Generic Object Detection using AdaBoost

Ben Weber

Department of Computer Science
University of California, Santa Cruz
Santa Cruz, CA 95064
bweber@soe.ucsc.edu

Abstract

Boosting has been shown to be an effective technique for face detection. This paper explores the implementation of a real-time object detection system as described by Viola and Jones. The system is initially tested on the problem of face detection and then applied to the problem of aircraft detection. Initial results show that detecting aircraft with high accuracy comes at the cost of many false detections. Reducing the number of false positives requires more training and possibly more complex classifiers.

1 Introduction

One of the main challenges of computer vision is efficiently detecting and classifying objects in an image or video sequence. Several machine learning approaches have been applied to this problem, demonstrating significant improvements in detection accuracy and speed. However, most approaches have limited object detection to a single class of objects, such as faces or pedestrians. Recent work utilizing boosting has shown that boosting may be applicable to multi-class detection problems.

A common benchmark for computer vision researchers is face detection. Given a set of images, a face detection algorithm determines which images have sub-windows containing faces. This task is trivial for humans, but is computationally expensive for machines. Most face detection systems simplify the face detection problem by constraining the problem to frontal views of non-rotated faces. Approaches have been demonstrated capable of relaxing these constraints, at the cost of additional computation.

Viola and Jones present an efficient face detection system in [12]. They utilize AdaBoost [4] to select a set of features and train a classifier. The detector uses a cascade structure to reduce the number of features considered for each sub-window. This approach is significantly faster than previous techniques and is applicable for real-time systems. Since boosting is used to select features for the classifier, the detection system is applicable to additional object classes. This paper explores the application of Viola and Jones detection algorithm to aircraft.

The remainder of this paper is structured as follows. Section 2 gives a summary of related work. Section 3 describes the object detection algorithm using AdaBoost. Section 4 discusses the implementation of AdaBoost for face detection. Section 5 gives a summary of results for aircraft detection. Finally, Section 6 provides conclusions.

2 Related Work

Several approaches have been applied to object detection: support vector machines [7], neural networks [9], example-based learning [11], and boosting [6]. Generally, there is a tradeoff between the speed of object detection and accuracy. AdaBoost has produced the best results for face detection, identifying faces with 95% accuracy while reporting only 1% false positives [12].

There are several extensions to the face detection problem. The face recognition problem requires an algorithm to classify the identity of a face. Gou and Zhang [5] apply AdaBoost to the face recognition problem using a majority voting strategy. The approach uses a constrained version of majority voting to reduce the number of pairwise comparisons without losing recognition accuracy. Another face detection problem is handling instances where frontal views of faces can be rotated. One approach to this problem is the use of steerable filters [3] to adaptively "steer" a filter to any orientation. The face detection problem can also be extended to include side profiles of faces. Schneiderman and Kanade present a solution to this problem by training separate classifiers on different views of faces and combining the results of the classifiers [10].

Object detection has been applied to real-time systems. Viola and Jones extend the AdaBoost algorithm to real-time face detection in [13]. Papageorgiou et al. demonstrate the use of support vector machines for face and pedestrian detection in a real-time system [8]. They note that pedestrians are harder to detect due to greater variations in color, textures, and patterns. Real-time systems can employ motion-based techniques to speed up object detection by suggesting areas of an image likely to contain objects based on previous frames.

The object classification problem requires a system to classify the type of an object in an image. Often, images contain a single distinct object and a sparse background. Opelt et al. show promising results for the object classification problem using boosting [6]. An even more complex problem is the generic object recognition problem. Generic object recognition requires a system to detect and classify several types of objects in a single image. One possible solution to this problem is to combine the techniques presented in [6] and [10]. Such a system would be computationally expensive, but capable of detecting and classifying several types of objects.

One of the main problems facing object recognition is collecting adequate training data. Robust object detection requires a diverse database of human-annotated examples. For many problem domains, constructing such a database is impractical. Abramson and Freund reduce the amount of annotation required by only providing annotations for hard to classify examples [1]. Therefore, training becomes an active learning process where misclassifications indicate the hard problems. Another challenge is finding negative examples that lead to efficient training of a classifier. Papageorgiou et al. solve this problem by using false positives as negative examples [8].

3 Object Detection

The object detection system uses a strong classifier to determine if sub-windows in an image contain a specified object. The strong classifier is composed of a set of weak learners with associated weights. Each weak learner uses a single image feature to produce a hypothesis. Viola and Jones show that AdaBoost can be used to both select a small subset of features and train the classifiers [12]. There are four steps to building an object detection system with AdaBoost: select a dataset with positive and negative training examples, train the threshold values for each feature, select and train a subset of the classifiers and train the attentional cascade. Once the detector is built, images are exhaustively scanned at all locations and scales to identify objects.

3.1 Image Features

The object detection system uses simple features to build weak learners. Viola and Jones suggest the use of features rather than pixel values directly, because it enables encoding of ad-hoc domain knowledge that is difficult to learn from a finite set of training data [12]. Also, a feature-based system is much faster than a pixel-based system. The features utilized are rectangular features similar to Haar basis functions, which have been used by Papageorgiou et al. [8].

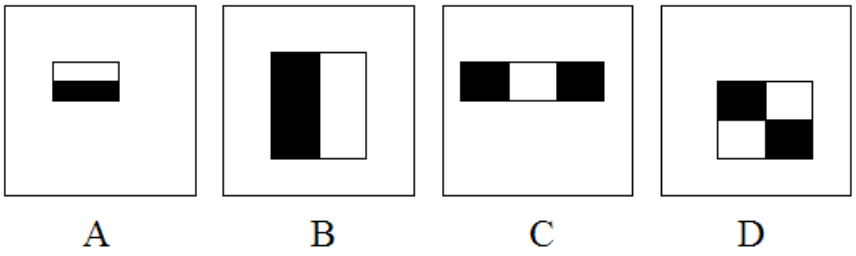

Figure 1 - Example rectangular features: two-rectangle features in (A) and (B), a three-rectangle feature in (C) and a four-rectangle feature in (D). Figure adapted from [12]

Features compute a value by subtracting the sum of the pixels in the dark rectangular regions from the sum of the pixels in the light rectangular regions. Three types of features are used for object detection. Two-rectangle features compute the difference between two vertically or horizontally adjacent regions of the same shape and size. Three-rectangle features compute the difference between three vertically or horizontally aligned rectangles. Four-rectangle features compute the difference between diagonally paired rectangles. The exhaustive set of features for the base resolution of the detector is over 270,000 features. Examples of features are shown in Figure 1.

3.2 Integral Image

An integral image representation is used to provide constant lookup times for the sum of pixels in rectangular regions of an image. This representation enables rectangular features to be computed using a minimal number of array references. The value of an integral image at location x, y contains the sum of the pixels above and to the left of x, y.

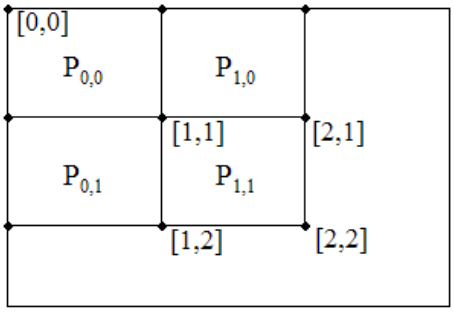

Figure 2 - Encoding of the integral image. ii(0, 0) = 0, ii(1,1) = $P_{0,0}$, ii(2,1) = $P_{0,0}$ + $P_{1,0}$ ii(1,2) = $P_{0,0}$ + $P_{0,1}$, ii(2,2) = $P_{0,0}$ + $P_{1,0}$ + $P_{0,1}$ + $P_{1,1}$

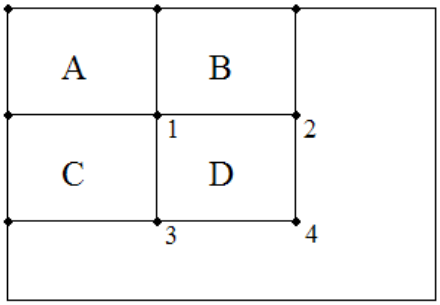

Figure 3 - Rectangular regions of an integral image. Adapted from [12]

The value of an integral image at location x,y is computed as: $ii(x, y) = \sum_{x' \leq, y' \leq y} i(x', y')$, where i(x, y) is the original pixel value and ii(x, y) is the integral image [12]. Viola and Jones show that the following recurrences can be used to efficiently compute the integral image in a single pass over the original image:

$$s(x, y) = s(x, y - 1) + i(x, y)$$
$$ii(x, y) = ii(x - 1, y) + s(x, y)$$

where s(x, y) is the cumulative row sum and s(x, 0) = 0 and ii(0, y) = 0. The top left corner of an integral image is shown in Figure 2. The variable $P_{x,y}$ refers to the pixel value of an image at location x, y. Viola and Jones show that the sum of pixels in a rectangular region can be found in four array references. The value of D in Figure 3 can be computed as 4 + 1 – (2 + 3). Therefore, the time required to compute a feature is not dependent on its size.

3.3 Weak Learners

The object detection system uses weak learners constrained to evaluating a single feature. For each feature, the weak learner determines the optimal threshold classification function, such that the minimum number of examples are misclassified [12]. A weak learner consist of a feature, f_j, and a threshold, θ_j:

$$h_j(x) = \begin{cases} 1 & f_j(x) < \theta_j \\ 0 & \text{otherwise} \end{cases}$$

The best weak learner has a misclassification rate of approximately 0.07. The strong classifier combines several weak learners to produce more accurate hypotheses.

3.4 Classifier Training

AdaBoost is used to find the best weak learners and the corresponding weights for these classifiers. The boosting algorithm maximizes the margin between a set of positive and negative examples. Pseudocode for the boosting algorithm is shown in Table 1. The algorithm is first given a set of positive and negative examples. Each of the examples is converted to grayscale, scaled to the base resolution of the detector and annotated with a

1 or 0 for positive and negative examples respectively. Next, the algorithm creates a weight vector for the examples. The initial weights are dependent on the number of positive and negative examples. If the number of positive and negative examples is equal, then the algorithm starts with a uniform weight vector.

The boosting algorithm performs a series of trials, each time selecting a new weak learner. At the beginning of each trial, the weights are normalized to sum to 1. Next, the algorithm selects the weak learner that produces the smallest misclassification error with respect to the weight vector. This step requires classifying all of the examples with the over 270,000 weak learners and is computationally expensive. Fortunately, the threshold value for each weak learner needs to be computed only once, because the hypothesis of a weak learner does not consider the weight vector. The best weak learner is then selected and used to update the weight vectors. The weights of correctly classified examples are multiplied by β, while the weights of misclassified examples do not change.

- Given example images $(x_1, y_1), \ldots, (x_n, y_n)$ where $y_i = 0, 1$ for negative and positive examples, respectively.
- Initialize weights $w_i = 1/2M, 1/2L$ for $y_i = 0, 1$ respectively, where M and L are the number of negatives and positives.
- For t = 1 to T
  1. Normalize the weights, so that $\sum_{i=1:n} w_i = 1$
  2. Choose the classifier, h_j, with the lowest error ε_j:

     $$\varepsilon_j = \sum_i w_i |h_j(x_i) - y_i|.$$

  3. Update the weights for each example:

     $$w_i = \begin{cases} w_i \beta_t & \text{example } x_i \text{ is classified correctly} \\ w_i & \text{otherwise} \end{cases}$$

     where $\beta_t = \varepsilon_t/(1 - \varepsilon_t)$ and $\alpha_t = - \log \beta_t$
- The final strong classifier is:

  $$h(x) = \begin{cases} 1 & \sum_{t=1:T} \alpha_t h_t(x) \geq \lambda \sum_{t=1:T} \alpha_t, \text{ where } \lambda = ½ \\ 0 & \text{otherwise} \end{cases}$$

Table 1 – Pseudocode for the AdaBoost algorithm. Adapted from [12]

Combined with normalization, this update results in most of the weight being placed on hard to classify examples. Therefore, as the number of trials increases, the error rates also increase. This leads to smaller α values for weak learners selected later in the training process. The final classification function is the sum of the predictions of the selected weak learners multiplied by the corresponding α values.

3.5 Attentional Cascade

The object detection system exhaustively scans all sub-windows in an image. Evaluating all sub-windows becomes intractable when the strong classifier selects several thousand features. A method to overcome this problem is the use of a degenerate decision tree to limit the number of features computed for each sub-window [2]. Several filters are used to build decision nodes for the tree. If any filter in the tree rejects a sub-window, then the sub-window if rejected. The structure of the decision tree is shown in Figure 4.

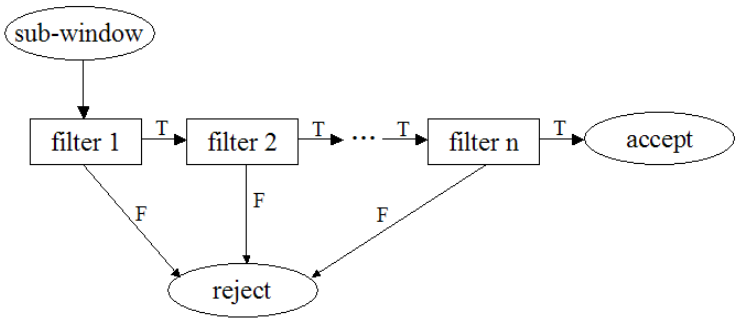

Figure 4 - Attentional cascade. Adapted from [12]

The decision tree is constructed such that the first filter evaluates a small number of features and the later filters add more and more features. The threshold values of features in each filter must be modified to avoid discarding positive sub-windows. This is achieved by increasing the threshold value, but this process leads to higher false positive rates. Therefore, the purpose of each filter in the decision tree is to progressively discard the harder to classify false positives. A well constructed decision tree significantly reduces the number of features evaluated for each sub-window while maintaining accuracy close to the exhaustive approach.

3.6 Scanning for Objects

The system detects objects by exhaustively scanning images. For each input image, the detector first converts the image to grayscale and computes the integral image. Next, the detector starts with an initial scale of 1.0 and evaluates every sub-window with the strong classifier. The scale is then increased and all sub-windows are evaluated at the new scale. The detector efficiently computes features at different scales by scaling the features, not the image. The scale is increased until the detection window is larger than the image width or height. A sub-window is marked as an object occurrence if the strong classifier returns a value of 1.

4 Implementation

Viola and Jones' object detection algorithm was utilized to implement a face detection system. Building the face detection system required obtaining an annotated dataset of faces, finding optimal threshold values for the features, training the classifiers and building a degenerate decision tree. Additionally, the system was extended to detect faces in real-time using a camera. The system is implemented in Java and detects faces at a rate of 5-10 frames per second for images with a resolution of 320x240. The detector uses a base resolution of 24x24 pixels.

4.1 Face Dataset

The CalTech 101 dataset was used to train the face detection system (available at http://www.vision.caltech.edu/archive.html). The dataset contains pictures of objects belonging to 101 categories, with each category containing 40 to 800 images. The face category contains over 400 images of frontal views of faces from approximately 30 different people. The dataset contains bounding box annotations for the locations of the faces. Other categories in the dataset include vehicles and animals. Four hundred images were selected from the face category as positives for the training dataset. Mirror images of the faces were also used, resulting in 800 positive examples. The positive examples were scaled down to the base resolution of 24x24 using the bounding box annotations. An example face scaled down to the base resolution is shown in Figure 5. Eight hundred images were selected at random from different categories for negative examples. One random sub-window was selected from each negative example and scaled down to the base detector resolution. A subset of the positive examples is shown in Figure 6 and a subset of the negative examples is shown in Figure 7.

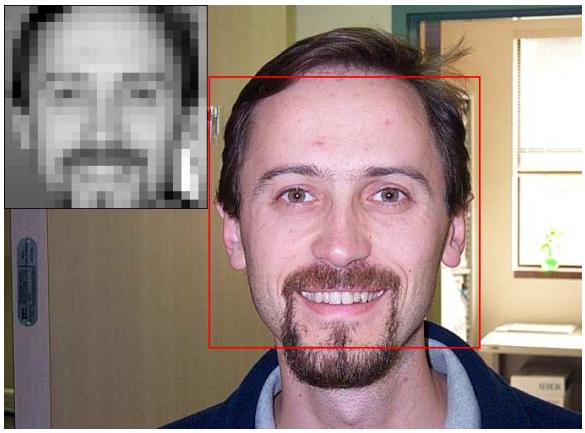

Figure 5 - Bounding box annotations are used to scale examples to the base resolution

Figure 6 - Positive examples for face detection

Figure 7 - Negative examples for face detection

4.2 Training the Detector

Training the detector consists of two steps: finding optimal threshold values for the features and using AdaBoost to select a subset of the features. An approximation was used to find the optimal threshold value for each feature, since the number of features is large. The value returned by a two-rectangle feature must be in the range (-1, 1) and the value returned by a three-rectangle or four-rectangle feature must be the range (-2, 2). The detector chose the best threshold value for each feature out of 50 linearly spaced values within these ranges.

The face detection system was trained to select the 2000 best features using the algorithm described in Table 1. The first three features selected by the AdaBoost algorithm are shown in Figure 8. The first feature detects the fact that a person's hair color is usually darker than their complexion. The second feature corresponds to the first feature reported by Viola and Jones, which detects that the area around the eyes is darker than the region below the eyes. The third feature appears to rely on the fact that the background is usually lighter than the hair color.

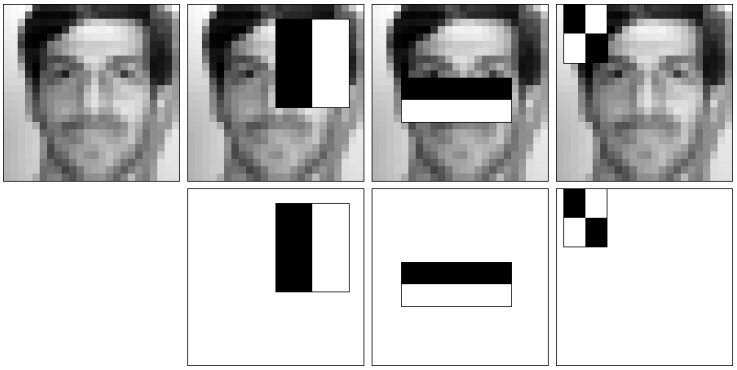

Figure 8 - First three features selected by AdaBoost

4.3 Real-Time Face Detection

The initial face detection system used the strong classifier described in Table 1. The classifier produced a large number of false positives when $\lambda = ½$. Increasing this value reduces the number of false positives, but also decreases the accuracy of the detector. It is also possible to modify the feature thresholds. However, increasing the threshold values results in a much slower detector, because in general more features are evaluated for each sub-window. Therefore, several filters were constructed to implement a degenerate decision tree. Each layer was trained to meet a target rate of false positives while maintaining an acceptable detection rate. The configuration for the filters is shown in Table 2. The resulting detector is not only more accurate, but also significantly faster.

	Filter 1	**Filter 2**	**Filter 3**	**Filter 4**	**Filter 5**	**Filter 6**	**Filter 7**
Number of features	1	5	10	50	200	1000	2000
Threshold increase	0	0.01	0.01	0.02	0.02	0.03	0.03
λ	0.55	0.75	0.75	0.75	0.8	0.8	0.75

Table 2 - Configuration of the decision tree for face detection

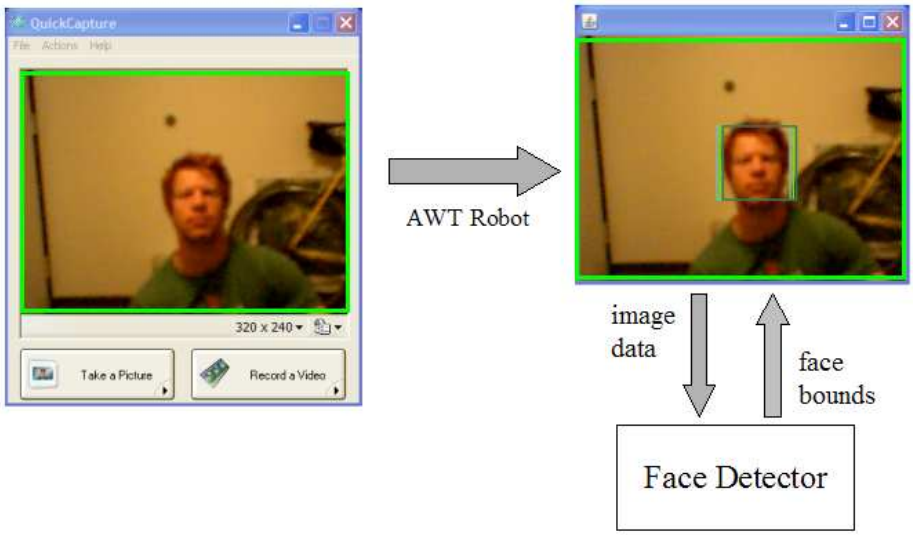

Figure 9 - Real-time system using screen captures to analyze camera data

The cascading face detector was used to implement real-time face detection. The system uses the java.awt.Robot class to perform screen-captures of a program displaying a camera feed. The screen capture is immediately drawn to the panel. The face detector runs in a separate thread and constantly checks the most recent image for faces. The detector then returns the results to the panel and the panel draws bounding boxes for the faces. Since the detector runs in a separate thread and does not force synchronization with the screen-capture, there is a face detection delay of approximately 100-200 milliseconds.

4.4 Face Detection Results

The accuracy of the face detector was analyzed on images files from the CMU Testset C (available at http://vasc.ri.cmu.edu/idb/html/face/). The dataset contains a variety of faces in different lighting conditions and is considered to be a difficult dataset. Each image was exhaustively scanned using a starting scale of 1.0, $\Delta scale = 1.25$ and a shift value equal to the current scale. For an image with a resolution of 320x240, there are 70428 sub-windows to check. The detector was tested on 65 images containing 182 faces. The detector correctly identified 54.3% of the faces and reported 83 false positives.

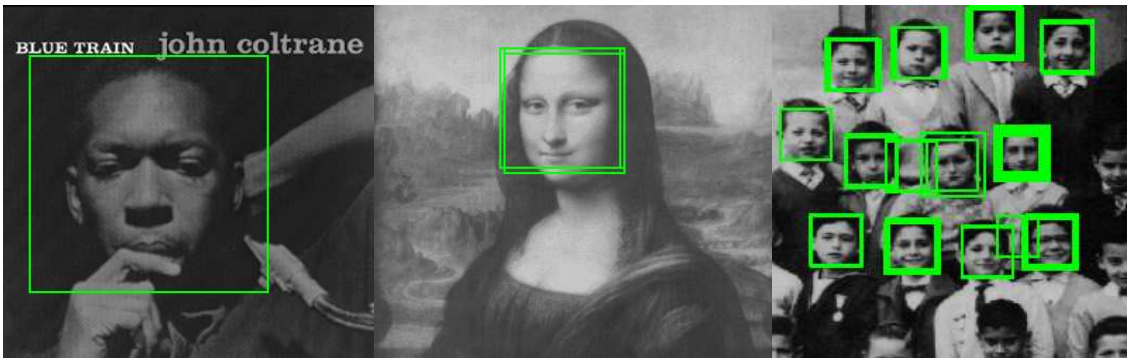

Figure 10 - Correctly identified faces on the CMU dataset

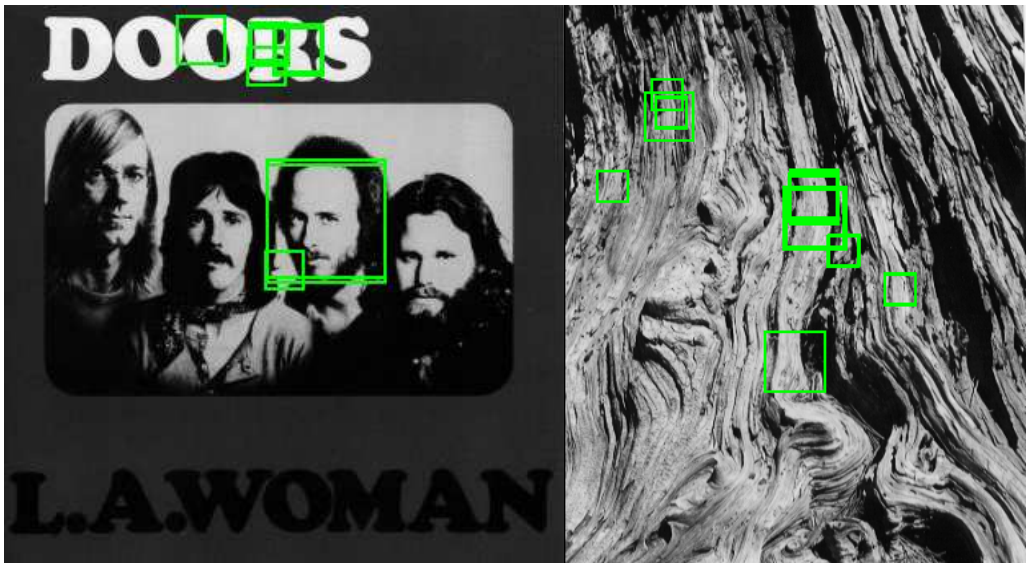

Figure 11 - False positives on the CMU dataset

5 Results

The object detection system was applied to the problem of aircraft detection. The system was trained on images from the aircraft category of the CalTech 101 dataset. A subset of the training images is shown in Figure 12. The 800 images were partitioned into two groups and separate classifiers were trained for both groups. The strong classifiers were then tested on the opposite dataset to validate the performance of the detectors. Each strong classifier used 5000 weak learners. A cascading approach was used, but no modifications were made to the threshold values. The configuration of the cascading detector is shown in Table 3.

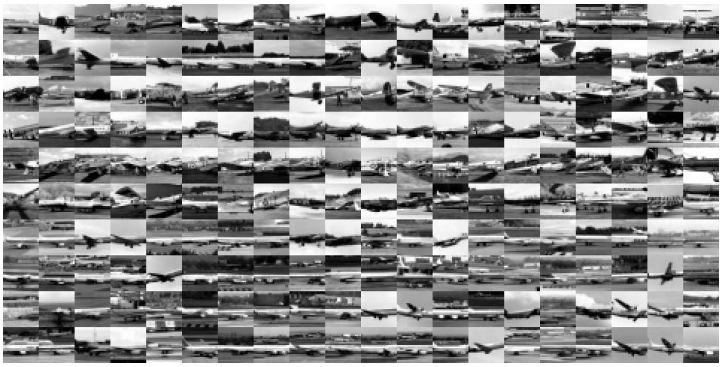

Figure 12 - Positive examples for the aircraft detector

ROC curves were generated by comparing the number of correctly identified aircraft versus the number of false positives. The false positive rate was computed as the ratio of images with false positives to the number of total test images. Therefore, multiple false positives on a single image did affect the ROC curve. Points on the ROC curve were generated by varying the value of λ. Increasing λ decreases the number of detections and false positives, while decreasing λ increases the number of detections and false positives.

The bottom left corner of the curve corresponds to $\lambda = 1$ and the top right of the curve corresponds to $\lambda = 0$.

	Filter 1	Filter 2	Filter 3	Filter 4	Filter 5	Filter 6	Filter 7
Number of features	1	5	10	50	200	1000	2000
Threshold increase	0	0	0	0	0	0	0
λ	λ	λ	λ	λ	λ	λ	λ

Table 3 - Configuration of the decision tree for aircraft detection

ROC curves for the aircraft detectors on the holdout datasets are shown in Figure 13. The detector for the first dataset did not show a noticeable difference between the training and testing datasets. This resulted from the first dataset being harder to classify than the second dataset. The detector for the second dataset demonstrated a difference between accuracy on the training and testing datasets. The second detector performed poorly on the test dataset, because the first dataset was more difficult to classify.

Overall, the ROC curves show that the number of false positives is approximately proportional to the number of correct detections. However, this is much better than a purely random classifier. The random classifier curve was generated using random predictions. The random detector marks a sub-window as positive if a random value is below a threshold. Several threshold values were used to generate the different points on the random classifier curve. The random detector has an accuracy of approximately 10%, which corresponds to the ratio of sub-windows in the dataset that are considered correct detections.

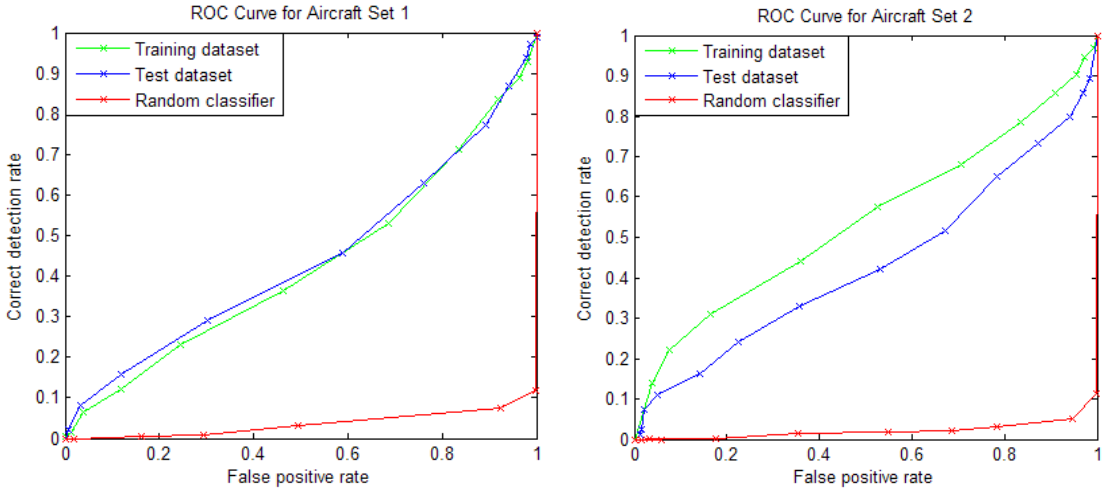

Figure 13 - ROC curves for the aircraft datasets

The strong classifier for the second dataset was tested on a validation dataset. The dataset consists of 10 images obtained from the web containing 20 aircraft. The dataset also contains 10 images randomly selected from the CalTech 101 dataset that do not contain aircraft. The results for the validation dataset are shown in Figure 14. The results show that the detector is incapable of accurately detecting aircraft without a reporting a large number of false positives. Example detections on the validation dataset are shown in Figure 15.

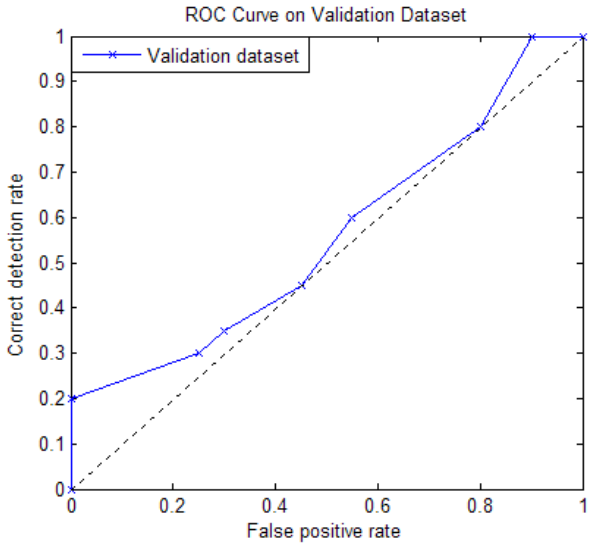

Figure 14 - ROC curve for the validation dataset

Figure 15 - Aircraft detections on the validation dataset

6 Conclusion

Generic object detection is one of the main challenges of computer vision. Boosting has been shown to be an effective technique for face detection and has been applied to other problems such as pedestrian detection. This paper has explored the use of AdaBoost for aircraft detection. The initial results show accurately identifying aircraft comes at the cost of many false positives. There are many explanations for the flat ROC curves demonstrated. It is expected that the accuracy of the detector will increase with more training data. Also, adding more features to the strong classifier and adding more filers to the decision tree will improve performance. However, constraining the classifier to rectangular features imposes a limit on the accuracy of the detector.

Aircraft detection provides a challenging problem for several reasons. There are many different classes of fixed-wing aircraft, such as commercial jetliners, kit planes and jets. Additionally, the examples contained aircraft at several different orientations. While the yaw was constant for the training images, the pitch and roll of the aircraft varied.

AdaBoost is potentially applicable to many classes of object detection problems. However, simpler approaches such as color-similarity techniques perform comparably with boosting for aircraft detection. One of the main problems with boosting is the amount of computation required to train strong classifiers. If more complex features are considered, then training becomes intractable. Future work should explore the use of more complex features and the use of heuristics for feature selection. Additionally, future work could explore the use of parallel computing to detect several classes of objects in real time.

References

1. Abramson, A. and Freund, Y. Active Learning for Visual Object Recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2005.

2. Fleuret, F. and Geman, D. Coarse-to-fine face detection. In *International Journal of Computer Vision*, 2001.

3. Freeman, W. T. and Adelson, E. H. The design and use of steerable filters. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(9), pp. 891-906, 1991.

4. Freund, Y. and Schapire, R. E. Experiments with a new boosting algorithm. In *Machine Learning: Proceedings of the Thirteenth International Conference*, pp. 148-156, 1996.

5. Guo, G. D. and Zhang, H. J. Boosting for fast face recognition. In *IEEE ICCV Workshop on Recognition, Analysis, and Tracking of Faces and Gestures in Real-Time Systems*, pp. 96-100, 2001.

6. Opelt, A., Pinz, A., Fussenegger, M. and Auer, P. Generic Object Recognition with Boosting. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(3), pp. 416-431, 2006.

7. Osuna, E., Freund, R. and Girosi, F. Training support vector machines: an application to face detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, vol. 24, 1997.

8. Papageorgiou, C., Oren, M. and Poggio, T. A general framework for object detection. In *International Conference on Computer Vision*, pp. 555-562, 1998.

9. Rowley, H., Baluja, S. and Kanade, T. Neural network-based face detection. In *IEEE Pattern Analysis and Machine Intelligence*, vol. 20, pp 22–38, 1998.

10. Schneiderman, H. and Kanade, T. Object detection using the statistics of parts. In *International Journal of Computer Vision*, 56(3), pp. 151–177, 2004.

11. Sung, K. and Poggio, T. Example-based learning for viewbased face detection. In *IEEE Pattern Analysis and Machine Intelligence*, 20(1), pp. 39–51, 1998.

12. Viola, P. and Jones, M. Rapid object detection using a boosted cascade of simple features. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2001.

13. Viola, P. and Jones, M. Robust Real Time Object Detection, In *International Journal of Computer Vision*, 1(2), 2002.