be stationary. The algorithm can be modified to accommodate mobile targets. Also provision can be made in the algorithm to come with the absolute location (with respect to the region of interest) of the target.

References

[1] X. Chen, "Personal correspondence," May 5 2006.

[2] G. Bradski, A. Kaehler, and V. Pisarevsky, "Learning-based computer vision with OpenCV," *Intel Technology Journal*, vol. 09, pp. 1–14, May 19 2005.

[3] K. Augustyn, "A new approach to automatic target recognition," *IEEE Trans. AES*, vol. 28, no. 1, pp. 105–114, Jan 1992.

[4] K. Murphy, A. Torralba, and W. Freeman, "Using the forest to see the trees: A graphical model relating features, objects and scenes," in *Neural Information Processing Systems Foundation (NIPS)*, 2003.

[5] P. Viola, and M. Jones, "Robust real-time object detection," *Intl. J. Computer Vision*, vol. 57, no. 2, pp. 137–154, 2004.

[6] Y. Amit, *2D Object Detection and Recognition, Models, Algorithms, and Networks*, MIT Press, 2002.

[7] S. Agarwal, A. Awan, and D. Roth, "Learning to detect objects in images via a sparse, part-based representation," *IEEE Tran. Pattern Anal. Machine Intell.*, vol. 26, no. 11, pp. 1475–1490, 2004.

[8] Z.W. Tu, X.R. Chen, A.L. Yuille, and S.C. Zhu, "Image parsing: Unifying segmentation, detection and recognition," *Intl. J. Computer Vision*, vol. 63, no. 2, pp. 113–140, July 2005.

[9] J.F. Gilmore, "Knowledge-based target recognition system evolution," *Opt. Eng.*, vol. 30, no. 5, pp. 557–570, May 1991.

[10] A. Dargar, A. Kamel, G. Christensen, and K. Nygard, "An agent-based framework for UAV collaboration," *Proc. ISCA International Conference on Intelligent Systems, Boston, MA*, pp. 54–59, 2002.

[11] P. Gaudiano, B. Shargel, E. Bonabeau, and B. Clough, "Control of UAV swarms: What the bugs can teach us," *in 2nd AIAA Unmanned Unlimited Systems, Technologies, and Operations Conference, San Diego, CA*, Sep. 15-18 2003.

[12] H. Parunak, S. Brueckner, and J. Odell, "Swarming coordination of multiple UAVs for collaborative sensing," *in 2nd AIAA Unmanned Unlimited Systems, Technologies, and Operations Conference, San Diego, CA*, Sep. 15-18 2003.

[13] J.A. O'Sullivan, S.P. Jacobs, M.I. Miller, and D.L. Snyder, "A likelihood-based approach to joint target tracking and identification," in *Proc. Conf. Recc. 27th Asilomar Conf. Signals, Syst., Comput.*, A.Singh, Ed., Los Alamitos, CA, Nov 1993, IEEE Computer Society, pp. 290–294.

[14] C. K. Sword, M. Smman, and E. W. Kamen, "Multiple target angle tracking using sensor array outputs," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 26, no. 2, pp. 367–373, 1990.

[15] B. Bhanu, "Automatic target recognition: State of the art survey," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 22, no. 4, pp. 364–379, 1986.

[16] J. Verly, R. Delanoy, and C. Lazott, "Principles and evaluation of an automatic target recognition system for synthetic aperture radar imagery based on the use of functional templates," *SPIE Proc. Automatic Target Recognition III*, vol. 960, no. 1, pp. 57–71, Apr. 1993.

[17] W. E. L. Grimson, *Object Recognition by Computer*, MIT Press, 1990.

[18] R. Schalkoff, *Pattern Recognition: Statistical, Structural and Neural Approaches*, Wiley, New York, 1992.

[19] F. Sadjadi, and M. Bazakos, "Object tracking using sensor fusion," in *Trans. of the 4th Army Conference on Applied mathematics and Computing*, 1987, pp. 1241–1247.

[20] B. Bhanu, and T. Jones, "Image understanding research for automatic target recognition," *IEEE AES Mag.*, vol. 8, pp. 15–23, Oct. 1993.

[21] X. Li, "Introduction to digital image processing," Course Notes, WVU, Spring 2004.

[22] M. Nixon and A. Aguado, *Feature Extraction and Image Processing*, 1st ed., Oxford: Newnes, 2002.

[23] A. Kuranov, R. Lienhart, and V. Pisarevsky, "Empirical analysis of detection cascades of boosted classifiers for rapid object detection," *in Proc. 25th German Pattern Recognition Symposium (DAGM), Magdeburg, Germany*, Sep. 2003.

[24] K. Murphy A. Torralba and W. Freeman, "Sharing features: Efficient boosting procedures for multiclass object detection," in *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2004.

[25] S. C. Zhu, "Statistical modeling and conceptualization of visual patterns," *IEEE Tran. Pattern Anal. Machine Intell.*, vol. 25, no. 6, pp. 691–712, 2003.

[26] R. Lienhart, and J. Maydt, "An extended set of Haar-like features for rapid object detection," in *IEEE Int. Conf. Image Proc.*, Sep. 2002, vol. 1, pp. 900–903.

Appendix A

Sample Results for the Correlation-Based ATR Algorithm

The following images show the results of application of the correlation-based ATR. Here the object of interest is a white box.

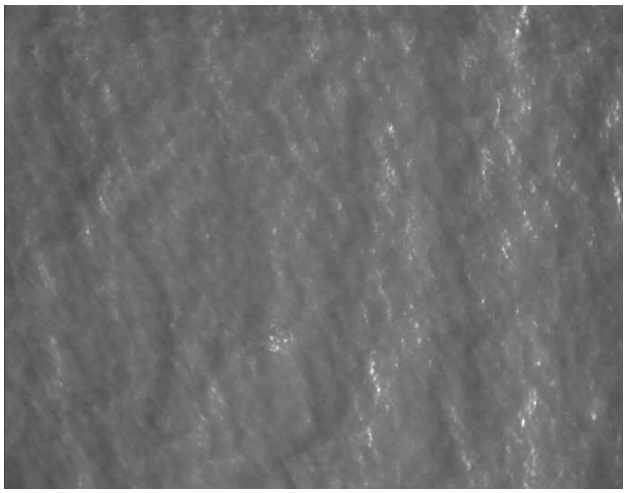

Figure A.1: Image 1

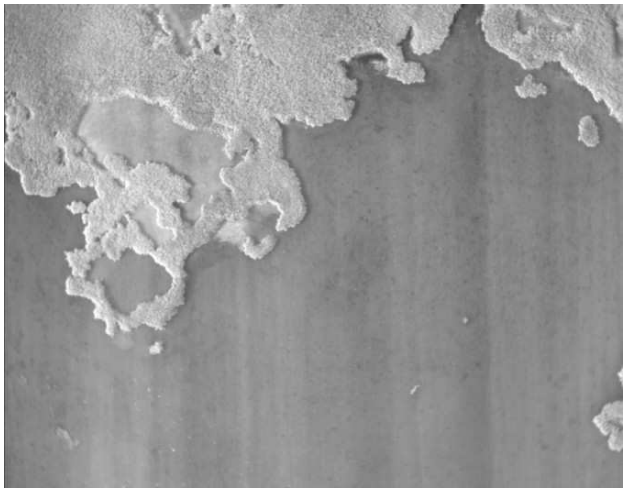

Figure A.2: Image 2

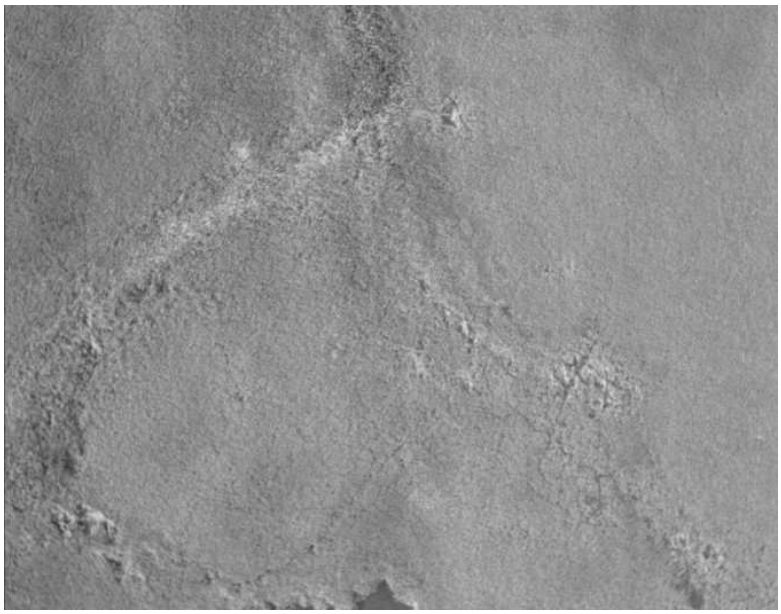

Figure A.3: Image 3

Figure A.4: Image 4

Figure A.5: Image 5

Figure A.6: Image 6

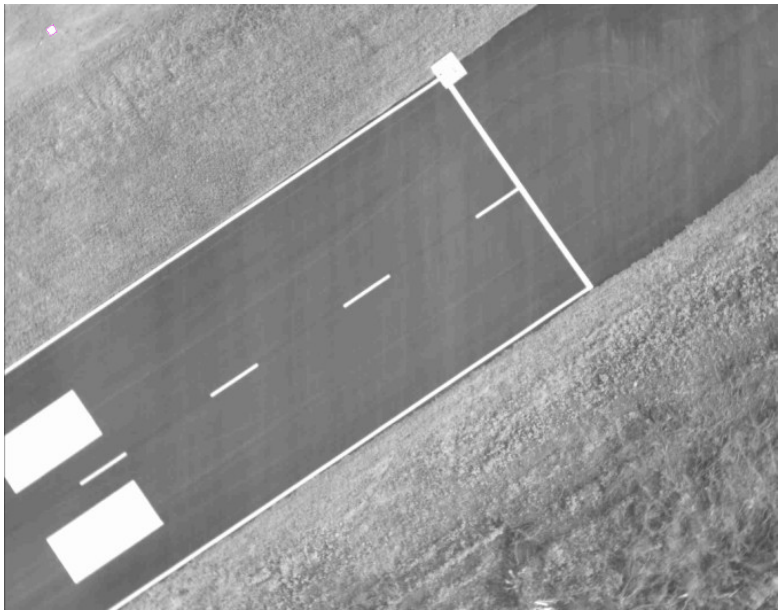

Figure A.7: Image 7

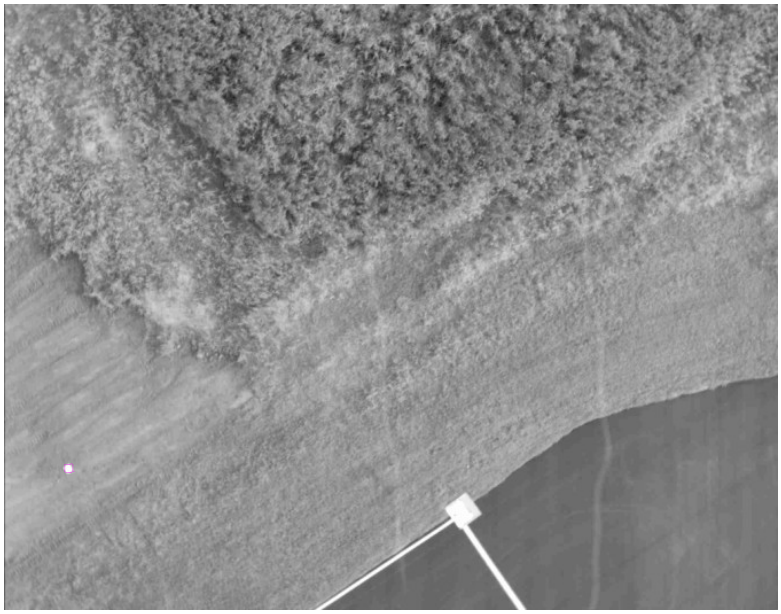

Figure A.8: Image 8

Figure A.9: Image 9

Figure A.10: Image 10

Figure A.11: Image 11

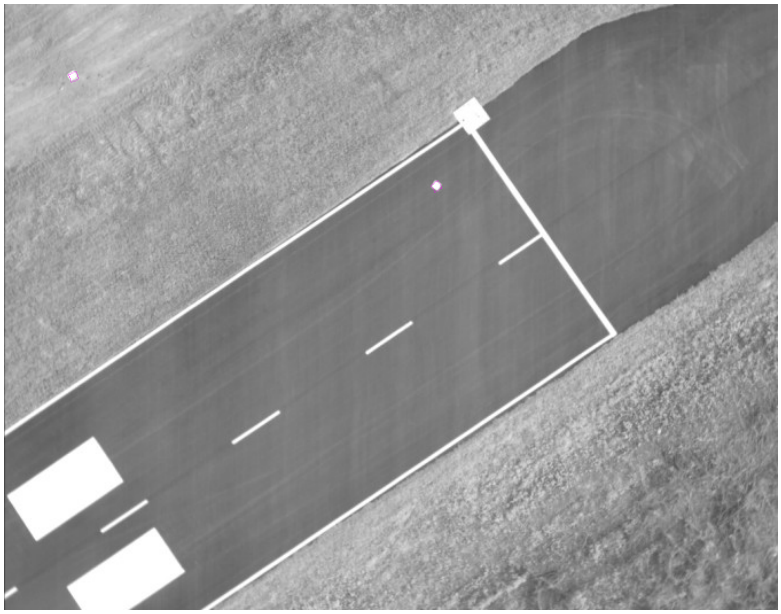

Figure A.12: Image 12

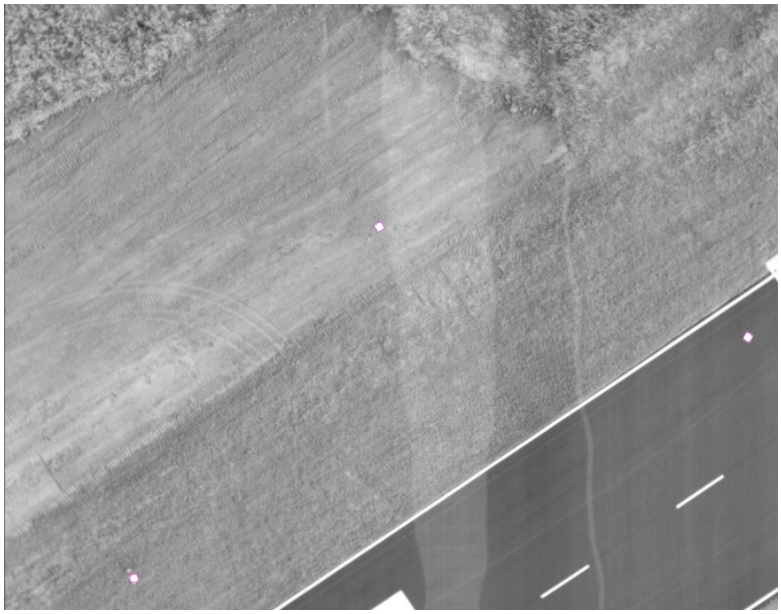

Figure A.13: Image 13

Appendix B

Command Line Arguments for the OpenCV Utilities used in the Training-Based Approach

The OpenCV utilities used for object detection are described below:

Createsamples Utility: Two types of samples are needed for training: positive samples and negative samples. Positive samples contain object of interest. Negative samples contain non-object of interest representations. Negative samples are passed through a manually created background description file, where each text line contains the filename (relative to the directory of the description file) of negative sample image. *createsamples utility* is used to create positive samples for training the classifier. They can be created from single object image (In this case, a large set of positive samples is obtained from the given object image by arbitrarily rotating, changing the color of object and placing the object on different backgrounds) or from a set of previously marked up images. The amount and range of randomness is controlled by command line arguments which are as follows:

-**vec** (vec file name): name of the output file containing the positive samples for training

- **img** (imagefile name): source object image (e.g., a company logo)

- **bg** (background file name): background description file; contains a list of images

into which randomly distorted versions of the object are pasted for positive sample generation

- **num** (number of samples): number of positive samples to generate

- **bgcolor** (background color): background color (currently grayscale images are assumed); the background color denotes the transparent color. Since there might be compression artifacts, the amount of color tolerance can be specified by bgthresh. All pixels between bgcolor-bgthresh and bgcolor+bgthresh are regarded as transparent.

- **bgthresh** (background color threshold)

- **inv**: if specified, the colors will be inverted

- **randinv**: if specified, the colors will be inverted randomly

- **maxidev**: maximal intensity deviation of foreground samples pixels

- **maxxangle**: maximum x rotation angles in radians

- **maxyangle**: maximum y rotation angles in radians

- **maxzangle**: maximum z rotation angles in radians

-**show**: if specified, each sample will be shown. Pressing Esc will continue creation process without samples showing. Useful debugging option.

-**w** (sample width): width (in pixels) of the output samples

- **h** (sample height): height (in pixels) of the output samples

If a collection of images is used to create positive samples -**info** argument should be specified instead of -**img**:

info (collection file name): description file of marked up images collection

Haartraining utility: The command line arguments for *haartraining* are as follows:

- **data**: directory name in which the trained classifier is stored

- **vec** (vec file name): file name of positive sample file (created by *createsamples* utility or by any other means)

- **bg** (background file name): background description file

- **npos** (number of positive samples): number of positive samples used in training of each classifier stage. Reasonable values are npos = 7000

- **nneg** (number of negative samples): number of negative samples used in training of each classifier stage. Reasonable values are nneg = 3000

- **nstages**: number of stages to be trained

- **nsplits** (number of splits): determines the weak classifier used in stage classifiers. If 1, then a simple stump classifier is used, if 2 and more, then CART classifier with number of splits internal (split) nodes is used

- **mem**: Available memory in MB for precalculation. The more memory you have the faster the training process

- **sym** (default)

- **nonsym**: specifies whether the object class under training has vertical symmetry or not. Vertical symmetry speeds up training process.

- **minhitrate**: minimal desired hit rate for each stage classifier. Overall hit rate may be estimated as (min-hit-rate $^{number-of-stage}$

- **maxfalsealarm**: maximal desired false alarm rate for each stage classifier. Overall false alarm rate may be estimated as (max-false-alarm-rate$^{number-of-stages}$)

- **weighttrimming**: specifies wheter and how much weight trimming should be used. A decent choice is 0.90.

- **eqw**

- **mode** (BASIC (default) — CORE — ALL)

selects the type of haar features set used in training. BASIC use only upright features, while ALL uses the full set of upright and 45 degree rotated feature set.

- **w** (sample width),

- **h** (sample height),

Size of training samples (in pixels). Must have exactly the same values as used during training samples creation (utility *createsamples*)

Performance Utility: Performance of the classifier is evaluated using the *performance* utility. It takes a collection of marked up images, applies the classifier and outputs the performance, i.e. number of found objects, number of missed objects, number of false alarms and other information. Its command line arguments are as follows:

- **data**: directory name in which the trained classifier is stored

- **info**: (collection file name): file with test samples description

- **maxSizeDiff** (max size difference) and -**maxPosDiff** (max position difference) determine the criterion of reference and detected rectangles coincidence. Default values are 1.5 and 0.3 respectively.

- **sf** (scale factor): detection parameter. Default value is 1.2.

- **w** (sample width) and - **h** (sample height): Size of training samples (in pixels). Must have exactly the same values as used during training (utility *haartraining*)

Appendix C

Sample Results for the Training-Based ATR Algorithm [1]

Scenario 1 is used for the detection of the objects of interest (tank, truck and tractor) from azimuth angle 0^0 to 30^0 at all possible distances and declinations. Results of application of training-based method to scenario 1 are shown in the figures Fig. C.1 to Fig. C.6 [1].

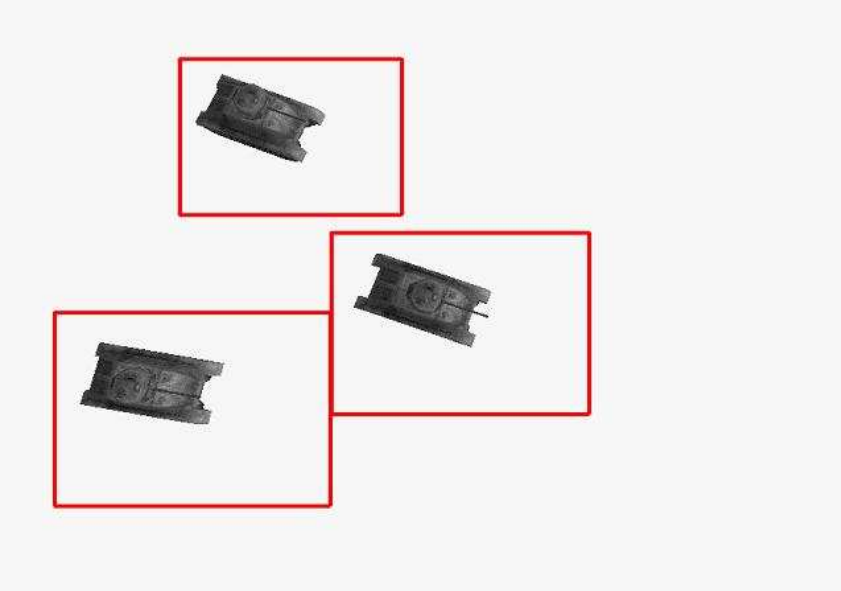

Figure C.1: Scenario 1: Image 1

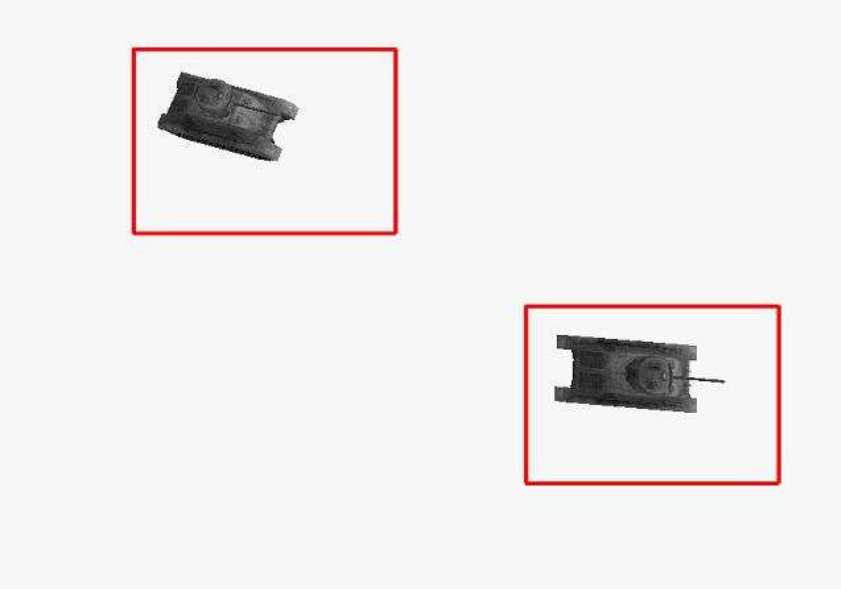

Figure C.2: Scenario 1: Image 2

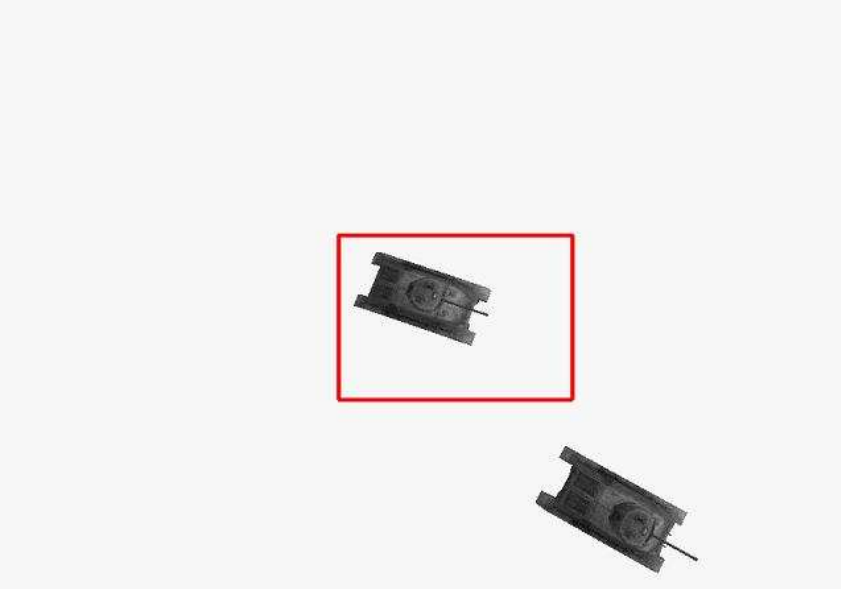

Figure C.3: Scenario 1: Image 3

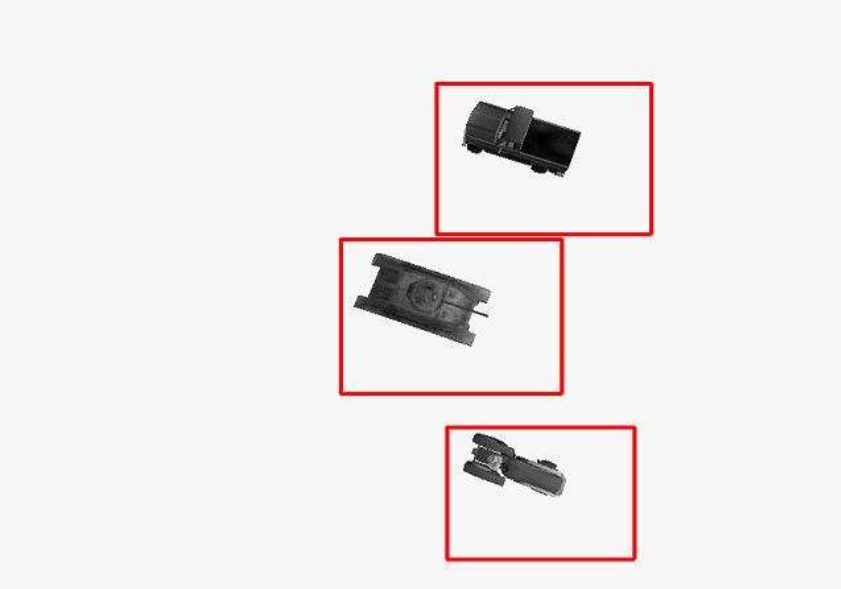

Figure C.4: Scenario 1: Image 4

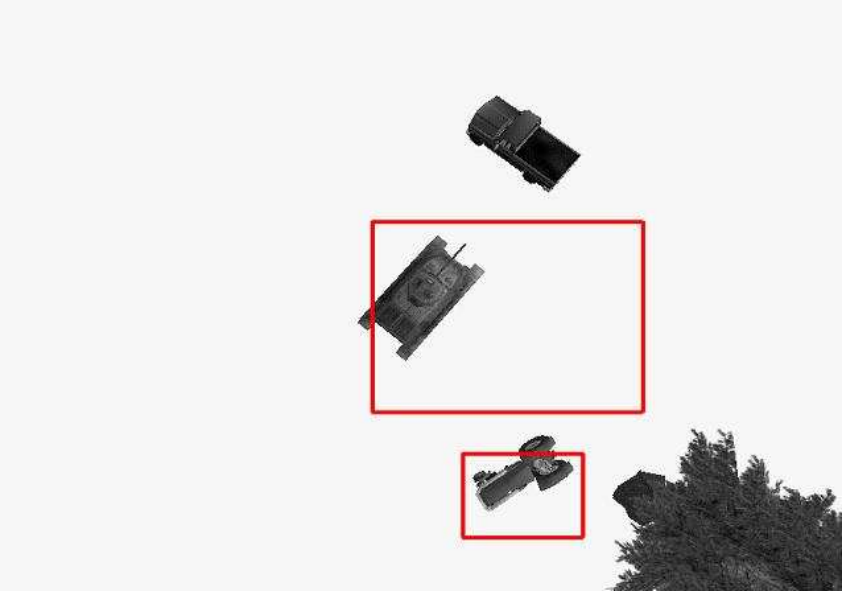

Figure C.5: Scenario 1: Image 5

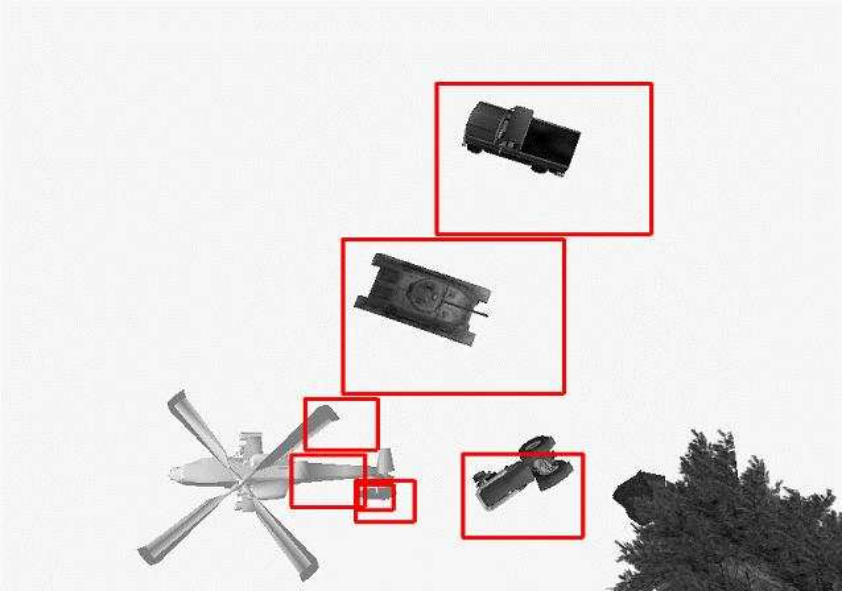

Figure C.6: Scenario 1: Image 6

Scenario 2 is used for the detection of the objects of interest (tank, truck and tractor) at different rotations from all view points. Results of application of training-based method to scenario 2 are shown in the figures Fig. C.7 to Fig. C.15 [1].

Figure C.7: Scenario 2: Image 1

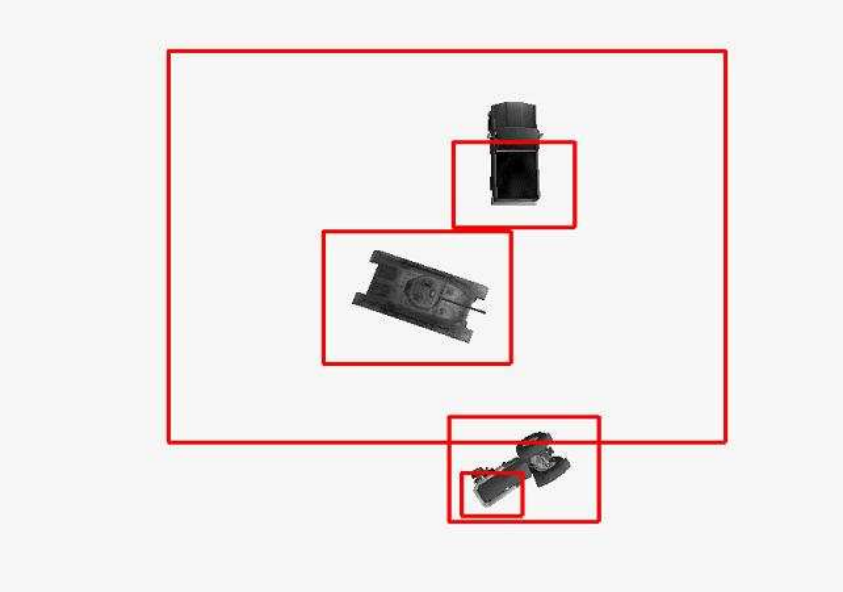

Figure C.8: Scenario 2: Image 2

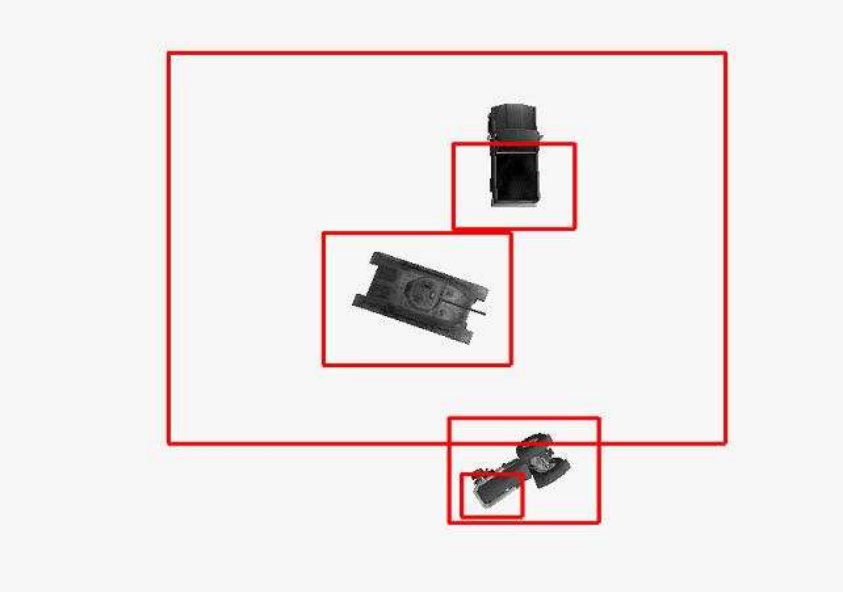

Figure C.9: Scenario 2: Image 3

Figure C.10: Scenario 2: Image 4

Figure C.11: Scenario 2: Image 5

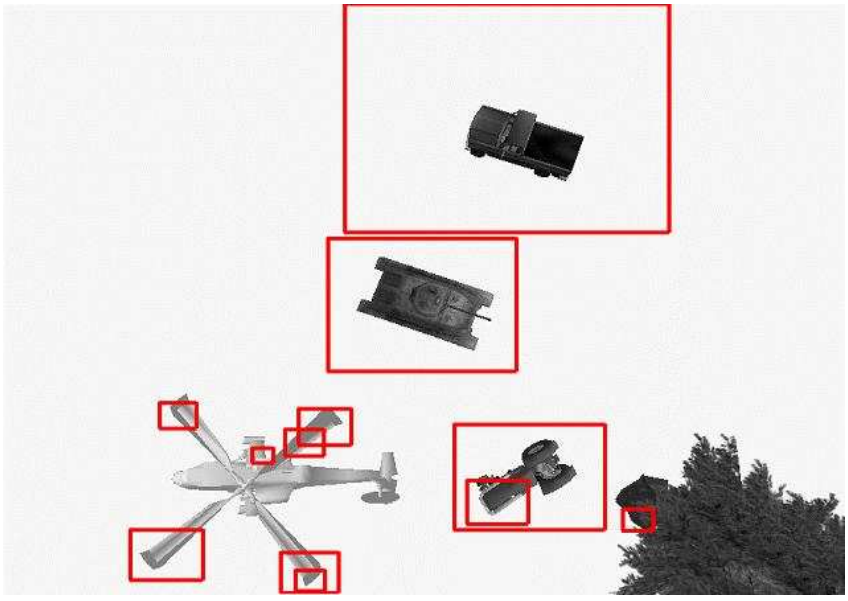

Figure C.12: Scenario 2: Image 6

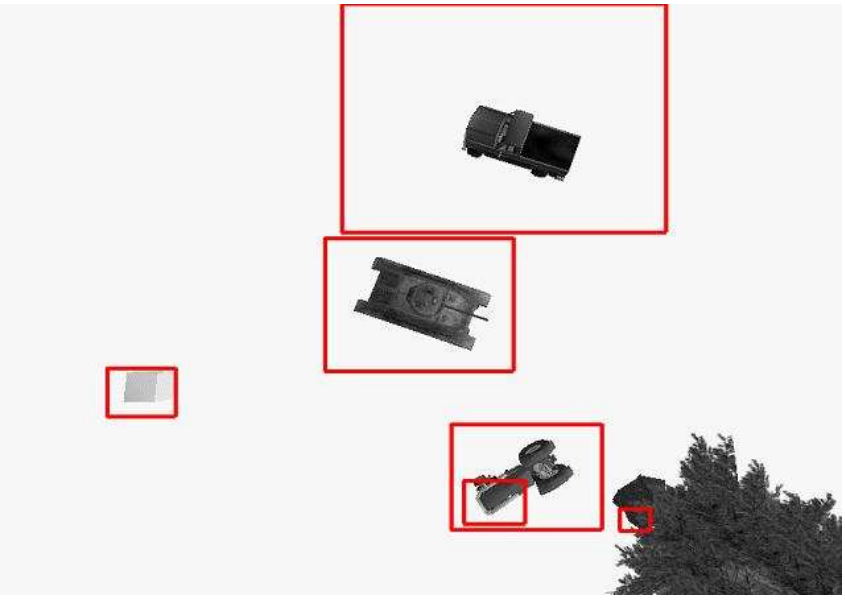

Figure C.13: Scenario 2: Image 7

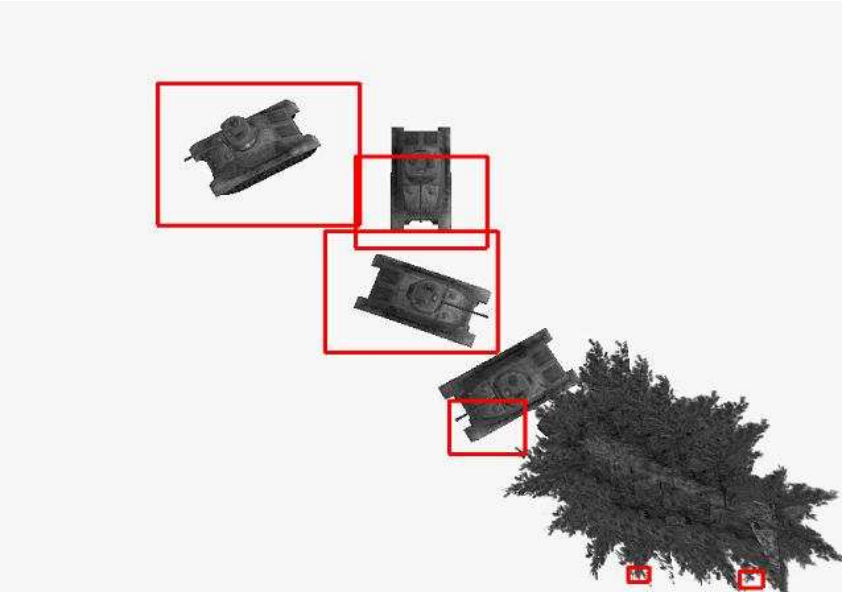

Figure C.14: Scenario 2: Image 8

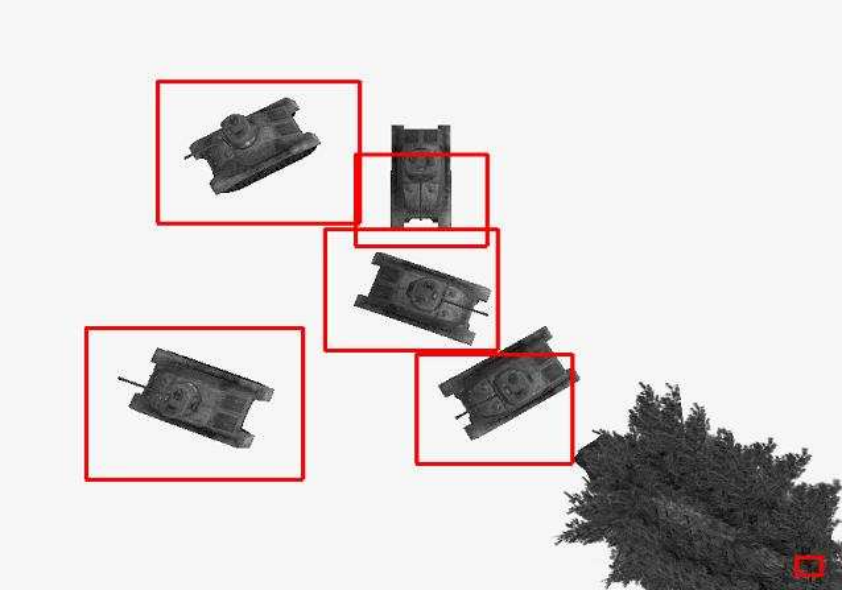

Figure C.15: Scenario 2: Image 9

Appendix D

Comparison Results for the Correlation-Based Algorithm and Training-Based ATR Algorithm [1]

This section illustrates the two algorithms by executing them on the same set of images. The images are labeled in pairs where the first image in the pair is obtained using the correlation-based method while the second image is obtained using the training-based method [1] .

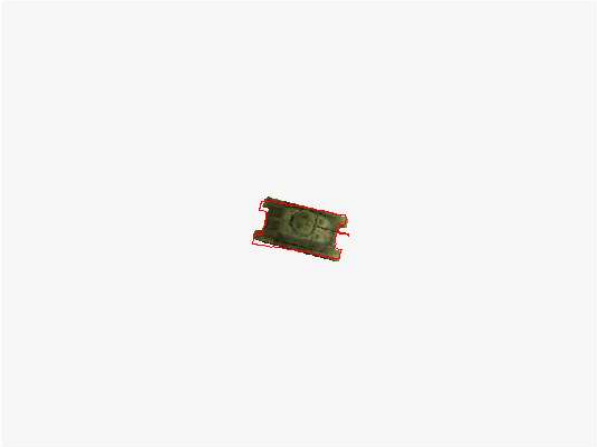

Figure D.1: Image 1a

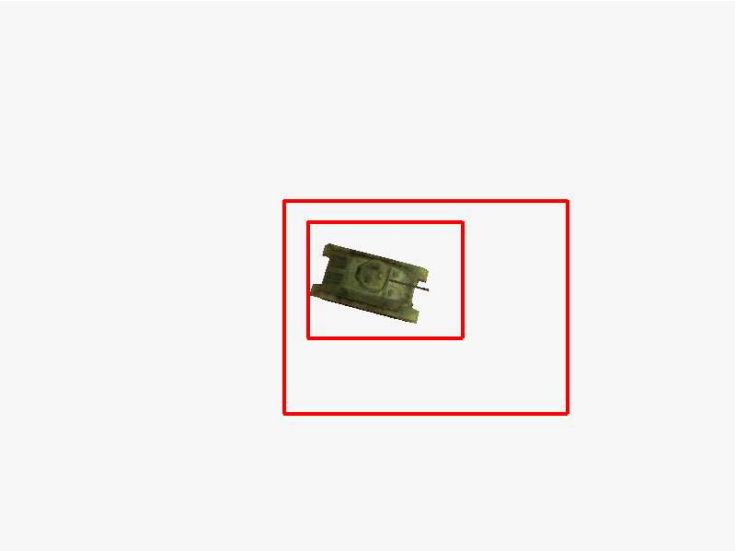

Figure D.2: Image 1b

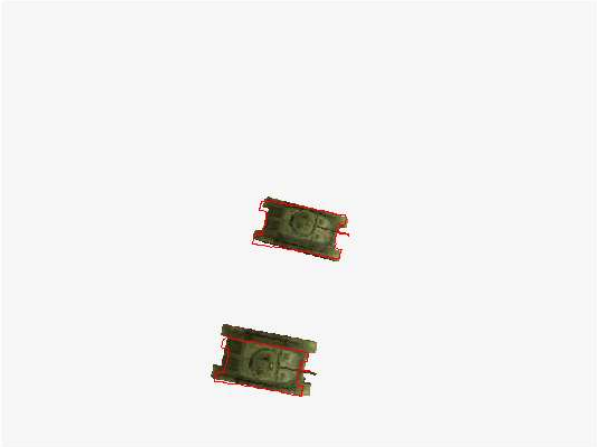

Figure D.3: Image 2a

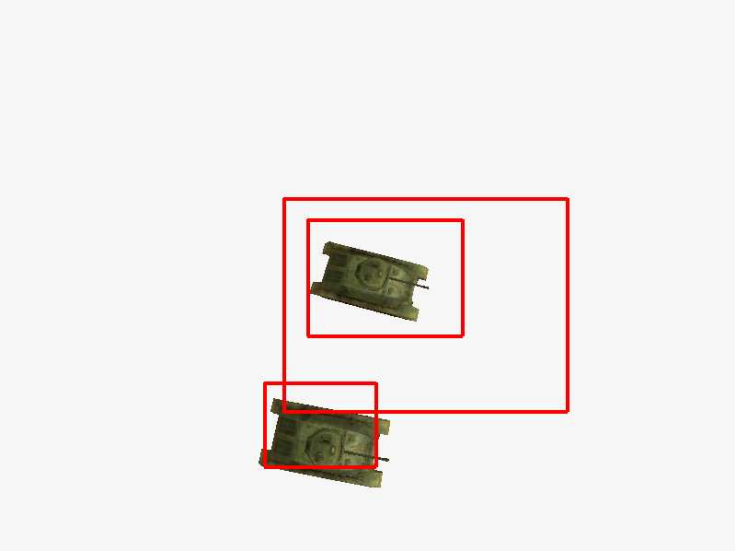

Figure D.4: Image 2b

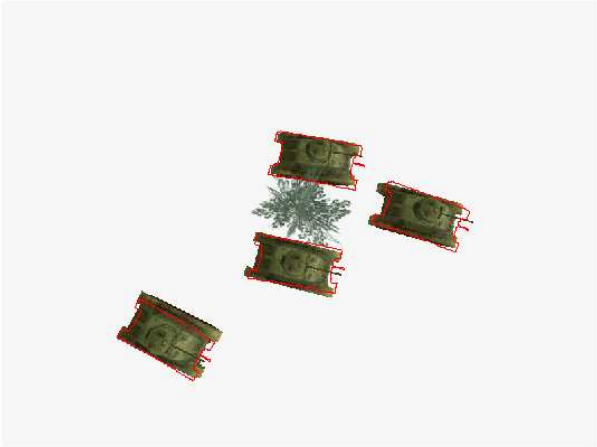

Figure D.5: Image 3a

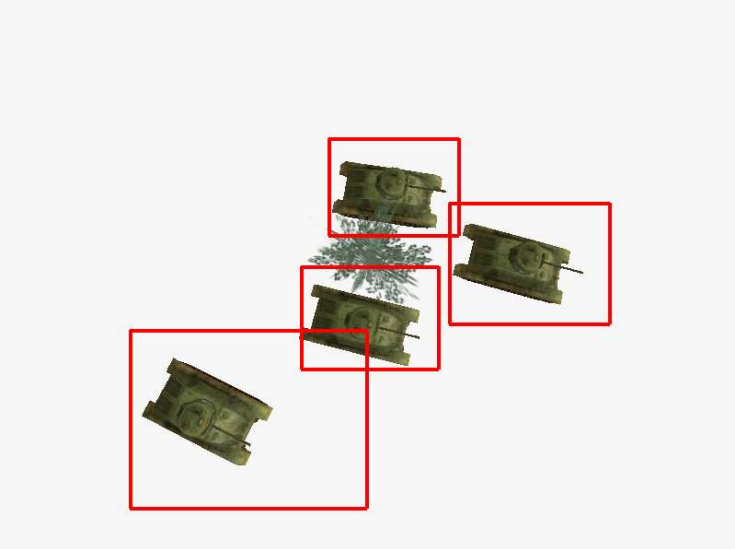

Figure D.6: Image 3b

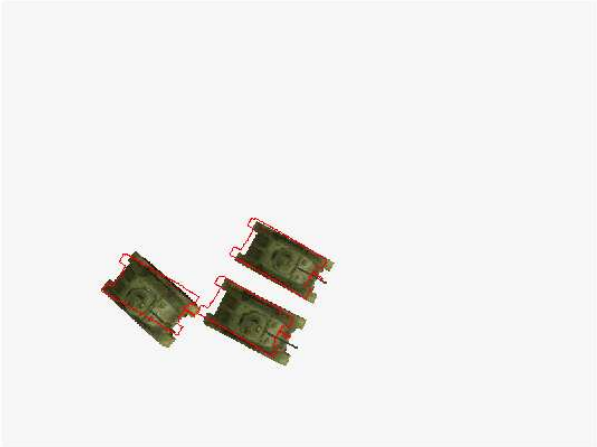

Figure D.7: Image 4a

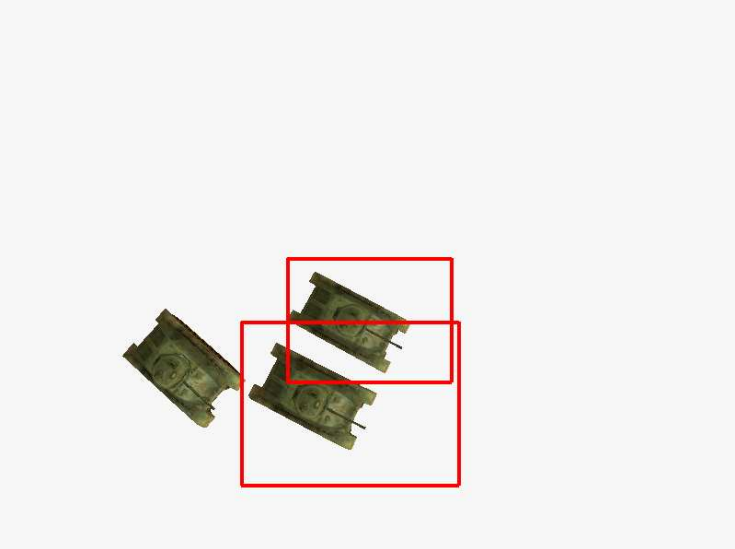

Figure D.8: Image 4b

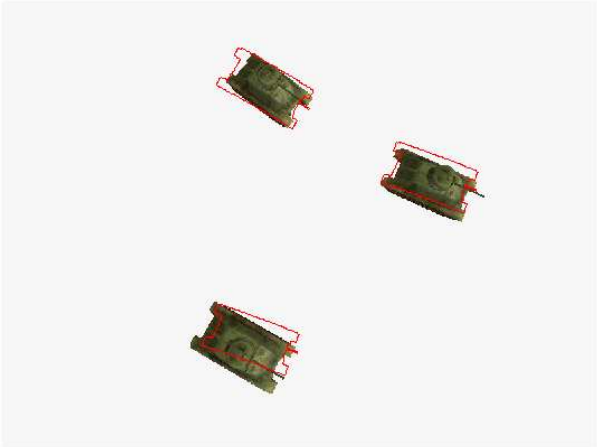

Figure D.9: Image 5a

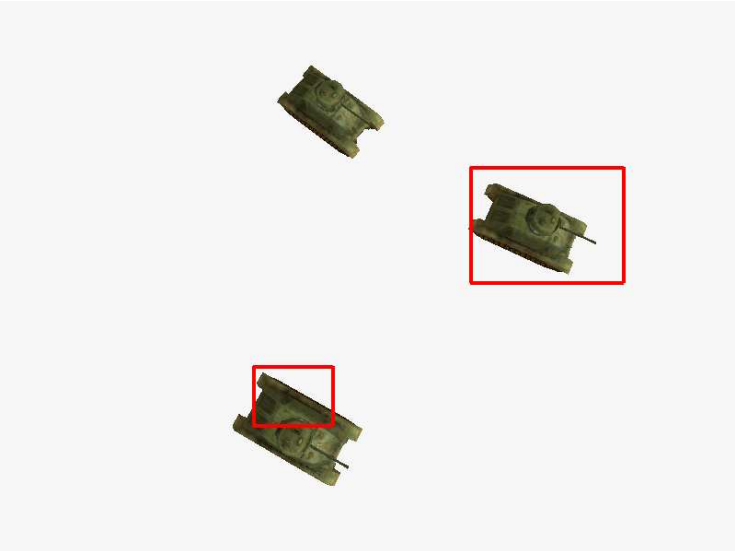

Figure D.10: Image 5b

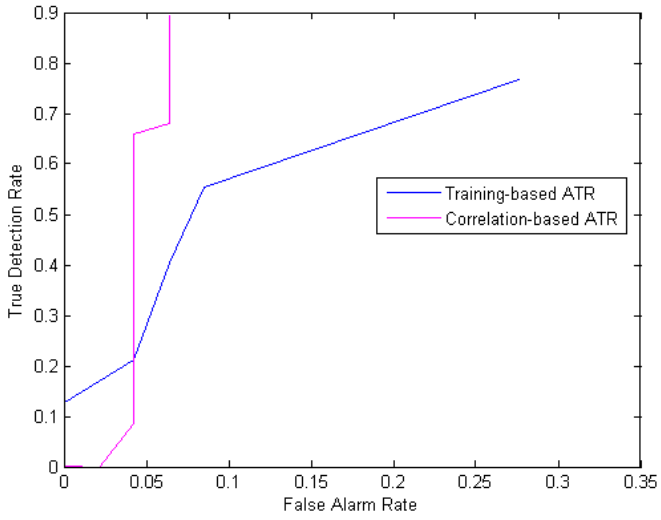

Figure D.11: ROC curves showing the comparison of the performance of the two ATR methods.