CSE 255 Winter 2015 Assignment 1: Eye Detection using Histogram of Oriented Gradients and Adaboost Classifier

Kevan Yuen
Electrical and Computer Engineering Department
University of California, San Diego
kcyuen@eng.ucsd.edu

***Abstract*—An image of an eye is known to have a certain shape, roughly ellipse shaped. And so the Histogram of Oriented Gradient (HOG) feature is a good descriptor for this type of detection. In this paper, the eye detector is trained with HOG features using an Adaptive Boosting (AdaBoost) algorithm to train and learn a set of weak classifiers. It is shown to work fairly well although it underperforms when compared to the detectors included with openCV. However, as will be discussed in this paper, there is a lot of room for improving the eye detector.**

Note to the grader of this paper: The section titles have a "Task" label with numbers corresponding to the assignment1.pdf instructions to help grade each section. I have tried my best to keep everything in their corresponding sections.

I. Introduction

One of the motivations for detecting eyes is to give facial landmark localization algorithms a better initialization than a face detector's box output. Often times, the face detector may not be able to detect a face due to partial occlusions or wide variety of poses. The box output size may also have a wide variation in width and height. In both cases, this will lead to a poor initialization. Individual component detectors of the face (such as eyes, nose, and mouth) is used to get a more accurate initialization than a box representing the face. In this paper, only the eye detector is presented due to a limited time frame for the project.

II. Related Work [Task 1,3]

When it comes to face detection, the Viola and Jones detector [1] is one of the most well known for being robust and fast. The method involves extracting Haar-like features, creating an integral image, and training the features with AdaBoost, and finally using cascaded classifiers where the complexity of the classifier increases as it passes each classifier. Eye detectors have been built around this algorithm and distributed through openCV, which will be used for comparison.

Dalal and Triggs in [2] uses HOG features for human detection. In [3], Zhu and Ramanan performs landmark localization by using the CMU Multi-PIE dataset which contains pictures of seated people in a controlled environment and lighting. The provided annotations in this dataset include camera angle and landmarks on the face. They extract multi-scale HOG features by rescaling the image at fixed intervals and extracting the HOG features to generate a feature pyramid. Using the camera angle annotation, global mixtures are used to capture the different poses the face can be. This provides the training with a variety of lighting conditions as well as poses. Using the learned shape of a face given a mixture model, optimal sets of detected facial landmarks from the HOG features that best match a face shape is detected.

The approach in this paper will use HOG features similar to [3] trained using the AdaBoost algorithm similar to the Viola and Jones detector [1].

III. Dataset [Task 1,3]

Although it would have been extremely advantageous to use the CMU Multi-PIE dataset which has a rich set of lighting and poses, the annotations between different camera angles did not contain the same number of annotations. This meant that some preprocessing of the annotations would have been needed to figure out which landmark belonged to the eyes, which would have been a bit time consuming given the time frame for the project. Instead the Helen dataset [4] is used along with a 68-point facial landmark annotation provided by [5][6]. This dataset, containing 2000 images, is chosen for its wide range of lighting, poses, and expressions, in both indoor and outdoor environments. An example image of this dataset along with annotated ground truth facial landmarks are shown in Figure 1.

Due to the uncontrolled environment of the Helen dataset, it may contain more than one face, however only one of the faces are actually annotated. This means that if this set is used for testing, it may detect eyes on the faces that were not annotated, requiring manual annotation work to label them as true positive detections rather than false positive detections. This is avoided (due to time constraints) by using the Helen dataset and 10% of the FRGC dataset for the validation test set, and the remaining 90% of the FRGC dataset as the actual unseen test set. The Face Recognition Grand Challenge (FRGC) dataset used for this paper contains 4950 images with only one face per image with the same set of annotations provided by [5][6], however the disadvantage is that it is mostly indoors in a controlled environment. Given the limited datasets and annotations availability, the FRGC dataset will be used to evaluate the performance of the eye detector.

The Helen dataset was originally used for landmark localization, while the FRGC dataset was originally used for face recognition. There are also other datasets such as XM2VTS, LFPW, and AFW with their landmarks provided by [5][6]. All of these datasets have most recently been used in an automatic facial landmark detection challenge through ibug [5][6].

As in [1][2][3], negative training examples are also needed. Due to the possibility of faces that are not annotated in the Helen dataset, the background scenery is unfortunately not used as negative examples for training since it would be highly possible to include an eye in the negative training set. Instead the negative examples from the INRIA Person dataset [2] are used, which contains 1219 images of scenes without any people. The INRIA Person dataset has been used in [2][3] as negative training examples.

Although the number of images are small, and appearing to not meet the 50,000 minimum datapoint requirement for this assignment, it can be quickly seen that the training and test set grows quite large. The positive examples of the training set will consist of two eyes from each Helen dataset image, generating a total of 4000 positive training examples. The negative examples of the training set will consist of all possible sliding windows with 75% overlap resulting in 1.15 million negative training examples. In addition to this, as will be discussed later, additional negative examples are included from windows around the faces in the Helen dataset, such as around the eyebrows, nose, mouth, and edges of the face, resulting in 111,264 additional negative training examples.

From looking at these training examples, it is clear the eyes have strong edges forming a specific shape similar to an ellipse, while the negative training examples will a variety of different shapes and edges. Most of the eyes were generally around 80x160 which is the size of the template that will be used. This will be the motivation of the design of our model which will be discussed later in the paper.

In summary, there will be about 4000 positive training images and 1.26 million negative training images extracted from the datasets. A lot of negative examples are required due to the wide range of background scenes that can occur in an image, while positive examples do not need to be quite as large in this case since most eyes look the same, generally speaking. The datasets contain RGB images, however the color channels are ignored and the images are converted to grayscale for simplicity with feature extraction.

IV. Predictive Task [Task 2]

The predictive task for this eye detector is to determine whether a pixel is an eye or not. This will be done by looking at a window of pixels around the current pixel being analyzed, extracting features from that window of pixels, then prediction using the model. The model will be evaluated using the FRGC test set and the annotated landmarks. A spline interpolation is applied to the 6 landmarks around the eyes, forming a very smooth curve surrounding the eye. Any pixels inside this curve will be considered an eye as ground truth, while any pixels outside this curve will be considered a non-eye. True positive rate (TPR) and false positive rate (FPR) given by eq. 1 and 2, respectively, will be used as measures of performance for the model.

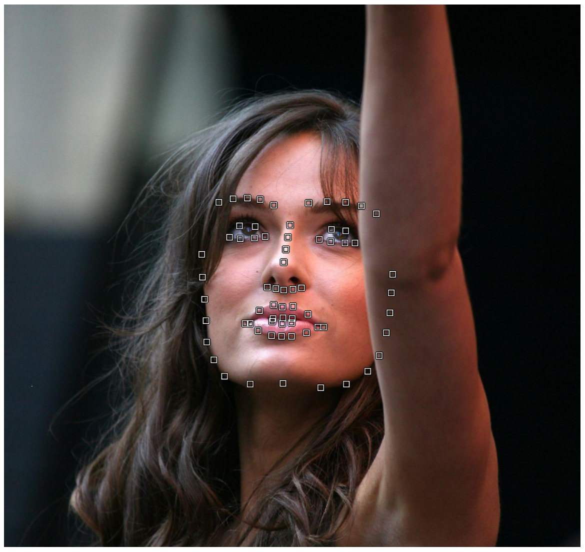
Fig. 1: Example image and ground truth facial landmarks from the Helen dataset

$$TPR = \frac{\text{\# of True Positives}}{\text{\# of True Positives + \# of False Negatives}} \tag{1}$$

$$FPR = \frac{\text{\# of False Positives}}{\text{\# of False Positives + \# of True Negatives}} \tag{2}$$

For comparison, openCV's built in eye detectors ('haarcascade_eye.xml' and 'haarcascade_eye_tree_eyeglasses.xml') with the minimum neighbors parameter set to 0. The reason for setting the minimum neighbors parameter to zero is so that openCV will not prune out detections based on how many neighboring detections there are, since that is a post-processing result of the detections.

The validity of the predictions will be assessed by seeing how well it performs using the TPR and FPR measures as well as looking at some of the output detections showing which pixels were considered an eye. The trained model parameters will also be analyzed later in this paper to see if it makes sense.

V. Features [Task 4]

A. Dataset Preprocessing

Since it was found in the Helen dataset that most eyes were generally 80x160 on average, positive training examples are extracted from the image by de-rotating the face so that the center of the two eyes (each determined by the 6 annotated landmarks) form a horizontal line. Then a window of pixels, centered on each eye, with the width calculated based on the two eye corner landmarks and the height being simply half the width, is extracted. This window of pixel is then resized to 80x160, which will minimize the distortion of the original image, is used as a positive training example. An example of this can be seen in Figure 3.

Negative training examples are extracted by placing windows (the same size used for the eyes) around each of the other non-eye landmarks, and resizing to 80x160. An example of this can be seen in Figure 2. Additional negative training examples are also extracted from the INRIA Person dataset by sliding a 80x160 window across the image with 75% overlap. Each window is extracted as a negative training example.

B. HOG Feature Extraction

As discussed before, the eyes have strong edges close to the form of an ellipse which will be the main motivation for using the HOG feature, which describes the strength of an edge as well as it's orientation. There has also been success with using HOG features for facial landmark detection in [2].

Given an image I of size 80x160, the HOG feature is extracted by taking the gradient of the image along the horizontal and vertical direction, resulting in two images (I_x, I_y). The gradient kernel used was $[-1, 0, 1]$ for simplicity purposes. The pixel (u, v) in the magnitude image (I_M) is then calculated by eq. 3.

$$I_M(u,v) = \sqrt{[I_x(u,v)]^2 + [I_y(u,v)]^2} \tag{3}$$

And the pixel (u,v) in the angle image (I_A) is calculated by eq. 4.

$$I_A(u,v) = tan^{-1}\left(\frac{I_y(u,v)}{I_x(u,v)}\right) \tag{4}$$

The image is then divided into 5x10 cells, e.g. each cell will be a 16x16 image, and a normalized weighted histogram of the angles is generated with 16 bins for each cell. The weighted part means that instead of adding 1 for each pixel in the cell that fits in the bin's angle range, the pixels magnitude is added instead. The reasoning behind this is that if a pixel had a weak magnitude (e.g. a weak edge), then it will not contribute as much to that bin angle compared to a pixel that had a strong edge. Each cell's histogram is then normalized such that the histogram will sum to 1. The resulting feature vector for this 80x160 image is 5x10x16, or 800. An extra feature with a value of 1 is added as a parameter for the AdaBoost training algorithm to learn, resulting in a feature vector length of 801. A visualization of the HOG feature is shown in Figure 3. Notice that there are 5x10 HOG visualizations per eye and 16 lines per HOG visualization. It is not immediately apparent but the cells near the edge of the eyes have darker lines (indicating higher bin count for that angle) along the direction of the edge of the eye.

VI. Model [Task 5]

The AdaBoost algorithm uses a weak classifier to find an optimal threshold in one of the data dimensions into positive and negative. Then after it has found the dimension and threshold, the weak classifier is called again iteratively, adjusting more weight to the misclassified examples as it continues. The end result is a strong classifier made up of a cascade of weak classifiers represented by eq. 5.

$$\text{score} = \sum_{i=1}^{d} \alpha_i f_i' \tag{5}$$

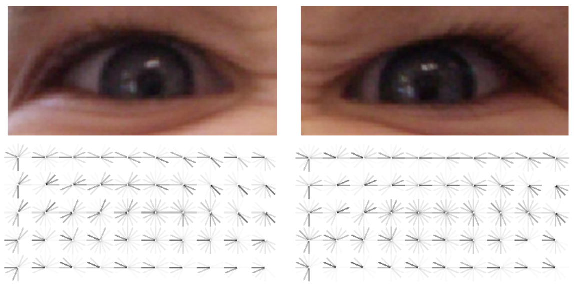

Fig. 3: Example of positive training samples and visualization of their corresponding HOG features

where d is the dimension of the feature vector f and

$$f_i' = \begin{cases} 1, & \text{if dir} = 1 \ \& \ f_i \geq \tau_i \\ -1, & \text{if dir} = 1 \ \& \ f_i < \tau_i \\ -1, & \text{if dir} = -1 \ \& \ f_i \geq \tau_i \\ 1, & \text{if dir} = -1 \ \& \ f_i < \tau_i \end{cases} \tag{6}$$

where 'dir' basically specifies which binary class the i^{th} feature, f_i, is likely to belong to if it is greater than or equal to the learned threshold τ_i.

Generally, if the 'score' is positive, then it is classified as a detection, otherwise it is classified as a no detection. However for exploratory purposes, the score will be thresholded at a range of values to generate an ROC (Receiver Operating Characteristic) curve using the TPR and FPR formulas given by eq. 1 and 2. Depending on the application, a user may want to have more detections at the cost of increasing the false positive rate. This would be true for my future work where the false positives can be pruned out using neighboring detections of the same kind (e.g. eye) by checking how dense the detections are in that area (more likely to be an eye) or by using the results of other detections (e.g. nose and mouth) by checking if there's a nose and mouth beneath the eyes in a learned face shape model.

The main reason for using AdaBoost is for its success in the well known Viola and Jones face detector [1] has trained using AdaBoost [7]. Using HOG features, the AdaBoost will select the optimal features (e.g. which cell in the image that was split into 5x10 parts, and which angle the edge should be facing), find an optimal threshold to decide which class that feature contributes towards (e.g. how strong the selected edge was), and a weight to decide how much contribution that feature should have towards classification.

Originally the model was designed using only the INRIA Person dataset for the negative examples, however the resulting model ended up having lots of false positives in other parts of the face, especially the mouth since it has a similar shape. The model was improved by including negative examples from other parts of the face, resulting in a lot fewer false positives in other areas of the face using the FRGC validation test set.

The next attempt to improve the model was by including rotated versions of the positive and negative training examples from the face. This led to an extremely longer training time

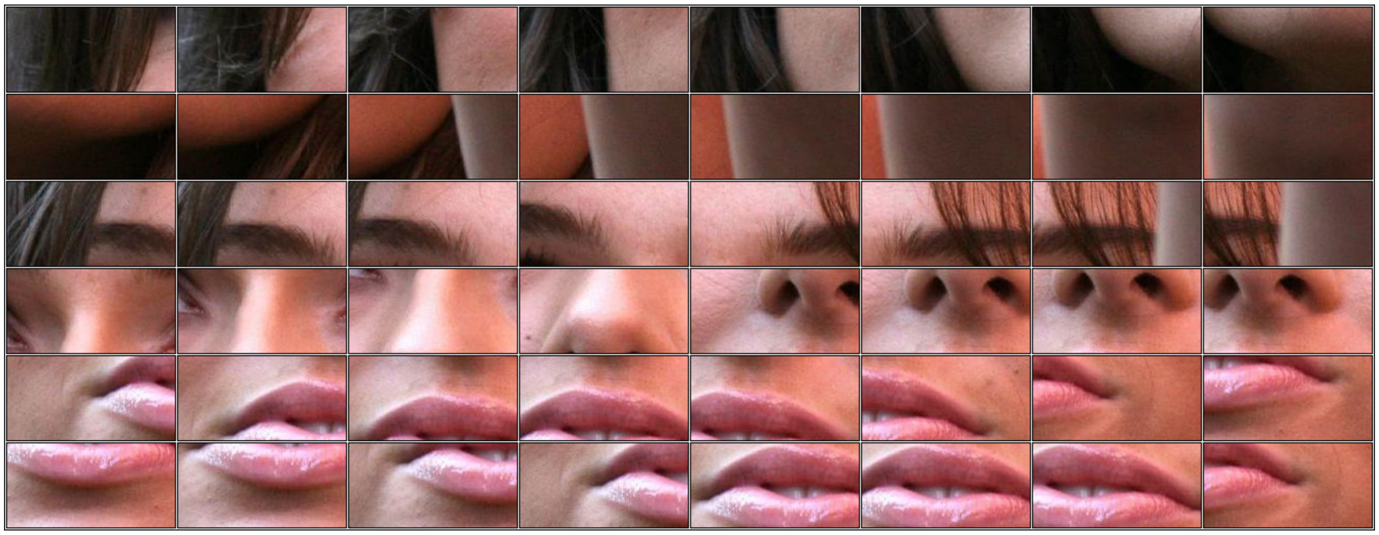
Fig. 2: Example of negative training samples extracted from a face

of roughly 17 hours, and a lot more false positive detections (again through FRGC validation test set). The reasoning for this is likely due to the difficulty of finding good thresholds (e.g. boundaries between the two classes) now that rotated versions of the eyes have been added. The rotated versions would likely need to be assigned to different classes for this method to work along with an extension of AdaBoost to multi-class.

Another possible attempt to improve the detector would have been to increase the bin width of the histograms, e.g. changing the number of bins from 16 to 8, in the case that the bin widths were too narrow to capture the features properly. Or to change the cell division of the images into either smaller or larger boxes within the 80x160 image. Due to the long training times, I did not have enough time to try more methods.

One issue that I ran into was when transitioning from a computer with 64 GB of ram to only 16 GB of ram during classification on the test set. The issue was running out of memory due to my method of classifying all the pixels in an image all at once, however this problem was solved by changing the way the features are stored from type double to type uint8. Since the features stored ranged from 0 to 1 and having all the decimal precision was not necessary at all, the features were multiplied by 255 then rounded off into uint8 type. The model's thresholds were also multiplied by 255 accordingly, there was little to no change in performance as expected since very precise decimal values were not required.

An SVM classifier was also tested as an alternative model for its popularity in machine learning and robustness in finding good boundaries between classes to minimize misclassification rates. The SVM classifier would fit just as well as the AdaBoost algorithm for the same reasons with finding good boundaries in the different feature dimensions. Several values of the parameter 'C' were trained on a subset of the training set and tested on using the FRGC validation test set. The AdaBoost classifier does not have any parameters to optimize with aside from the input features. However a threshold can be chosen on the AdaBoost score as already discussed, depending on the application. For comparison with SVM, the threshold was set to zero for the AdaBoost, also trained on the same subset of the training dataset and tested on the FRGC validation test set. The advantage of the SVM classifier is that it is faster to train, however the AdaBoost classifier was found to have performed better than the SVM classifier when applied to the validation test set. Unfortunately training with AdaBoost does take a couple of hours each time. Because of this, there was not have enough time to train and optimize both classifiers using the entire training set. The AdaBoost classifier was chosen for its better performance through fewer false positives and higher true positives on the FRGC validation test set, with both classifiers trained on the smaller training subset. It is not apparent why the SVM classifier did not perform as well as the AdaBoost algorithm, it could be due to the AdaBoost's boosting portion that helps it perform better although at the cost of longer training times.

VII. Results [Task 6]

After training the AdaBoost classifier on the entire training set with 100 iterations (e.g. 100 α_i values will be generated or 100 weak classifiers), the classifier is tested on the FRGC test set. This is done by essentially sliding a 80x160 window across the image, extracting the HOG features as it goes along then classifies the pixel, outputting an AdaBoost classifier score, resulting in an image of scores. This is repeated for all images in the test set. A range of thresholds (used to generate varying TPR vs FPR for the ROC curve) are set on the score to determine whether it was an eye pixel or a non-eye pixel. False positives are the sum of all pixels that are not in the eye but were classified as an eye. True positives, however, are defined a little differently due to the nature of the detectors. If there is even a single pixel classified as an eye within the eye, then it is considered one true positive, regardless of how many other pixels were classified as eye or non-eye within that eye. Since there is only one face per image and thus two eyes, there is at most two True Positives per image. The resulting ROC curve is shown in Figure 4, along with the TPR and FPR of the openCV eye detectors. Note that the x-axis has very small values, but given a 1000x1000 image, and a false positive rate of 0.01%, that is 200 misclassified pixels.

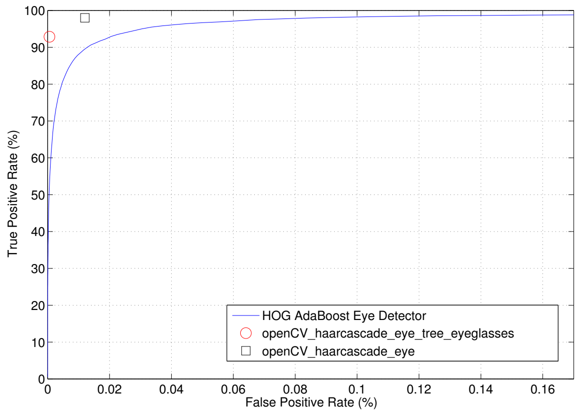

Fig. 4: The Receiver Operating Characteristic Curve showing the TPR vs FPR of the HOG AdaBoost Eye Detector on the FRGC test set. The openCV eye detectors are also shown as single points with a fixed TPR and FPR due to a lack of input arguments that can adjust the TPR vs FPR similar to the AdaBoost score.

Both openCV detectors still perform considerably better than the HOG AdaBoost Eye Detector. One immediate difference between the two detectors is that the openCV eye detectors are able to detect at different sizes, whereas the eye detector in this paper was not designed for multi-scale due to time constraints. In order to implement multi-scale, a feature pyramid would need to be built as done in [3]. This is basically done by rescaling the images at fixed intervals and extracting the features, then checking if any positive detections were found at any of the scaled resolutions, which is usually required due the eye detector being trained only on 80x160 windows but in reality the eyes could actually vary in size a lot.

The AdaBoost Model parameters that classifies the image as an eye are visualized in Figure 5. The visualization is the same as 3. These are the main oriented edges that contribute towards classifying it as an eye. The darker edges have a stronger contribution or weight. The model basically checks if these features are present and meet the threshold (which is not visualized here). Notice that the edges form an ellipse where the eyes should be, so the parameters actually make a lot of sense, although it does have some parameters that don't really make sense such that those on the upper left corner.

The parameters that classify the image as a non-eye are visualized in Figure 6. Some of the features here make sense as some of the angles appear to be perpendicular to the direction the edges of the eye should generally be in those areas. Notice here most of these edges though appear to be random and don't really form a specific recognizable shape. This is to be expected since they are some parameters learned from the shape of the eye as well as a lot of random negative images of background scenery and other parts of the face. AdaBoost has found learned those features to consider those as non-eyes if those are present given the training set.

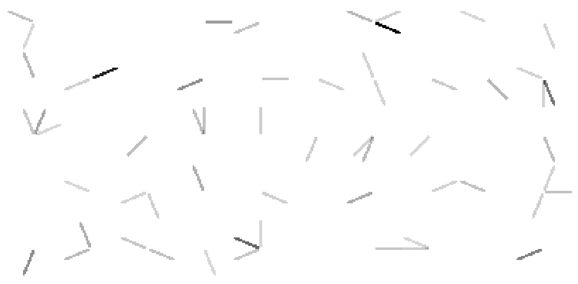

Fig. 5: AdaBoost Model's positive detection parameters visualization

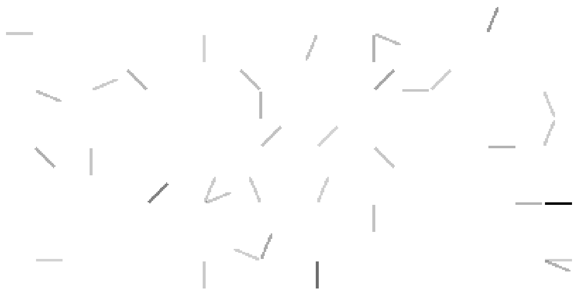

Fig. 6: AdaBoost Model's negative detection parameters visualization

VIII. Conclusion [Task 6]

Although the HOG AdaBoost Eye Detector didn't outperform the openCV and instead underperformed, there is still motivation to continue work. An easy way to improve performance is by training for more iterations. Due to time constraints, the AdaBoost was trained on 100 iterations, thus it is composed of only 100 weak classifiers. If the maximum number of iterations was set to a larger value or to run until "convergence", it could perform better assuming it will not end up overfitting which has not been seen yet based on the results of the validation test set and the actual test set.

The model parameters made sense, but not perfectly. More time being spent on aligning the positive training images as well as picking a more optimal cell division of the images would improve performance. It was noticed later that sometimes the eyes are not aligned since some eyes open "taller" than other eyes. Sometimes the eyes are closed as well. It is possible for the closed eyes, those may need to be in a separate class using the feature extraction discussed in this paper. This would ideally also result in a parameter visualization that makes more sense.

It was found that at least one image in the INRIA Person negative training set contained an image of an eye. So From this, I would say it is fairly robust to mislabeled examples in the data.

Another issue is that sometimes the eyes are not around a dimension of 80x160, and so the features may not be represent the eyes very well because of this. As mentioned before, a feature pyramid with different levels corresponding to different scales of the image can solve this problem. There is also the problem of eyes being rotated, one naive method is to also create a feature pyramid with different levels corresponding to different rotations of the image, however I would experiment with more efficient methods such as using circular cells rather than square cells so that the features are more rotationally invariant in order to reduce the number of HOG feature extractions required.

The false positive rate is the largest problem keeping the HOG AdaBoost Eye Detector from performing as well as the openCV eye detectors. This can be seen in Figures 7 and 8. This can be remedied through adding more negative training examples by taking the misclassified pixels (of the training set) and adding the window around that pixel as a negative training example so that AdaBoost can relearn the parameters and improve. Notice that some of the other parts of the face and hair was considered an eye detection. This may be due to lack of negative training examples in those areas since the only negative face examples came from where the landmarks were annotated, which did not include the hair, forehead or neck. It is a difficult problem for the eye detector, however, to differentiate between an eye and a mouth sometimes. Perhaps other types of features will need to be examined or this is left for the next stage of false positive pruning through the use of eyes, nose, and mouth component shape matching.

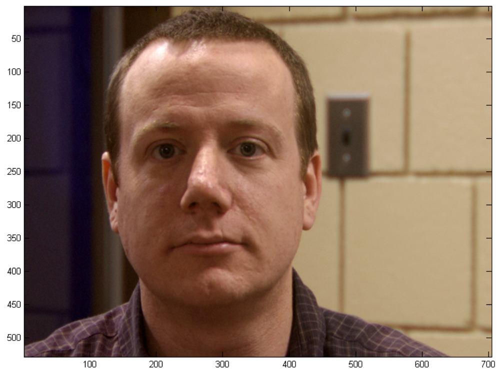

Fig. 7: Example image from FRGC test set

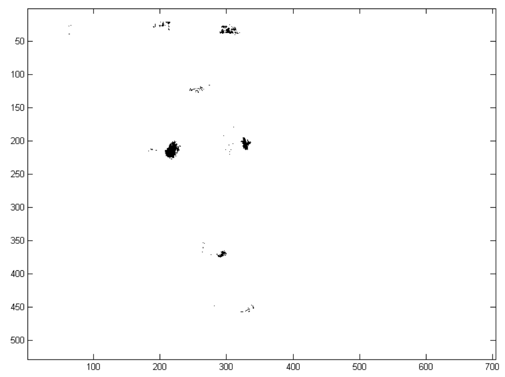

Fig. 8: Classified image from the FRGC test set. Black indicates pixel is an eye, white indicates non-eye.

Acknowledgment

References

[1] P. Viola and M. J. Jones, "Robust real-time face detection," *International journal of computer vision*, vol. 57, no. 2, pp. 137–154, 2004.

[2] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 1. IEEE, 2005, pp. 886–893.

[3] X. Zhu and D. Ramanan, "Face detection, pose estimation, and landmark localization in the wild," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, 2012, pp. 2879–2886.

[4] V. Le, J. Brandt, Z. Lin, L. Bourdev, and T. S. Huang, "Interactive facial feature localization," in *Computer Vision–ECCV 2012*. Springer, 2012, pp. 679–692.

[5] C. Sagonas, G. Tzimiropoulos, S. Zafeiriou, and M. Pantic, "A semi-automatic methodology for facial landmark annotation," in *Computer Vision and Pattern Recognition Workshops (CVPRW), 2013 IEEE Conference on*. IEEE, 2013, pp. 896–903.

[6] ——, "300 faces in-the-wild challenge: The first facial landmark localization challenge," in *Computer Vision Workshops (ICCVW), 2013 IEEE International Conference on*. IEEE, 2013, pp. 397–403.

[7] Y. Freund, R. Schapire, and N. Abe, "A short introduction to boosting," *Journal-Japanese Society For Artificial Intelligence*, vol. 14, no. 771-780, p. 1612, 1999.