CS 4337 Final Project Report
AdaBoost Face Detection

Cameron Chalupa
Ethan Reed
Zoe Becker
Department of Computer Science
Texas State University
78666 San Marcos, Texas

1 Abstract

Accuracy and efficiency go hand in hand when developing a facial detection algorithm; we set out to create a program that prioritizes both. Using skin detection and a cascading Adaptive Boosting algorithm we aimed to preserve the accuracy of the model while drastically reducing computation time. Our test data was composed of cropped grayscale images of faces, cropped grayscale images of nonfaces, and colored images of different sizes and resolutions. The facial detection metrics that we set out to return were: accuracy on grayscale faces, accuracy on grayscale nonfaces, accuracy on colored images, and the number of false positives returned on colored images. Along with the accuracy we also wanted to analyze the performance differences based on the use of cascades and skin detection(colored images only).Our final results were as follows: Grayscale Faces: 97.14%, Grayscale Nonfaces: 99.87%,Colored Images: 42.857% accuracy, 54 false positives, 46.62s execution time. If we were to remove the cascading and skin detection aspects of our algorithm, the total execution time of our test data took 277.78 seconds. This displays the 5.96 speedup that our model achieved compared a facial detection algorithm that solely employed the Adaptive Boosting algorithm.

2 Introduction

Facial detection algorithms have many applications from Snapchat filters to airport security. While highly accurate methods are crucial, achieving a balance between accuracy and efficiency is arguably more important. In our facial detection implementation, we leverage skin detection with cascading AdaBoost models to deliver a solution that strikes a balance–providing both precision and computational speed.

3 Methodology

3.1 Pre-processing

We felt that the data we were given needed some adjustment in order to train our model more accurately. For example, many of the cropped faces still contained some background in them, and the nonface images were sized differently than the cropped faces. In order to keep everything consistent, we applied the following adjustments to both the training sets and the test sets.

3.1.1 Non Face Data

In the training set we were given the non face data consisted of plain, uncropped images. There were also many times fewer non face examples than face examples. To remedy this training imbalance we decided to take multiple smaller windows of the same scale as the training faces from the larger plain image. We took 130 random random windows from each of the non face images to come up with 3510 non face examples, slightly more than the 3049 training faces. Finally, the non face images were in color, so we converted them to grayscale to match the training faces, which were all in grayscale.

3.1.2 Face Data

For the cropped face data we noticed that there tended to be some background visible in many of the images. We decided to crop these images even further so that the model was only training on facial features, and not things that tend to vary greatly such as background and hair. Since the actual face was generally in the same place in the images, we decided to simply crop each to a fixed size of 60x60 around the center of the images.

For the face photo data, we split the entire image into subwindows using a stride of 3. We chose to do a stride of 3 in order to reduce the amount of windows needed, therefore speeding up the algorithm at the relatively low cost of a detection centroid being slightly off.

3.2 Model Training

For our facial detection model, we used the AdaBoost algorithm(Adaptive Boosting). Adaboost is an ensemble learning algorithm used for classification. The main idea behind AdaBoosting is to sequentially train a series of weak classifiers and assign weights on them based on their performance. Once the model has ran its designated amount of rounds, a final strong classifier is created. This strong classifier is a weighted sum of the weak classifiers.

3.2.1 Classifiers

After experimentation with 1000, 2500, 5000, and 7500 initial weak classifiers being passed into the model, we found 5000 weak classifiers returned the highest quality classifications without excessive long training time requirements. With this configuration, we could confidently proceed with further fine-tuning of our system, ensuring that our facial detection component meets both accuracy and efficiency requirements.

3.2.2 Rounds

After experimentation with different numbers of rounds, specifically 15, 25, 28, and 100 rounds, we determined that utilizing 28 rounds resulted in a good balance between accuracy and prevention of over-training in our facial detection model. This choice was made after careful consideration of the error rates associated with each configuration. It's worth noting that over-training became noticeable around the 35th round as the error for every round up until round 100 remained at 0.0. Once this model was ran on the test set the accuracy was roughly 40% lower than the 28-round model. With this as a clear sign of over-training, we reverted to the 28-round model and began working on the other aspects of our facial detection pipeline.

3.3 Skin Detection

For the colored face photos we decided to implement skin detection in order to increase the accuracy of our face detector. We used a histogram based approach for our skin detection algorithm. The idea was that in order for an image to be detected as a face it must be detected by our AdaBoost model as well as have a certain percentage of skin pixels.

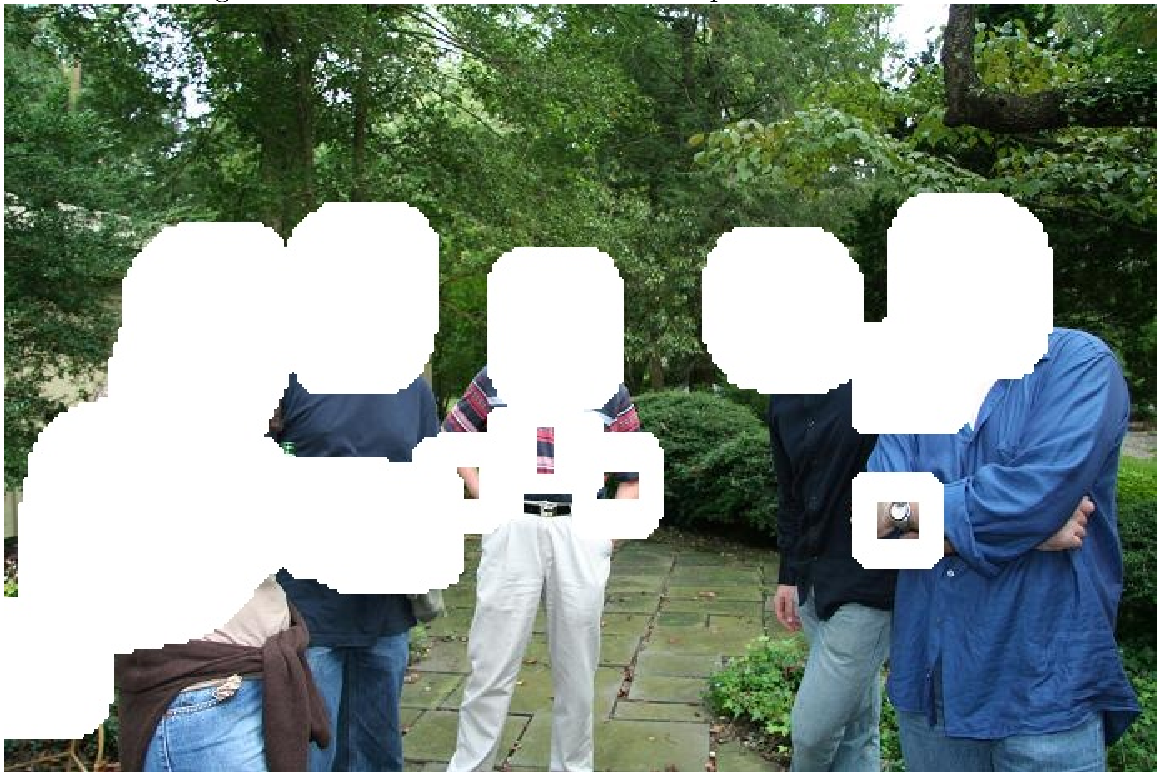
3.3.1 Histograms

We used the positive and negative histograms given in Assignment 5 for our skin detector. They are included in the training folder of the project submission as well.

3.3.2 Implementation

In our implementation skin detection is the first step of the algorithm for the large face photos, after they are divided into subwindows. We first run skin detection on the entire large image, which returns a probability matrix of the likelihood of each pixel in the image being a skin pixel. A probability greater than 0.5 indicates that the pixel is likely a skin pixel. We then compute the total percentage of skin pixels in each window, and exclude all windows below our desired threshold. In our experiments we found that keeping windows containing more than 30% skin pixels worked well. Using higher thresholds tended to exclude too many good windows, and smaller thresholds included many bad windows. Below is a visualization of the windows that were kept. The regions in white are the windows that will be passed on to the next phase of our detector.

Image 1: Visualization of windows that passed skin detection



3.4 Cascading

To enhance time efficiency for our facial detection model, we strategically implemented a cascading approach with three distinct stages in our pipeline. The idea was to have each stage be progressively harder to pass with the use of strong classifiers and AdaBoost score thresholds; allowing only the strongest candidate frames to make it into our final filtration algorithm.

3.4.1 Classifier Count

Each stage was used a different number of strong classifiers for prediction, allowing us to progressively filter and prioritize frames of interest. In the initial stage, we employed 3 strong classifiers to quickly eliminate non-face regions, enabling rapid rejection of irrelevant portions of the image. Subsequent stages, while more computationally intensive, focused on refining the detection process by utilizing a larger number of strong classifiers(Stage 2: 20, Stage 3, 50). In choosing the classifier counts, we focused on speed for the first threshold, a balance for the second threshold, and finally an accuracy focus for the last threshold.

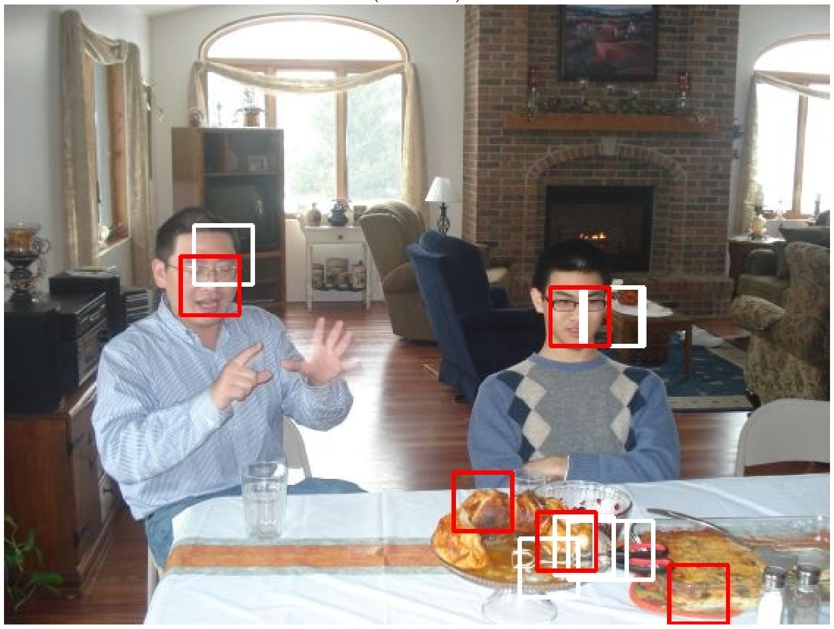
3.4.2 Thresholds

While different strong classifiers were used in each stage to attain a prediction score, it is up to the thresholds to determine which frames continue to the next stage or not. While creating our thresholds we had to take into account that because a lower amount of strong classifiers are used in the earlier stages of our cascade, there must be a certain level of leniency. Taking that into account, the thresholds used progressively increased each stage. Our thresholds were implemented in the following progression: (Stage 1: -3, Stage 2: 0, Stage 3: 3). It was through experimentation and fine tuning that we came to these final thresholds. A starting threshold of -20 allowed almost every frame through the first stage, and an ending threshold of 10 let zero frames through the last. The current [-3, 0, 3] list hurts our accuracy for the cropped gray scale images by about 2%, but we found these values to be worth it when considering the amount of false positives increases rapidly for lower thresholds.

3.5 Detection Filtering

Since multiple windows can contain most or all of a face inside them and be detected by AdaBoost, we needed a way to filter out the overlapping windows. Another challenge to this was that typically there are detection windows that only contain part of the face, precluding random choice. We came up with a way to keep only the windows which have the highest detection score and exclude all other overlapping windows in that area. In the algorithm we find the window with the highest score, keep it, remove all overlapping windows, and repeat until there are no windows remaining.

Image 2: Visualization of detected windows that were kept (red) and dropped (white)



3.6 Multiscale Approach

A challenging aspect of identifying faces in images comes down to variations in scale. Faces can appear at different sizes within an image due to varying distances from the camera or differences in image resolution. Depending on how the model is trained, or how the image is processed before being passed to the model for evaluation, the prediction score can vary greatly.

3.6.1 Implementation

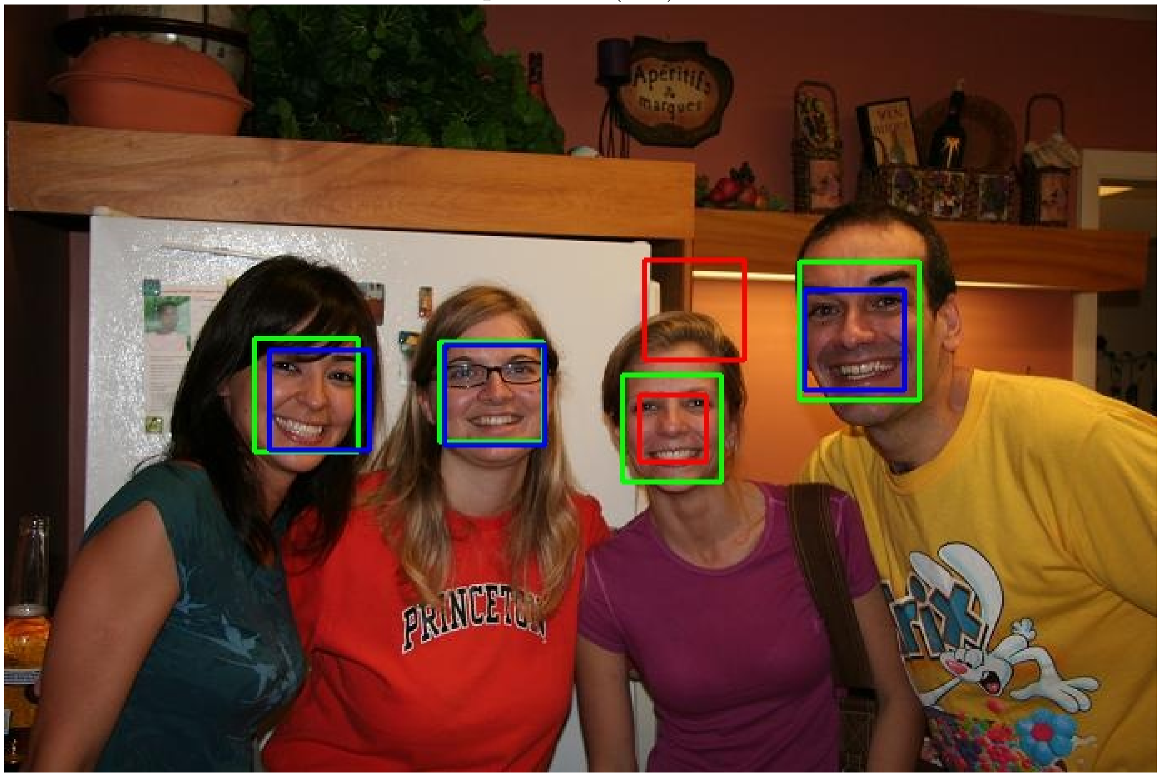
To address the challenge of scale variations in face detection, we implemented a comprehensive approach that combined model adjustments and image processing techniques. Our strategy involved training two distinct AdaBoost models: one specialized for 40x40 pixel images and another tailored for 60x60 pixel images. Each input image underwent evaluation twice, once for each model, yielding candidate frames from both scales. This dual evaluation allowed us to capture facial features across different sizes effectively. We could then use our detection filtering algorithm to attain a final selection of candidates. In retrospect, while it might seem that training a single model and resizing frames for compatibility could offer a more generalized solution, our experimentation revealed otherwise. When attempting to resize much smaller frames to fit our original model, which was trained on 100x100 pixels, we encountered a noticeable loss in accuracy. It became evident that a one-size-fits-all approach wasn't optimal, an our dual model strategy outperformed the resized model in terms of accuracy. This approach not only mitigated the loss of precision but also resulted in a lower amount of false positives.

4 Results

4.1 Computing Accuracy

Coming up with a way to measure the accuracy of model was a challenge. We wanted a detection marked correct if at least 50% of our bounding box overlapped with the correct bounding box. However, the correct bounding boxes were of varying size and shape. We decided to use the Intersection over Union (IoU) of our boxes with the correct boxes as our metric. We considered a detection correct if the IoU was at least 0.5.

Image 3: Visualization of ground truth (green), correct detections (blue), and false positives (red)



In image 4 three of our detections (in blue) had an IoU greater than 0.5, but our bounding box for the third person from left misses some of her mouth and eyes, and is just under the 50% threshold needed for a correct detection.

4.2 Grayscale face/nonface results

These results use the standard accuracy formula since the classes of each image are known.

Test Set	Accuracy
Faces	0.9714
Non faces	0.9987

Table 1: Accuracy table for grayscale face/nonfaces

4.3 Face photo results

Configuration	Accuracy	False positives	Run time (s)
Final model	.4286	54	46.62
No cascade	.4286	54	91.86
No skin detection	.4286	117	140.22
No cascade or skin detection	.42856	117	277.78

Table 2: Results table for face photos

Image 4: Correct detection

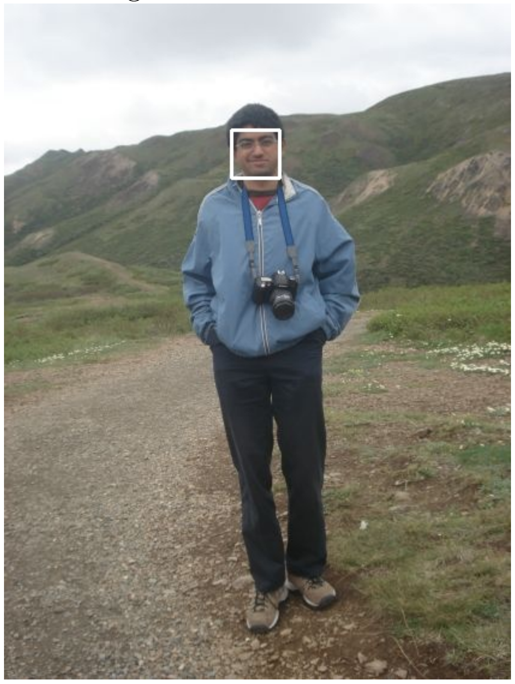


Image 5: Mostly correct detection with false positives

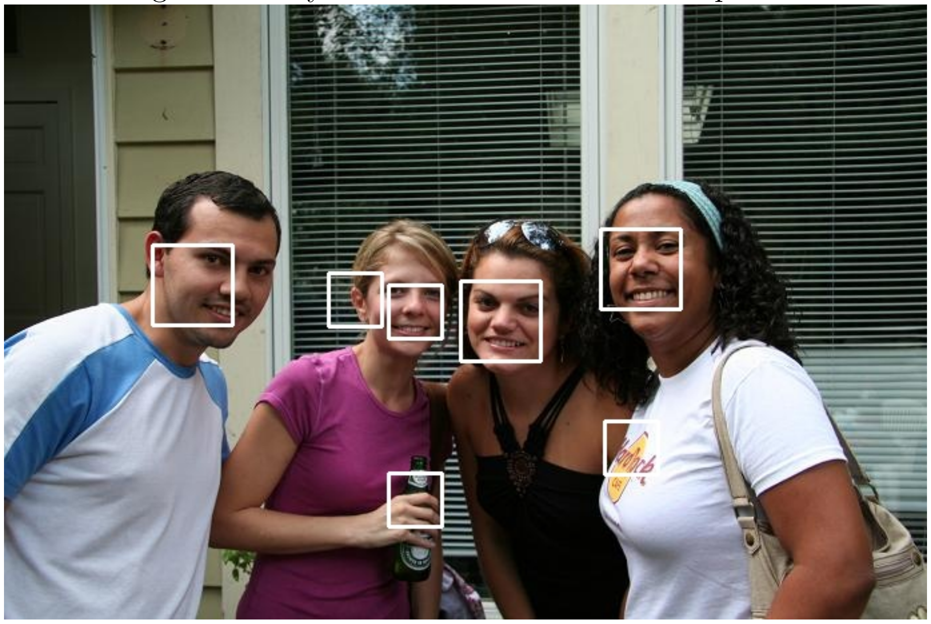
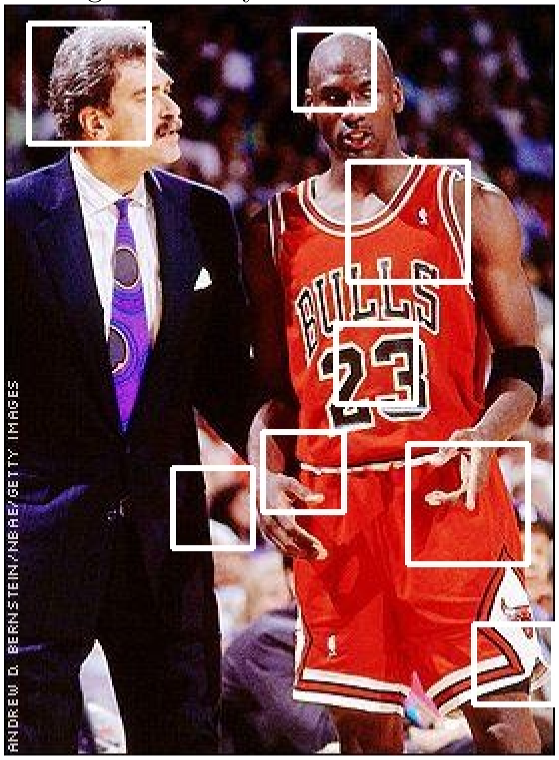


Image 6: Mostly incorrect detection



4.4 Discussion

We had impressive results in the grayscale test sets, with over 97% accuracy for the test faces and over 99.87% accuracy for the test nonfaces. These results may be slightly skewed on the non faces side due to using the same Adaboost score thresholds in the cascade stages for both test sets. As noted above we used a nonzero threshold for the AdaBoost scores in order to combat false positives.

For the face photos we had pretty poor results, only capturing 42% of the correct bounding boxes, with 54 false positives. However, through the use of the classifier cascade and skin detection we were able to speed up the runtime of our model by almost 6x while preserving accuracy and significantly reducing the number of false positives.

While we had lots of false positives, many of them captured at least some of a face, even if they didn't reach the 0.5 IoU threshold needed for correct detection. We had several cases where our bounding box has the proper centroid, but the scale of the box is off, a problem we ran into with our multiscale approach. We noticed in our experiments that the smaller scale model tends to produce higher scores than the larger bounding boxes, even if both scales detect a face in the same area and the larger bounding box would be a better fit.

5 Conclusion

Our model was very accurate at detecting faces at a single scale, but struggles to accurately detect variable scale faces. Even though our nonface accuracy was very high, the sheer number of windows that need to be tested in a given image typically results in many false positives. We are able to remove some of these through the use of skin detection, but ultimately skin detection alone is not enough, and in some cases hurts the model by excluding valid faces. Overall, while the facial detection algorithm we sought out to create may have not met the accuracy goals we would have liked, it employed a impressive amount of speedup. We see our algorithm that balances accuracy and efficiency as a success.

6 Reproducing Visualizations

In our final submission we have turned off visualization since they assume certain directories were created on the host machine. In order to reproduce the visualizations, the four result directories listed in the submission folder must be created, and their paths specified in the relevant visualization sections of our code. They are currently commented out, but the relevant sections can be found by looking for "VISUALIZATION CODE" comments.