

Learning to Detect Planes at Low-Resolutions

Stavros Petridis, Christopher Geyer, and Sanjiv Singh
{sp104, cgeyer, ssingh}@ri.cmu.edu

August 29, 2006

Abstract

1 Introduction

The use of computer vision for detecting obstacles in the flight path of an aircraft is investigated in this paper. Such a system can be used to warn the pilots and help them avoid collisions and would also be useful aboard unmanned aerial vehicles (UAVs). Any aircraft detection technique should provide high detection rate, low false alarm rate and early detection. The main difficulties we face are the presence of image noise, the almost stationary nature of the target on collision course and the possible presence of heavily cluttered background. That system should also maintain the same level of performance under all conditions, and that is a drawback of all the existing methods since their performance heavily depends on the nature of the scene. In addition, the algorithm should be able to run in real time imposing severe constraints on execution time and the obstacles can vary from subpixel to a few pixels in size.

Obstacle detection in the flight path of an aircraft is a key component of a collision avoidance system. That also includes a collision risk estimation component together with a component that performs appropriate avoidance maneuvers in order to maintain minimum separation distances. Such an approach must demonstrate a level of performance which meets or exceeds that of a human pilot as stated in FAA order 7610.4 [5].

There are several different approaches which address the problem of obstacle detection in the framework of collision avoidance including morphological filtering [4], dynamic programming [1], and recursive max filter [10].

McCandless [9] proposed an optical flow method but this is suitable only for moving objects and therefore is not useful for a target on collision course.

Gandhi [6] proposed a two stage approach, an image processing stage followed by a tracking stage. The image processing stage isolates potential features and the tracking stage tracks these features to distinguish the real targets from background clutter. For detecting objects on a

collision course morphological filtering is used in the image processing stage and the rate of translation and expansion in the tracking stage. For detecting crossing objects a series of filters is applied to the image followed by a tracking algorithm based on Kalman filter.

Carnie [3] implemented a similar approach using morphological filtering followed by a dynamic programming algorithm to enhance detection.

The use of morphological filtering is popular on computer vision based collision avoidance systems [4], [3], [6]. However, this approach generates a significant number of false positives and requires tracking of the features over a large number of frames. Carnie reports that even after the dynamic programming algorithm a significant number of false positive is present.

Apart from the detection methods that are focused on aircraft detection there are also several methods who address the more general problem of object detection. One of the most popular methods was introduced by Viola and Jones [15],[16]. It has been mainly applied in the area of face detection but it can be easily applied in other situations as well. For example, Viola and Jones have used this method to detect pedestrians on the street [17]. Several variations of the original approach exist[2], [7], [13], [18] which aim to improve some weaknesses of the original detection framework.

A different approach is presented in [12] where multiple classifiers are used based on the statistics of some features. Another object detection method is proposed in [8] which is based on eigenspace density estimation.

In this paper we present an approach based on pattern recognition that initial experiments show that achieves a high detection rate (around 80%) and a reasonable number of false positives per frame (around 3). The system was tested on over 15000 test frames which is a much higher number than the typical number of test frames used by the majority of the existing plane detectors (less than 1000). It also seems to be insensitive to small scale changes. It is based on the Viola and Jones classifier [16] which is a very popular method for face detection. They report execution time of 15 frames per second which is a desired property for a system that wants to achieve real time op-

eration. The Viola and Jones classifier tries to detect faces from their structure whereas our classifier tries to separate the target from the background since targets which are far away do not have any structure. They appear as a row or a rectangle of few pixels. So we can think of the classifier as a trainable image processing filter.

Since we have a good classifier that operates on a frame-to-frame basis we use a very simple temporal filtering to reduce non-recurring false positives. Our implementation can be thought of as the first stage of the Gandhi's approach, where a classifier trained by AdaBoost is used instead of a morphological filter.

1.1 Challenges of Collision Detection

All the above approaches try to achieve a performance level equivalent to a human as this is a FAA requirement in order to use such a system on UAVs, as mentioned above. Almost all of the existing methods operate on a frame-to-frame basis and then apply a temporal filter or a tracking algorithm to use the temporal information contained in a video sequence. However, humans detect targets in a different manner. Humans fuse information from many different sources including spatial and temporal information but they also use contextual information and probably this is the most important source of information. Therefore a human is capable of identifying the detection on the bottom left of 1(left) as an island since it is located in the sea even if the island has the same appearance with an aircraft (Figures 1(right) and 2(a)). On the other hand it is very likely that most of the existing algorithms will not be able to distinguish the boat from the aircraft.

Another example is shown in Figure 2. Given that Figure 2(a) is an aircraft, it is likely that humans will identify the rest of the windows (2(b-e)) as aircrafts just like the detection algorithm, since their appearance is similar. However, when a human sees the frame of Figure 3 immediately rejects the same windows and identifies only Figure 2(a) (detection in the centre of Figure 3) as a real target since he uses contextual information, while the detection algorithm can not do the same. On the other hand, a detection algorithm can detect a target earlier than a human, as it is capable of detecting very small sized features, something which is hard for humans. That is also reported on the results of Carnie [3], where their detection algorithm detects the plane at distances which are 35-40% greater than those of the human observer.

From the above examples it is obvious that there is an upper limit in the performance of the detection algorithms. Even if we train a very good classifier it will be impossible to differentiate between a real target and a false



Figure 1: Top: An image of the coast in which a boat on the water could be confused with a plane because of its apparent similarity, zoomed in on the bottom, to images of planes (see Figure 2(a)).

positive that looks like a target. The use of temporal information is useful in several situations, e.g non recurring false positives, but still there are some situations, e.g. when there is a consistent detection of false positive like Figure 1, that it is not possible to distinguish the real targets from false positives. Therefore the use of additional information, e.g. contextual information, is necessary in order to have a system that achieves a performance level equivalent to a human, as stated in the FAA order.

2 Background

2.1 Target Resolution

Initially we assume an object with width w which occupies an angle α (degrees per object) at a distance d (Figure 4, [11]). The ratio of the horizontal resolution, Res , to the horizontal field of view (FOV), defines the number of pixels occupied by the object per degree. Therefore by multiplying the angle α with the above ratio we get the number of pixels occupied horizontally by the object.

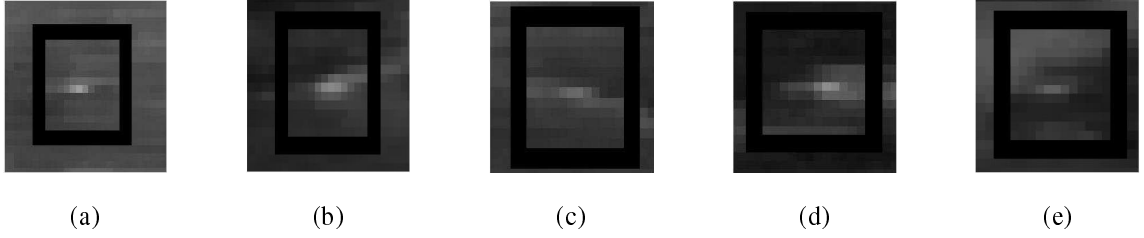


Figure 2: (a) True positive from Figure 3 (b)–(e) False positives from Figure 3 .

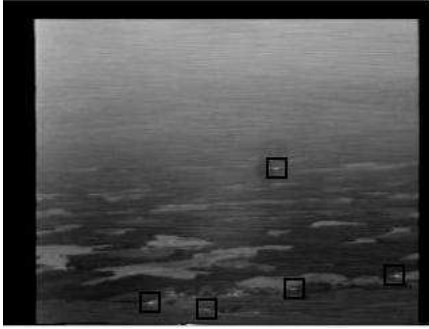


Figure 3: A frame from a collision course sequence with one true positive (top) and four false positives.

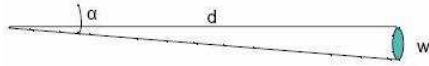


Figure 4: An object with width w occupies an angle α at a distance d .

$$\alpha \frac{Res}{FOV} = \arctan(w/d) \frac{Res}{FOV} = noPixels/Object \quad (1)$$

So if we consider that the smallest target we want to detect has a width of 11 meters, e.g. Cessna, and take into the account that the horizontal resolution and the FOV of the camera used in our data is 320 pixels and 9.75 degrees respectively then we can calculate the number of horizontal pixels occupied per object as a function of distance. The results are shown in Figure 5. We also considered the case that the target's width is 18m which corresponds to

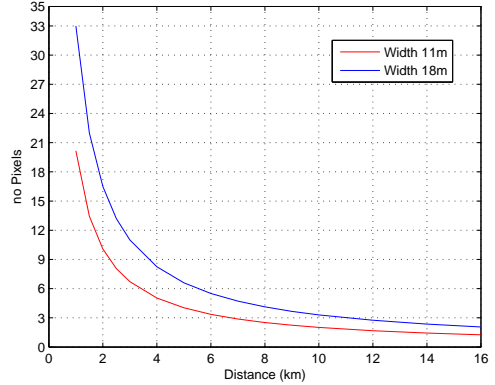


Figure 5: The number of pixels occupied by the target (in width) as a function of the distance from the host aircraft.

the Beech King Air 200 that appears in our data.

Figure 6 shows the number of pixels occupied by the target horizontally as a function of the distance between the two aircraft. If we consider a relative speed of 650 km/h we can find the time to collision simply by dividing the distance from Figure 5 with the relative speed between the two aircraft. However, in reality there are also other factors which affect the number of pixels occupied such as lighting.

2.2 Geometry of Collision Course

Initially we consider a point (x_0, y_0, z_0) moving with constant velocity (v_x, v_y, v_z) in the three dimensional space. Axes X and Y lie on a plane which is parallel to the image plane (X is horizontal and Y is vertical) whereas Z axis is perpendicular to the image plane. The perspective projection of the point onto the image plane is given by

$$u = \frac{tv_x + x_0}{tv_z + z_0}, v = \frac{tv_y + y_0}{tv_z + z_0} \quad (2)$$

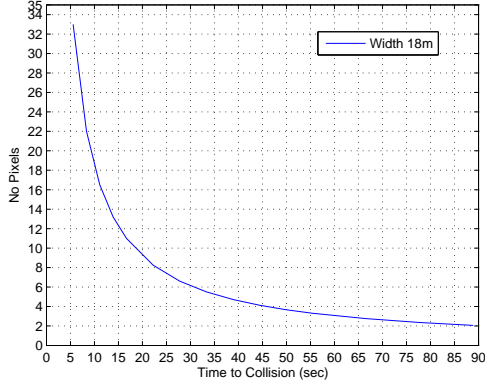


Figure 6: The number of pixels occupied by the target (in width) as a function of the time to collision with the host aircraft.

If we assume that $y_0 = 0$ and $v_y = 0$ then from eq.2 we see that $v = 0$. If the target is on collision course then (assuming the host aircraft velocity has been subtracted)

$$v_x \approx sx_0, v_z \approx sz_0 \quad (3)$$

Therefore from eq. 2 and 3

$$u \approx \frac{tsx_0 + x_0}{tsz_0 + z_0} \quad (4)$$

So far, we have assumed to take the projection of a point that corresponds to the center of the target. However, a target has size. So if we take into account the size of the object then locally, near the object we have

$$u \approx \frac{tsx_0 + x_0 + dx}{tsz_0 + z_0} = \frac{tsx_0 + x_0}{tsz_0 + z_0} + dx \frac{1}{tsz_0 + z_0} \quad (5)$$

In eq. 5 the term $\frac{1}{tsz_0 + z_0}$ is the scale multiplier. If we observe the target twice, at times t_0 and t_1 respectively, and we also observe an expansion of the target by α , then we have

$$\frac{\frac{1}{t_1sz_0 + z_0}}{\frac{1}{t_0sz_0 + z_0}} = \alpha \quad (6)$$

The approximate range is $tsz_0 + z_0$ so we can compute the time to collision by setting the range equal to zero.

$$t_csz_0 + z_0 = 0 \Rightarrow t_c = \frac{-1}{s} \quad (7)$$

If we solve for s from eq. 6 then we have

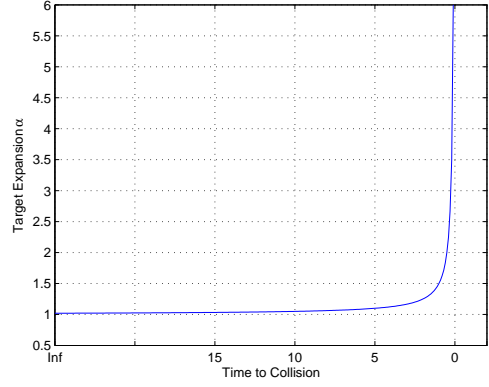


Figure 7: Target expansion α as a function of time to collision.

$$s = \frac{1 - \alpha}{t_1\alpha - t_0} \quad (8)$$

From eq. 7 and 8 we can compute the time to collision as a function of α

$$t_c = \frac{t_1\alpha - t_0}{\alpha - 1} \quad (9)$$

So if we can accurately estimate the expansion of a target between two observations then we can also estimate the time to collision. Time to collision is expected to decrease linearly, assuming that the relative velocity between the two aircrafts is constant. So assuming a linear decrease for t_c we can estimate the behavior of α from eq. 9 ($t_0 = t - \Delta t, t_1 = t$) as a function of time as shown in Figure 7. From the profile of α we see that it is very hard to have a reliable estimate for α since it takes values very close to one for a very long period and only a few seconds before the collision occurs it increases dramatically.

In addition, a plane on collision course is expected to be almost stationary in the image plane. Since the target aircraft flies towards the host aircraft then the in-plane motion is expected to be small whereas the motion in the axis which is perpendicular to image plane is large. Unfortunately in a 2D-image plane we can not measure the perpendicular motion so the plane looks almost stationary. Therefore the use of optical flow or a moving object detector is not suitable for that class of problems.

3 System Overview

The system we propose here is based on the framework introduced by Viola and Jones [15]. It consists of six stages

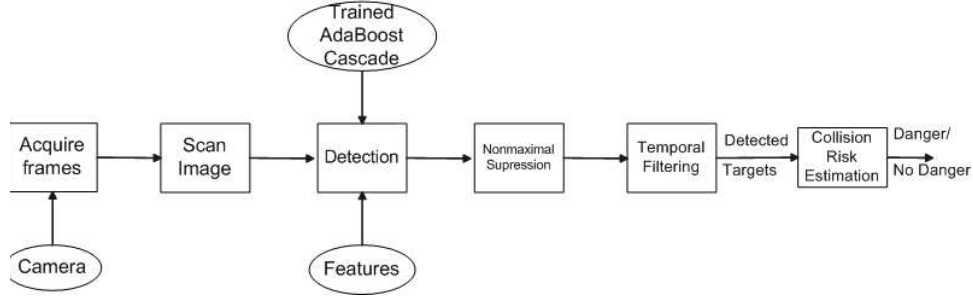


Figure 8: The proposed architecture for collision detection. We address in this paper detection, and spatial and temporal filtering.

as shown in figure 8.

In the first stage an image frame is acquired by the camera and then in stage two a sliding window is used to scan the frame. The next 3 stages which are described in more detail in the next section are the main parts of the system and they are consisted of the detector followed by a non-maximal suppression stage which is followed by a temporal filtering stage. Then the detected targets are passed to the final stage which determines which targets (that also includes the false positive detections) pose a threat to the aircraft. The main focus of this paper was the development of stages 3 to 5.

The system shown in Figure 8 should work in real time. However, the detector should be trained off - line and when the training process ends then it can be used on line as well.

3.1 Cascade — Training

A cascade is a structure which consists of a series of classifiers as shown in Figure 9. When a pattern is fed into the cascade then it is labeled as positive if it successfully passes all stages whereas there are multiple exits for the patterns who fail at some point and those are labeled as negative.

In the third stage, a cascade classifier is applied to all subwindows of each frame. The key point is that within any single frame the vast majority of subwindows are negative so the cascade attempts to reject as many negatives as possible at the earliest stage possible. So the first stages consist of simple and fast classifiers which reject most of the negative subwindows while keeping almost all of the positive subwindows. Subsequent stages eliminate additional negatives but they are more complex and require additional computation time. Stages in the cascade are con-

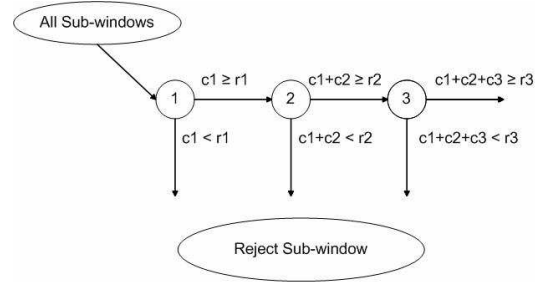


Figure 9: The cascade structure we used in this paper.

structed by training classifiers using AdaBoost as in [15]. However we do not use the original cascade where each classifier outputs "Yes" or "No" but we follow the boosting chain approach [18] where each classifier outputs a real value which is added to the sum of the outputs of all the previous classifiers. If the new sum is greater than the stage's threshold then the pattern is fed into the next classifier, otherwise it is discarded. Using this approach, the history of a pattern is taken into account and that is beneficial as explained below. So the cascade's function can be summarized in the following three steps.

1. Given an input pattern initialize $s = 0$.
2. For all stages $s = s + \text{stage output } (c_i)$, if $s < \text{stage threshold } (r_i)$ then exit with negative response
3. Exit with positive response.

The way this cascade works can be clearly seen in Figures 9 and 10. The output value of the i -th stage is c_i and

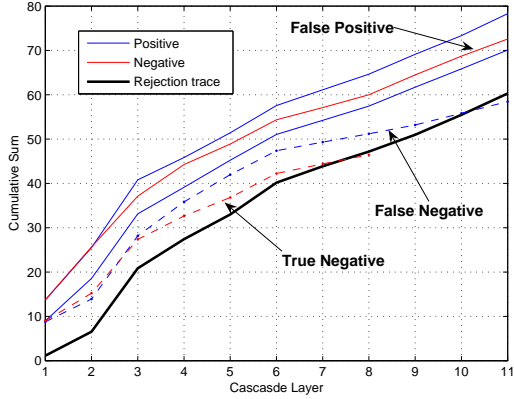


Figure 10: An example of how the cascade works

the corresponding rejection threshold is r_i . A real example of how the cascade works is shown in Figure 10.

The X-axis represents the stages of the cascade and the Y-axis represents the cumulative sum of the stage outputs. Similarly to [2] we can define a rejection trace (black trace) which represents the thresholds of each stage. So as long as the partial sums are greater than the rejection threshold then the pattern is kept. In Figure 10 blue lines represent airplanes (targets) and red lines represent negative patterns. So the red lines that are above the rejection threshold until the final stage are false positives. The red dashed-lines represent rejected negative subwindows whereas the blue dashed-lines represent false negatives. Using this approach the performance of a pattern in the prior stages is taken into account and also a pattern may fail one or more stages but as long as it stays above the rejection threshold it can be correctly classified. On the other hand, in the original hard cascade when a pattern fails one stage then no matter how well it performed in the previous stages is discarded.

Similarly to [15] each stage is trained using a single positive training set and a negative set which consists of the false positives created by scanning the partial cascade on a validation set. The training of each stage by AdaBoost is described in table 1, and is exactly the same algorithm used in [15].

Initially all the patterns are given a weight, which is the same among the positive and negative patterns respectively. In each iteration the best single feature that separates the negative from the positive patterns in terms of the weighted error is selected and the weights are updated according to the rule in step 4 of the loop. Since the weights are updated in each iteration it is rare that the same feature

1. Given example images $(x_1, y_1), \dots, (x_n, y_n)$ where $y_i = 0, 1$ for negative and positive samples respectively.
2. Initialize weights $w_{1,i} = 1/2m, 1/2l$ for $y_i = 0, 1$ respectively, where m and l are the number of negative and positive samples respectively.
3. For $t = 1, \dots, T$,

- (a) Normalize the weights:

$$\hat{w}_{t,i} = w_{t,i} / \sum_{j=1}^n w_{t,j}.$$

- (b) For each feature, f , train a classifier h_f (eq 11), which is restricted to using a single feature. The classifier is trained by selecting the threshold θ_f that best separates the positive and negative samples.
- (c) Choose the classifier with the lowest weighted error
- (d) Update the weights to

$$w_{t+1,i} = w_{t,i} \beta_t^{1-e_i},$$

where $\beta_t = \varepsilon / (1 - \varepsilon_t)$ and $e_i = 0$ if sample x_i is classified correctly, or $e_i = 1$ if it is incorrectly classified.

4. The final strong classifier is

$$h(x) = \begin{cases} 1 & \sum_{t=1}^T a_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^T a_t \\ 0 & \text{otherwise} \end{cases}$$

where $\alpha_t = -\log \beta_t$.

Table 1: AdaBoost

will be selected in two consecutive iterations. In addition, this process can also be thought of as a feature selection mechanism since the best feature is selected in each iteration.

The number of features in each stage is increased and the thresholds determined by AdaBoost are adjusted until the goals for the false positive rate and detection rate per stage are met as described in [15]. So new stages are added to the cascade until the final goals for the false positive and detection rate are met.

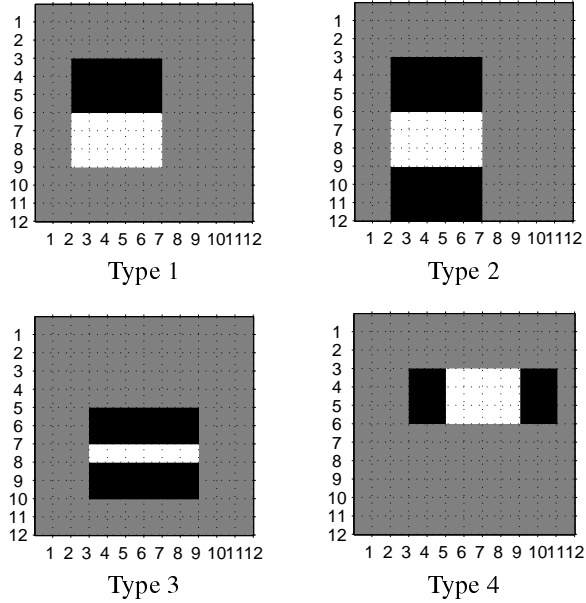


Figure 11: The types of features used in this paper.

3.2 Features

The 4 types of features from which the training system will extract the best features in this system are shown in Figure 11. Apart from the three commonly used filters, e.g. type 1 and type 2, two more filters were introduced to take into account the nature of the problem. A target in a large distance is expected to have greater width than height as shown in Figure 2(a). Unlike face detection, we can not detect the plane from its structure but we want to distinguish the target from the background then the inclusion of filters 3 and 4 helps towards this goal. The features are used in all locations and scales that fit in the detector.

The output of each feature is the sum of the pixels which lie within the white rectangle subtracted from the sum of the pixels which lie in the black rectangle. If the number of the white and black pixels is not the same then before the subtraction takes place the sum of the smaller number of pixels is multiplied by a constant, α , such that the number of white and black pixels is the same. For example in type 3 features the number of black pixels is four times larger than the number of white pixels, therefore the sum of the white pixels is always multiplied by 4 before subtracting it from the sum of the black pixels.

$$\sum_{p \text{ in black region}} I(p) - \alpha \sum_{p \text{ in white region}} I(p) \quad (10)$$

A classifier, h which consists of the feature value, f ,

and a threshold, θ is defined as follows:

$$h(x) = \begin{cases} 1 & \text{if } f < \theta \\ 0 & \text{otherwise} \end{cases} \quad (11)$$

During training with AdaBoost, the best threshold, θ , for each feature is found and then the feature with the smallest weighted error is selected (steps 2 and 3, Table 1).

The five best features selected by AdaBoost are shown in Figure 12. Notice that all of them measure the difference in intensity between the target (mainly rows 6 and 7) and the background. The first and the fifth best features are of type 3 and the second best feature is of type 4. This fact also justifies the introduction of the new features.

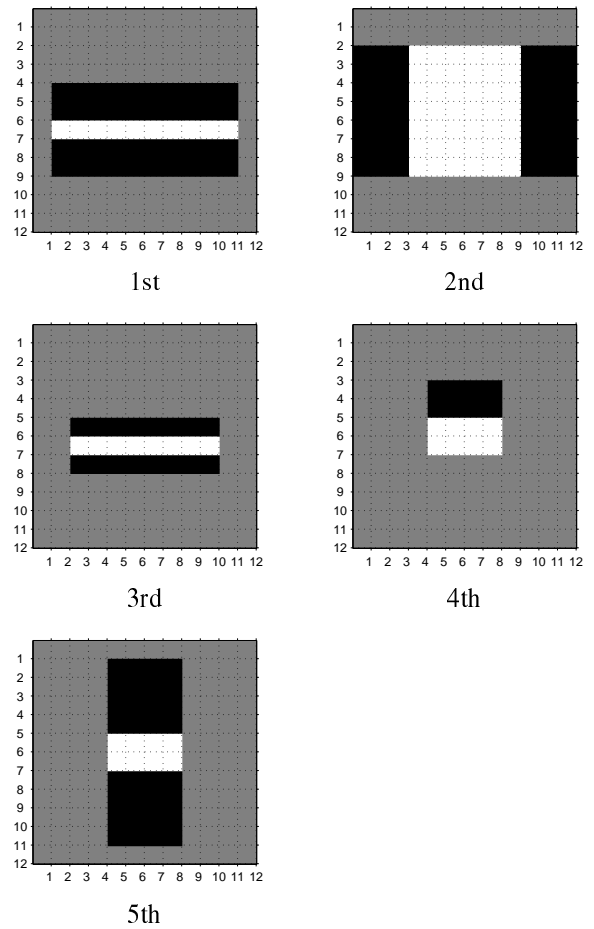


Figure 12: The five best features selected by AdaBoost.

The output of the first feature for the training set used is shown in Figure 13 together with the threshold selected by AdaBoost, 448.5. The separation between positives and negative samples is clear. There are only a few false

positives whereas there are no false negatives. Figure 14 shows the distribution of the positive and negative response of the first feature and it is clear that the overlap is small.

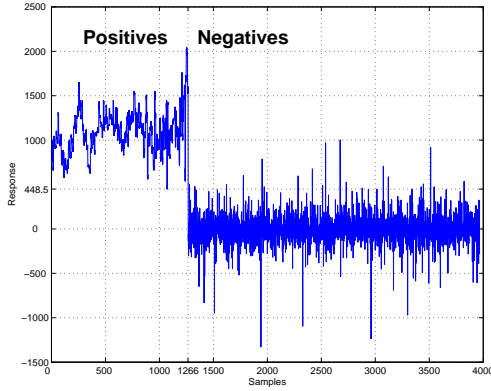


Figure 13: The output of the first feature for the training set

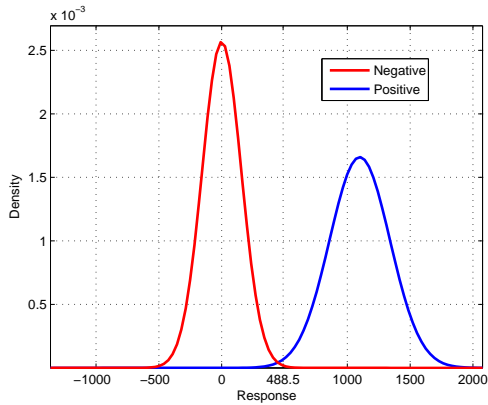


Figure 14: The distribution of the positive and negative output (see Figure 13)

Figure 15 shows three images that do not contain targets. The distributions of the responses to the best feature of all the 12×12 sub-windows that exist in each image are shown in Figure 16. The distribution of the responses of the positive samples of the training set is also shown in figure 16 for better comparison. We see that the widths of the distributions are slightly different and the overlap with the positive distribution is small. However, that overlap is enough to generate a few false positives per frame. All the negative distributions are centered to zero which means

that the value of the best feature is close to zero for a significant percentage of the images' sub-windows. From eq. 10 we see that the feature value is close to zero when the sum of the white pixels is almost equal with the sum of black pixels. Practically, that means that there is no significant difference in those two areas. On the other hand, the positive distribution is centered at 1100 and that reveals a significant difference between the black and white area as a result of the existence of a target whose pixel distribution is different from the background.

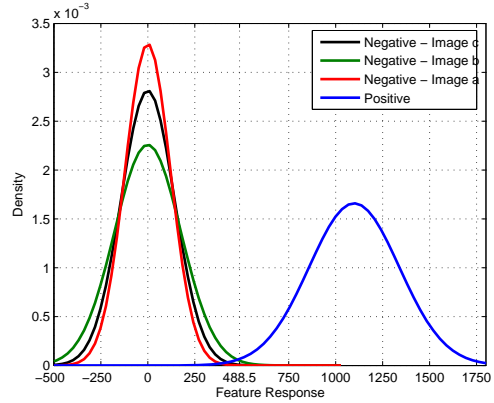


Figure 16: The (negative) distributions of the responses to the best feature of all the sub-windows from the three images from Figure 15. The (positive) distribution of the response of the training set is also shown (same as Figure 15).

3.3 Output of the Classifier

After scanning a frame with the classifier we get a matrix in which each cell corresponds to a subwindow of the original image. Therefore the new image is $n - 1$ pixels shorter in each direction (horizontal, vertical) where n is the height and width of the detector (we consider a square detector). Each pixel contains the output value of the classifier applied to a subwindow so it is expected that most pixels will be zero, as most of the subwindows should be rejected by the classifier, and few pixels will have a positive value. For example, if we consider a 3×3 detector (Figure 17) then the pixel (1,1) of the right image is the output of the classifier when applied to the 3×3 subwindow on the top left (rows 1 to 3, columns 1 to 3). Similarly, the selected pixel (2,2) on the right image is the classifier's output for the selected window on the left image

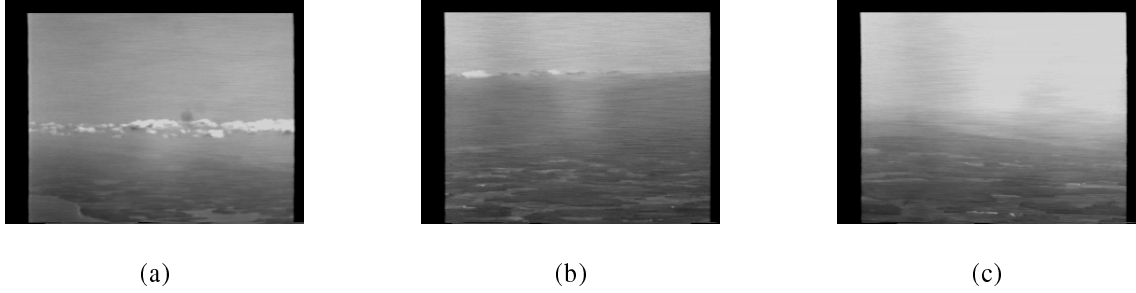


Figure 15: Three images which do not contain planes.

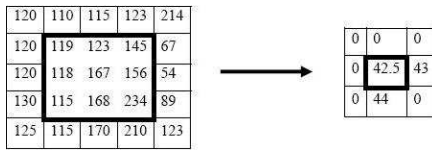


Figure 17: Left: Image, Right: Output of the classifier

3.4 Non-Maximal Suppression

In the fourth stage, non-maximal suppression is performed to the output of the classifier to get a single feature for the entire detection, either it is a true target or a false positive. The cascade classifier is insensitive to small changes in translation so multiple detections usually occur near each positively detected pixel. A 13×13 neighborhood is defined for each pixel and only the maximum value is kept within this window whereas the rest positive pixels are suppressed. Such a neighborhood is large compared to the 3×3 neighborhood used in the crossing object detection in [6] but our decision is based on the fact that it is very unlikely that two planes will be so close to each other. But even if this is the case then we will obtain at least one detection in the neighborhood.

3.5 Temporal Filtering

In the fifth stage, a simple temporal filtering is used. For each detection in frame k we check a 5×5 neighborhood around the detected pixel in frame $k-1$. If a detection exists in that neighborhood in the previous frame then the detection in frame k is displayed otherwise not. For example in Figure 18, there is a detection in pixel (2,3). We check the 5×5 neighborhood in the previous frame and since there are no detections then the current detection is not displayed in the frame. We chose a 5×5 neighborhood since a target on collision course is expected to be almost

stationary. Carnie [3] for the same problem chose a 3×3 neighborhood. However, there are some cases where this condition is violated, e.g. sequence 3 in collision results section. Of course, this approach is not useful for detecting crossing objects as it makes the assumption of a stationary target. As a result the performance is degraded as shown in the crossing results section.

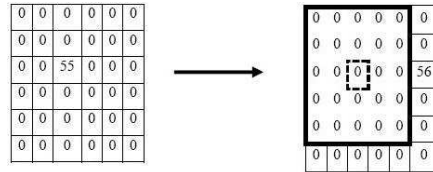


Figure 18: Left: Output of the classifier at frame k . Right: Neighborhood checked for detections at frame $k-1$

We should note that even if the detection is not displayed in the frame this does not mean that it is discarded. If it was discarded then it would be impossible for new detections to appear in a frame as the previous frame neighborhood for a new detection is always empty. This simple filtering intends to reduce the number of false alarms as target detections tend to be detected in a consistent way whereas it may be possible to have a false alarm only in a single frame due to image noise. This implies that in order to have a new detection in a specific position, this detection must be present in the same area at least for two consecutive frames. Of course, also the detection rate is decreased by a small factor as when the target is small or when the contrast is low then the aircraft is not detected consistently in all frames as assumed. There are also some cases, Figure 24, where this fact can degrade the overall performance even if the pre-filtering performance is good.

In addition a second approach to the temporal filtering was used. In this approach we consider the n previous frames for every detection which is displayed if there are



Figure 19: A heavily cluttered scene from our data

at least $m_j = n$ detections in those n frames. This filter is implemented in the same way as the one described above, i.e. for each detection in frame k we search in its neighborhood in the previous frame and then the maximum detection becomes the center position for the neighborhood in frame $k-2$. In case, there are no detections in one neighborhood in frame $k-1$ then the same center position is considered for frame $k-2$ but the neighborhood size is increased by one pixel in each directions (vertical, horizontal).

4 Data Collection and Classifier Training

We had access to about 4 hours of video that was captured during test flights conducted at NASA Langley Research Center in 1997 [9]. A camera was mounted below the nose of a Boeing 737 and used to record video of a Beech King Air 200 flying in simulated collision courses with the Boeing 737. The horizontal and vertical fields of view of the camera were 13 and 9.75, respectively. The video tapes were digitized using a frame grabber resulting in a video sequence with a frame rate of 30 frames/sec, resulting in image dimensions of 320×240 pixels and at 256 grayscales. We did not have access either to the navigational data or to the distance between the target and the host aircraft for each frame. In total, there are 15 sequences with aircrafts and 3 different classes of maneuvers. There are 4 sequences where the target aircraft is on collision course (110 sec), 5 where the target aircraft flies perpendicular to the host aircraft (22 sec) and 6 where the target aircraft flies directly away from the host aircraft (480 sec). A frame from our data is shown in Figure 19..

4.1 Training on the Dataset

We manually extracted windows in all 15 sequences that contain imagery of aircraft (10min). In 8 sequences there is low clutter in the background whereas in the remaining 7, the background is heavily cluttered, including clouds, ground and sea regions. We also digitized 11 sequences of medium and high clutter with no aircrafts to better estimate the number of false positives.

In order to train the cascade we used 1266 positives patterns from 2 of the crossing object maneuvers. For each positive example, we include about the vertical axis and its left and right translation by up to three pixels were included in the training set in order to generate a broad, generalizable training set. All the patterns were contained in a 12×12 window. Examples of positive and negative training images are shown in Figures 20 and 21 respectively.

Figures 22(b) and 22(c) show two aircrafts which fly perpendicular to the host aircraft and Figure 22(a) shows an aircraft on collision course. Since the appearance of the aircraft in these two cases is quite similar we used a single classifier trained on examples from a crossing object maneuver, and use it as a classifier for collision course trajectories as well.

The first two classifiers of the cascade were trained using the positive training set and 2 different negative sets containing 2700 patterns each. The negative patterns used to train the subsequent classifiers were obtained by scanning the partial cascade across 200 frames taken by half of the sequences with no planes and collecting false positives. For each layer, 3000 false positives were used for training. There are approximately 14 million negative subwindows in the 200 frames.

In order to adjust the threshold of each layer a separate validation set was used as in [15]. The validation set consisted of 978 positive patterns taken from the third crossing object maneuver and 6200 negative patterns.

The final detector is a 13 layer cascade of classifiers which includes 335 features. The number of features used by each classifier is : 8, 12, 16, 22, 32, 33, 34, 36, 44, 48, 50. All the types of features (as shown in Figure 11) were used to train the classifiers

The MATLAB implementation of the cascade takes about 3 seconds to scan a 320×240 frame, with the potential to be significantly faster when implemented in a low-level language. Viola and Jones report an execution time of 15 frames/sec. Furthermore, scanning a frame is a highly parallelizable task so it possible to achieve a real time operation of scanning 30 frames per second.

In addition a second classifier was trained using the same positive but different negative sets from the first

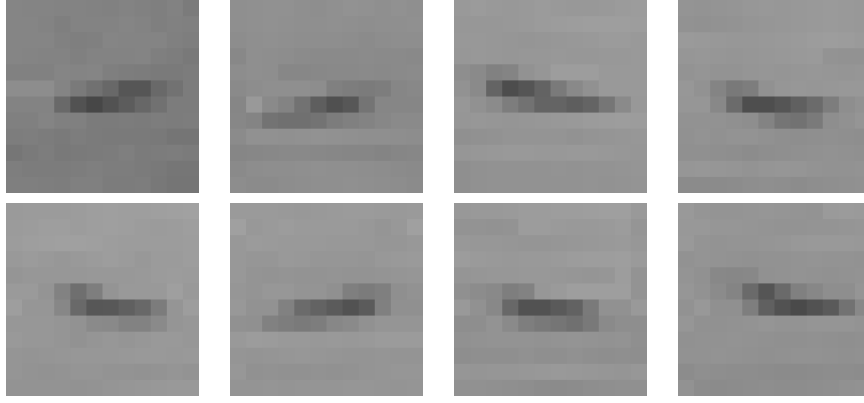


Figure 20: Examples of training (positive) images

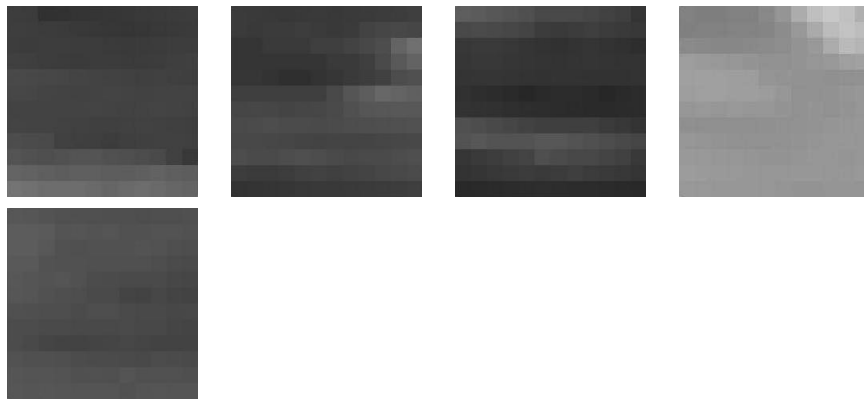


Figure 21: Examples of training (negative) images

classifier. For that classifier only the first three types of features were used but no features were excluded as in the previous case, making a total of approximately 20000 features. The final detector is a 7 layer cascade of classifiers which includes 139 features. The number of features used by each classifier is: 4, 6, 13, 21, 23, 32, 40.

5 Results

Table 2 shows the detection rate as the target approaches to the host aircraft. The third sequence from the collision course results section was used in this experiment. The background is smooth in this sequence so it is reasonable to consider the detection rates of table 2 as upper limits. In a heavily cluttered sequence or in a sequence with low contrast it is expected that the detection rate will be worse. The part of the sequence that was used for that experiment was divided into three sub-parts which roughly

correspond to three different target widths, 3 to 6 pixels, 6 to 10 pixels and 10 to 13 pixels.

Figure 23 shows the detection rate as a function of the time to collision. Again the same part of sequence 3 was used as in Table 2. In order to get a single point in that curve we average ten consecutive frames. We see that in the beginning the detection rate is low and as the target approaches the host aircraft (time to collision decreases) the detection rate increases. We should also note that the detection rate for about 6 – 7 seconds it is high but the detections are not consistent as shown from the fluctuation around 80%. After a point the target is detected consistently and consequently the detection rate becomes constant. The background of this sequence is smooth and that is why the detection rate reaches 100% detection for the first time so early. That curve also depends on other factors, like contrast, lighting conditions etc, but its profile is expected to be the same in all sequences.

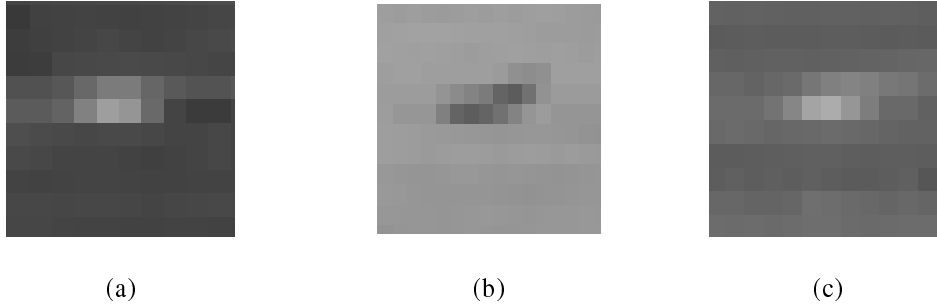


Figure 22: (a): Aircraft on collision course, (b)–(c): Target aircraft flying perpendicular to host aircraft

Frames	10-210	211-510	511-601
Plane Width in Pixels (Approximately)	3-6	6-10	10-13
Time to collision (sec)	36.87-30.2	30.2-20.2	20.2-17.2
Detection rate for 12x12 detector	60.95%	95%	100%
Detection rate for 18x18 detector	40.48%	89.67%	97.8%
Detection rate for 24x24 detector	50.48%	95%	91.21%

Table 2: Detection rate for three time periods from a collision course scenario

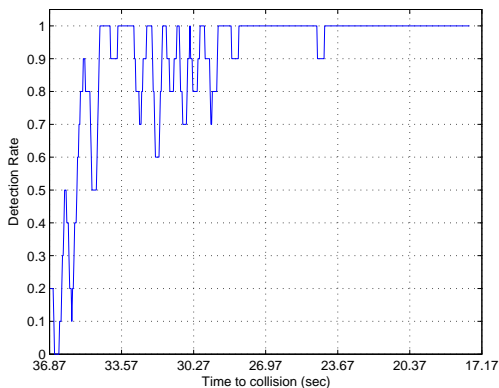


Figure 23: Detection rate as a function of time to collision

We have observed that the basic detector 12×12 is insensitive to scale changes, thus even though it was trained using samples with width from 5 to 8 pixels it can easily detect targets with width over 8 pixels. In addition, when the plane occupies more than 12 pixels it can still detect the wings since it detects objects with appearance similar to Figure 22(a). No matter how large is the target its wings (or at least a part of them towards the end) almost always look like Figure 22(a). Therefore using a multi-scale approach slightly improves the overall performance

but it generates more new false positives. Given that the detection rate of the basic detector when the width of the target is between 5 and 12 pixels is pretty good the inclusion of larger scales will mainly improve the performance in over 12 pixels cases. For targets with width less than 12 pixels it is expected that the performance will not be improved as shown in table 2. From Figure 6 we see that the plane occupies more than 12 pixels just 15 seconds before the collision occurs so even if the detection rate is improved that is not very useful. On the other hand, a 5 pixel plane (in width) corresponds to 40 seconds before the collision occurs which is a sufficient time for collision avoidance purposes. So it is more important to improve the performance of small targets (less than 5 pixels) which correspond to even further distances and allow more time to avoid a collision.

5.1 Collision course scenario

We tested the first classifier on the collision course maneuvers and the results are shown in Table 3, Figure 24 and Figure 25. The background of sequence 1 and 3 is smooth whereas the background of sequences 2 and 4 is heavily cluttered. Figure 4 shows a frame from sequence 2 together with the classifier's output (squares). So in this frame we see that the plane is correctly detected (top right) but there also four false positives.

We tested the classifier from the point that the target occupies at least 3 – 4 pixels since the performance of the classifier for targets less than 4 – 5 pixels is poor. In sequences 1 and 3 the classifier was stopped at the point where collision would occur if the aircrafts were flying at the same altitude.

By varying the threshold of the final classifier in the cascade and then by removing layers we can construct the classifier’s ROC curve. Figures 24 and 25 show the ROC curves for the second and fourth sequence respectively. The red line corresponds to the initial classifier with no temporal filtering used. The blue line corresponds to the use of temporal filtering as described above using only the previous frame and the green line corresponds to the use of temporal filtering using the last 2 frames. In the first case (sequence 2) the classifier ran for the part of the sequence between 25 seconds to collision and 10 seconds to collision whereas in the second case (sequence 4) the classifier ran for the part of the sequence between 20 seconds to collision and 5 seconds to collision. Because of the nature of the scene the target has a width of 3 – 4 pixels just 20 – 25 seconds before collision which is much different from the 45 seconds expected according to Figure 6.

In Figures 24 and 25 three different operating point for the classifier are shown. The first operating point represents the classifier we selected to use through the training procedure and consists of the first 11 layers of the cascade having an output threshold of 60.3025. The second operating point represents the same classifier with the addition of the temporal filtering. Finally, the third operating point represents the full classifier, consisting of all the 13 layers trained, but the output threshold is increased to 75.6 from 72.5084. Those three operating points are also considered in the crossing scenario.

Both sequence 2 and 4 are heavily cluttered so the use of the classifier with no temporal filtering (operating point 1) generates a significant number of false positives per frame, 7.9 and 14.67 respectively. However, the detection rate is very good 84.3% and 89.14% respectively. The use of temporal filtering (operating point 2) reduces the number of false positives per frame to 4.9 and 8.69 respectively as expected but on the other hand reduces the detection rate to less than 80%. This is the consequence of the nature of the scenes which leads to a non-consistent detection of the target when it is far away. The use of the classifier with the increased threshold (operating point 3) seems to work well in both cases. However, the results in the remaining sequences are worse than the other two operating points and in particular in sequence 1 where the background is smooth it achieves a detection rate of only 48.51%.

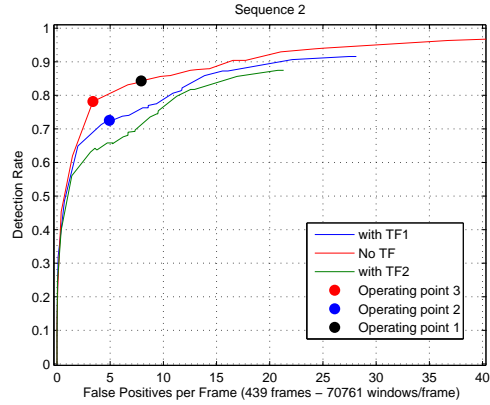


Figure 24: ROC curves for the 2nd collision course sequence in our data.

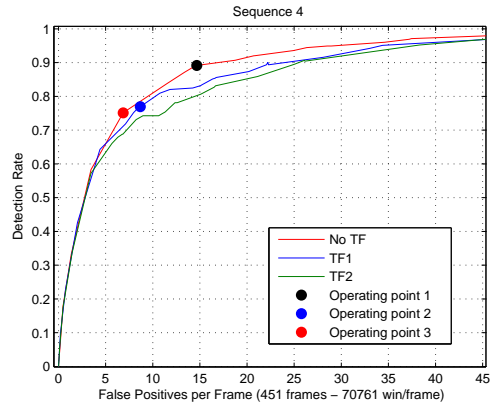


Figure 25: ROC curves for the 4th collision course sequence in our data.

The performance in sequence 3 is better than in sequence 1 because the contrast is low in the latter one. That also justifies the degraded performance of the temporal filtering since the target is not detected consistently when it is far away resulting in discarding the single-frame true detections. However, the most important reason for the degraded performance of the temporal filtering is that the target is not stationary in sequence 1 so the assumption for an almost stationary target is often violated and that also affects the performance of the temporal filtering.

So the first operating point has always the highest detection rate but also the highest false positive rate. The use of the temporal filtering reduces the number of false positives per frame but its performance on the detection rate depends on the nature of the scene, e.g. contrast, and

	Operating Point 1 Detection Rate/FP	Operating Point 2 Detection Rate/FP	Operating Point 3 Detection Rate/FP	Operating Point 4 Detection Rate/FP	Frames
Sequence 1	81.68%/3	66.67%/1	48.51%/0	75.41%/0	606
Sequence 3	96.38%/3	88.61%/0	84.27%/2	92.59%/0	1107

Table 3: Detection rates and number of false positives from the remaining collision course sequences (1 and 3).

	Operating Point 1	Operating Point 2	Operating Point 3	Operating Point 4
Avg. Detection Rate over all sequences	87.875%	76.32%	72.56%	81.75%
FP per frame	3.88	2.33	1.89	2.57

Table 4: Average detection rate and number of false positives per frame.

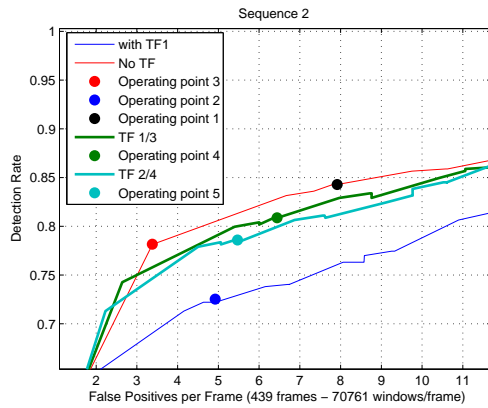


Figure 26: ROC curves for the 2nd collision course sequence in our data, using 2 more temporal filters.

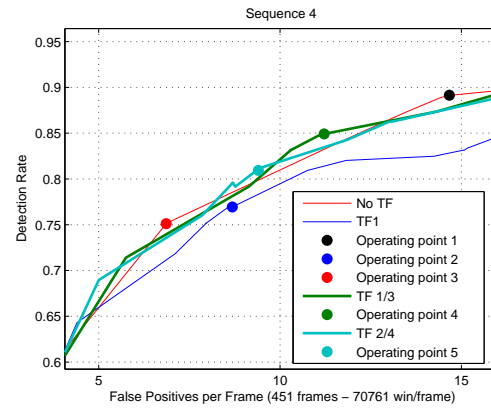


Figure 27: ROC curves for the 4th collision course sequence in our data, using 2 more temporal filters.

the assumption of a almost stationary target which is not always true. Finally, the third operating point achieves the best performance in the heavily cluttered sequences 2 and 4 but on the other hand has the worst performance on the low clutter sequences 1 and 3.

The use of the second temporal filtering approach, as described in section 3.5, results in the ROC curves shown in Figures 26 and 27. Those curves are very close to the ROC curves from figures 24 and 25 so we zoom in the area of interest only. After trying a series of different values for m and n (see section 3.5) we found that the set of values $m = 1, n = 3$ and $m = 2$ and $n = 4$ give good results. For example, $m = 1, n = 3$ means that in order to consider one detection as valid then at least one detection should occur in the current detection neighborhood in the last 3 frames. Comparing this approach with the previous tem-

poral filtering method we see that both sets of values result in a higher detection rate than the first temporal filtering method with a small increase in the false positive rate. However, since we do not have a reliable system for collision risk estimation which can further decrease the false positive rate we selected the first temporal approach (operating point 2) for comparison with the operating point 1 (no temporal filtering) in the remaining sequences, since it generates less false positive detections .

Apart from the four collision course sequences there were also six sequences where the target aircraft is flying directly away from the host aircraft. If we play those sequences backwards then it seems that the target aircraft is on collision course. The main drawback of this approach is that the time to collision does not correspond to a real collision course scenario. In such a scenario the relative

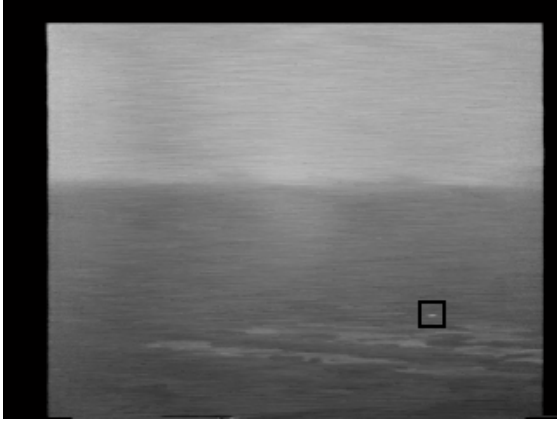


Figure 28: A plane successfully detected on a crossing maneuver.

velocity between the two aircrafts is the sum of their velocities. On the other hand, when the aircraft flies directly away then the relative velocity is the difference between the two velocities. So when we play that video sequence backwards the target aircraft approaches the host aircraft with a much lower speed. This results in an increased time to collision and this is obvious from the table 5 in which the number of frames used is much higher than the number of frames used in the real collision course sequences. This is explained by the fact that since the relative velocity between the two aircrafts is smaller then the rate of expansion will be smaller. In other words, the width of the target aircraft will be 5 pixels for a longer period of time than in a real collision course. This is true for any width, e.g. 6, 7 etc pixels. That also explains the increased detection rate in all cases. Even when a scene is cluttered, like sequence 6 or 10, the time that the width of the aircraft is between 5 and 12 pixels (in which case the detection rate is generally high) is increased resulting in a high detection rate.

5.2 Crossing scenario

We also tested the first classifier on the crossing target maneuvers and the results are shown in table 6. Sequences 1 and 4 were used for training and sequence 2 was used for validation so it is expected a high detection rate and a low false positive rate for those sequences. Figure 28 shows a frame from sequence 2 in which the plane is successfully detected and there are no false positives.

The remaining sequences (3 and 5) were used for training. The background of sequence 5 is almost uniform and that is why the target is detected in all frames with zero false positive alarms. Sequence 3 is heavily cluttered but

the performance of the classifier is very good. This may be the result of the similar background with sequence 4 and the lighting conditions seem to be the same. Temporal filtering (operating point 2), which assumes an almost stationary target, degrades the performance for that class of maneuvers since the target is not stationary but it is moving. Comparing the performance of operating points 1 and 3 we see that the former has a slightly better detection rate but on the other hand generates more false positives, which is consistent with both ROC curves (Figures 24 and 25). We should note here that most of the missed detections occur when the target enters or leaves the scene and therefore the classifier fails to detect a partial target.

5.3 False Positives

In order to get a better estimate of the false positive rate we tested the classifier on a series of sequences with medium to high clutter that they did not contain aircrafts. The results are shown in table 7. Two examples of highly cluttered scenes from sequence 3 and 9 are shown in Figures 29(a) and 29(b) respectively, where the false positive windows are also shown.

We see that the operating point 3 has a slightly better performance in terms of the number of false positives per frame. It is also obvious the effect of the nature of the scene to the performance of the operating points as mentioned also in the previous scenarios. There are some scenes where the second operating point clearly has an advantage over the third, for example sequence 1, but there are also some other scenes in which the third operating point is clearly better, for example sequence 3.

We also tried a fusion approach in this case. The output of the classifier at operating point 1 was added to the output of the second classifier trained as described in section 5.1. Then a new threshold was applied to the new sum, which was the sum of the thresholds of the two classifiers. Finally, temporal filtering was applied to the thresholded output. This is similar to the analog fusion described in [4]. The key point is that the 2 classifiers since they were trained on different data sets and they also use different filters they will generate false positives in different locations. Indeed, we see a reduction in the number of the false positives per frame to 1.7.

From all the results presented in this section we see that the number of false positives per frame is highly dependent on the background. A smooth background generates a small number of false positives whereas a heavily cluttered background generates a high number of false positives. That is expected since the classifier detects target-like objects so high cluttered background increases the

	Operating Point 2 Detection Rate/FP	Total number of frames
Sequence 5	83.15%/0	902
Sequence 6	90%/264	1181
Sequence 7	92.28%/0	1400
Sequence 8	88.47%/115	1301
Sequence 9	87.44%/0	1401
Sequence 10	85.98%/663	1270
Average	87.89	-

Table 5: Detection rate and number of false positives for the sequences that the plane is flying directly away from the host aircraft when they are played backwards.

	Operating Point 1 Detection Rate/FP	Operating Point 2 Detection Rate/FP	Operating Point 3 Detection Rate/FP	Frames with planes / Total number of frames
Sequence 1	98.45%/2	69.77%/0	98.45%/2	129/233
Sequence 2	97.65%/6	84.04%/0	94.84%/0	213/411
Sequence 3	97.09%/20	58.25%/6	97.09%/7	103/200
Sequence 4	99.34%/3	70%/0	99.34%/3	151/375
Sequence 5	100%/0	87.5%/0	100%/0	41/196
Average	98.506%	73.91%	97.94%	-

Table 6: Detection rates and number of false positives for the crossing target maneuvers.



Figure 29: Two images that do not contain planes. There are four false positives in each of them.

probability of regions with the same pixel distribution as a real target. In addition, we see that the detection rate is mainly affected by the contrast of the sequence and in smaller degree from the background. Ideally, the detection rate must be independent of the background, but that is not always true. For example it is easier to detect a dark target in a bright background than in a dark background.

5.4 Comparison with Other Systems

We used some of the sequences that McCandless [9] and Gandhi [6] used for their experiments. However, we can not directly compare our results since both of them pro-

vide results for a crossing target scenario only and we can not match the results from our crossing target sequences with their results since we do not know which sequences they used.

McCandless used two crossing target sequences with 270 and 160 frames respectively and reports a detection rate of 82% and 78% respectively with zero false alarms.

Gandhi reports the crossing target results shown in table 8 (MD = Misdetection rate, FA rate = number of false positives per frame). The performance in half of the sequences is very good, with a detection rate over 88% and zero false alarms whereas in the rest of the sequences the detection rate varies from 20% to 67% and there are two sequences with high number of false positives per frame. In all five crossing target sequences we have a detection rate over 97% with a very low false positive rate. However, that is not a fair comparison as 2 of the sequences were used for training and 1 for validation and only 2 of the sequences were used for testing as our focus was on collision course targets.

Carnie's results for a different set of sequences are shown in table 9. They correspond to a collision course target using two different morphological filtering approaches. They provide the first frame in which the target is detected, the first frame that consistent detection be-

	Operating Point 2 No FP/frame	Operating Point 3 No FP/frame	Fusion 1-2 No FP/frame	Total number of frames
Sequence 1	4.3104	6.1582	4.2269	335
Sequence 2	1.6228	1.3982	1.5509	334
Sequence 3	3.4617	1.5797	1.0973	483
Sequence 4	1.0463	0.77671	1.1116	627
Sequence 5	3.5757	2.9186	2.0957	700
Sequence 6	1.1496	0.9937	1.1386	635
Sequence 7	0.0259	0.0014	0	733
Sequence 8	0.98696	1.8087	0.96522	460
Sequence 9	0.46543	0.8887	0.45868	593
Sequence 10	1.9365	1.5102	1.9324	488
Sequence 11	6.73	5.3086	4.1543	700
Average	2.301	2.12	1.703	-

Table 7: Number of false positives per frame for 11 eleven sequences that do not contain planes.

Distance		# Frames	MD rate	FA rate
<i>nmi</i>	<i>km</i>			
1.5	2.78	120	0.061	0.000
1.8	3.33	130	0.113	0.000
2.0	3.70	150	0.394	0.000
2.4	4.44	210	0.059	0.000
3.0	5.55	210	0.056	0.000
4.7	8.70	300	0.335	0.183
5.0	9.26	340	0.803	0.147
5.4	10.00	410	0.643	0.000

Table 8: Gandhi’s results [6] for crossing target maneuvers.

gins and the number of false positives per frame. They do not provide detection rates or the length of their sequence. The lowest number of false positives per frame they report is 2.62 but since no detection rates are provided we can not compare our results. In addition, their detection algorithm detects the plane at distances which are 35 – 40% greater than those of the human observer [3].

Although we can not directly compare our results with other methods we should note that we tested our method in over 10000 frames and in order to estimate the number of false positives per frame we used an additional 6000 frames approximately which contain no planes. Those numbers are much higher than the number of frames used for testing by most of the existing methods which they use a few hundred test frames.

COMPARISON BETWEEN MORPHOLOGICAL FILTERING APPROACHES				
	τ	CMO Approach	PS Approach	% Change
First Detection (Frame #)	0.025	5	5	-
	0.030	20	20	-
	0.035	53	53	-
Consistent Detection (Frame #)	0.025	144	144	-
	0.030	171	171	-
	0.035	190	190	-
Total False Alarms / Frame	0.025	15.9	11.9	-24.8%
	0.030	4.86	4.85	-0.3%
	0.035	2.62	2.62	-
Intermittent False Alarms / Frame	0.025	4.82	1.4	-70.9%
	0.030	0.585	0.60	2.5%
	0.035	0.34	0.34	-

Table 9: Carnie’s results [3] for a collision course scenario.

6 Conclusions

In this paper a different approach to the problem of aircraft detection was presented. Instead of using an image processing method we used a learning method to learn the targets from real data. The system described here is a very popular method in the area of face detection and with few modifications was successfully used in the problem of target detection in video sequences. The advantage of this method is that it detects the targets on a frame to frame basis with a high detection rate, usually around 80%, and lower false positive rate than the commonly used morphological filtering which is susceptible to false positives. Although our system can not be directly compared with other systems since it was tested on different video se-

quences the fact that achieves a false positive rate which usually lies between 1 and 5 false positives per frame, and in some cases up to 10, without further processing is an encouraging result. Gandhi et al [6] track the detections over 100 frames and Carnie et al [3] use a dynamic programming stage to reduce the number of false positives generated by the morphological filtering. The above results were obtained using over 15000 test frames which is a much higher number than the typical number of frames used by the majority of the detection systems. The described system has also the potential of real time execution. Viola and Jones reported an execution time of 15 frames per second which is a desired property for a system that must work on real time. In addition, this system is suitable for the detection of both moving and stationary targets since the classifier detects potential targets in a static frame. Of course, the use of temporal information is beneficial, as it further reduces the false positive rate. Another advantage is that it is very easy to change the operating point of the classifier which is a desired property for a system and can lead to the creation of an adaptive detector that adjusts its thresholds depending on the lighting conditions and the image contrast.

We were mostly focused on the detection part and developed a detection framework that works on a frame to frame basis. In this work we used very simple temporal filters which take into account only the previous 1 to 4 frames so future work includes the use of more complicated temporal filters or tracking algorithms which are expected to further enhance the system's performance. The consistent detection of targets which occupy less than 5 pixels is something that has to be addressed too. In addition the use of contextual information should be very beneficial to the system since it is one of the main sources of information that helps humans to reliably detect targets. Finally the development of a reliable system to estimate the collision risk is an important issue that should be addressed in order to have a final collision avoidance system.

References

- [1] Y. Barniv. Dynamic programming solution for detecting dim moving targets. *IEEE Trans. On Aerospace and Electronic Systems*, 21, 1 (Jan. 1985), 144-156
- [2] L. Bourdev and J. Brandt. *Robust object detection via soft cascade*. In Proc. CVPR, 2005
- [3] R. Carnie, R. Walker and P. Corke. *Image processing algorithms for UAV "Sense and Avoid"*. In Proc. ICRA, p.2848-2853, 2006
- [4] D. Casasent and A.Ye. Detection filters and algorithm fusion for ATR. *IEEE Trans. On Image Processing*, VOL. 6, NO. 1, p. 114-125, January 1997
- [5] Federal Aviation Administration, "Chapter 12, Section 9, Remotely Operated Aircraft," in *Order 7610.4: Special Military Operations*. Washington D.C., USA: US Government Printing Office
- [6] T. Gandhi, M. T. Yang, R. Kasturi, O. Camps, L. Coraor and J. McCandless. *Detection of obstacles in the flight path of an aircraft*. *IEEE Trans. On Aerospace and Electronic Systems*, VOL. 39, NO. 1, January 2003
- [7] R. Lienhart and J. Maydt. *An extended set of haar-like features for rapid object detection*. In Proc. ICIP, 2003
- [8] B. Moghaddam and A. Pentland. Probabilistic visual learning for object representation. *IEEE Trans. On Pattern Analysis and Machine Intelligence*, VOL. 19, NO. 7, p.696-710, July 1997
- [9] J.W. McCandless. Detection of aircraft in video sequences using a predictive optical flow algorithm. *Optical Engineering*, 3: 523-530, 1999
- [10] K. Nishiguchi, M. Kobayashi and A. Ichikawa. Small target detection from image sequences using recursive max filter. In Proc. Of SPIE, 2561 (July 1995), 153-166
- [11] S. Rydbergard. Obstacle detection in a See-and-Avoid System for Unmanned Aerial Vehicles. *MSc Thesis*, Royal Institute of Technology (KTH), Sweden
- [12] H. Schneiderman and T. Kanade. Object detection using the statistics of the parts. *International Journal of Computer Vision* 56(3), 151-177, Kluwer, 2004
- [13] J. Sun, J.M. Rehg and A. Bobick. Automatic cascade training with perturbation bias. In Proc. CVPR, pages 1063-69,2004
- [14] K. Sung and T. Poggio. Example-based learning for view-based human face detection. *IEEE Trans. On Pattern Analysis and Machine Intelligence*, VOL. 20, NO. 1, p.39-51, July 1998
- [15] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In Proc. CVPR, pages 511-518, 2001

- [16] P. Viola and M. Jones. Robust real time object detection. *Second international workshop on statistical and computational theories of vision - modeling, learning, computing and sampling*. Vancouver, July 2001
- [17] P. Viola, M. Jones and D. Snow. Detecting pedestrians using patterns of motion and appearance. *International Journal of Computer Vision* 63(2), 153-161, Springer, 2005
- [18] R. Xiao, L. Zhu and H.J. Zhang. Boosting chain learning for object detection. *In Proc. ICCV*, 2003