Implementation of Classroom Attendance through Facial Detection and Recognition

Sahit Bollineni
University of Michigan
sahitb@umich.edu

Juan Orozco
University of Michigan
jcorozco@umich.edu

1. Introduction

In recent years, in order to improve the quality of education and ensure that students are obtaining an optimal educational experience, some instructors have made attendance mandatory. Currently, this has been enforced through the use of some sort of sign-in sheet or clickers that record student responses. Unfortunately, in many instances, this has led to a rise in academic dis-integrity and the policies have not resulted in the intended effect. In order to solve this issue, this project aims to implement an innovative approach to record classroom attendance through the use of computer vision techniques. This will reduce the time spent by both instructors and students in recording attendance and can serve as a platform for advanced analysis of students' behavior during class.

2. Approach

In order to determine classroom attendance using computer vision techniques, an image of the students present in the class is taken from the front of the classroom, as seen in Figure 1.

Figure 1. Classroom Image

Using this image, face detection and face recognition are performed. Face detection is used to determine the location of the faces in the classroom image and extract sub images of all the faces found. Then, face recognition is used to determine the identity of the students present by comparing the faces detected with the face images from the students enrolled in the class. Finally, using the identities of the students present, attendance is recorded. This process is briefly illustrated in Figure 2.

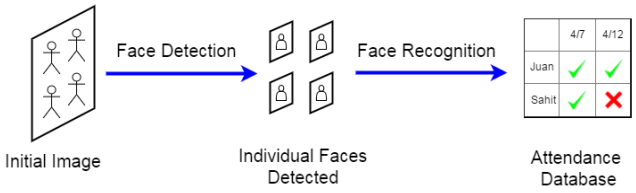

Figure 2. High-Level System Architecture

For this project, various face detection and face recognition algorithms were investigated and a total of two methods for each task were implemented from scratch. For face detection, a skin-color blob detection algorithm [1] and the Viola-Jones algorithm [2,3] were implemented. For face recognition, eigenfaces and neural networks were implemented. We implemented these different methodologies to understand different forms of face detection and face recognition algorithms and to investigate how higher accuracy and robustness are achieved. These algorithms were implemented using MATLAB R2015a and they are discussed in the following sections.

2.1 Challenges

The known challenges of face detection and recognition include: (1) Pose variation, which is present when the subject's face is angled away from the direction of the camera; (2) Feature occlusion, where the subject's face is partially covered by another object; (3) Ambient conditions such as poor lighting or low image resolution.

2.2 Face Detection

2.2.1 Skin-Color Blob Detector

This method takes a novel approach to the problem of face detection using a skin-color blob detection algorithm. This approach is very resistant to the subject's pose changes and will function properly even when the subject may not be facing the camera directly. This is very beneficial in a classroom environment as the student may not be facing the front of the classroom at all times. The initial image is filtered to display regions where color resembling that of the human skin is found. To do this, the image is first converted into the YC_bC_r color space in an effort to diminish the effect of the lighting conditions of the image. After determining the skin-colored regions present in the

image, morphological operations are performed to reduce the effect of noise within the image as well as amplify regions where skin-color regions are detected. A blob detection filter is then applied to determine the possible areas where a student's face may be present. In an attempt to reduce incorrectly detected regions, the potential regions are then operated on again with a finer skin color filter and then checked to see whether the region contains a face-shape, at which point, the potential zone is determined to contain a student's face.

2.2.1.1 Modifying the Color Space

Initially, the input image is given in the standard RGB color space. Each of the RGB values at a certain pixel is greatly affected by the lighting conditions of the environment at which the image was captured [1]. Thus, the image was converted into the YC_bC_r color space using the equation shown in Section 3.1.1.1. Figure 3 below shows an example of an image converted from RGB to the YC_bC_r color space:

Figure 3. RGB to YC_bC_r Color Space

Within the YC_bC_r color spectrum, the Y component refers to the luminance, or light intensity, at that pixel, the C_b component refers to the blue-difference at that pixel, and the C_r component refers to the red-difference at that pixel. Now, by disregarding the Y component, we can reduce the effects of the lighting conditions and focus on the C_b and C_r values to perform the actual skin-color filtering. To determine the range of values of C_b and C_r to be denoted as skin color, some calibration of the threshold values must be done using known face data as can be seen by the experiments outlined in Section 4.1.1.1.

2.2.1.2 Binary Image Processing

After the image has been converted into the YC_bC_r color spectrum and filtered using the calibrated values for skin colors, the resulting image can then be improved through the use of the binary morphological operations of erosion and dilation. Erosion is used to eliminate very small background noise. Dilation is then used to amplify the resulting the pixels detected and fill in holes. Figure 4 below shows the skin-color filtered image as well as the results of the erosion and dilation operations:

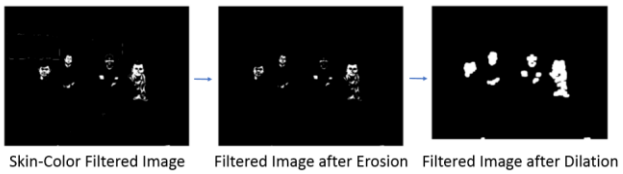

Figure 4. Binary Image Processing

2.2.1.3 Blob Detection

Now that the image has been filtered into an image with blobs where skin-color pixels are found, the image can then be passed through a blob detector. The blob detector detects blobs of multiple scales in the image using a Laplacian of Gaussian filter. The filter itself is scaled to sizes that can be expected of faces in the image and cascaded through the entire image. Non-maxima are suppressed to only show blobs that fit the best. Figure 5 shows the image after being passed through the blob detector:

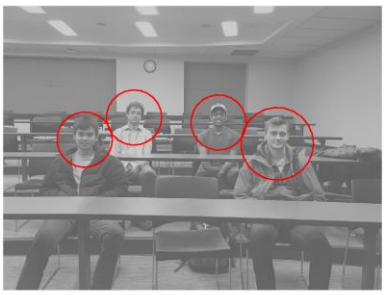

Figure 5. Image after Blob Detection

2.2.1.4 Face Verification

The regions indicated by the blob detector are once again extracted from the initial image. Then, another skin-color filter is applied with finer bounds. This result is once again passed through the blob detector to verify that the image is actually a face and not background noise or a skin region that is not part of a face. If the face recognition scheme used is able to also detect whether the image is a face or not accurately, this verification process is not required.

2.2.2 Viola-Jones

This algorithm, as introduced by Paul Viola and Michael Jones in 2001, is able to accomplish rapid object detection using a boosted cascade of simple features [2,3]. It is able to do so by (1) using a new representation of an image called the integral image, which allows for quick evaluation of simple features, (2) implementing a machine learning algorithm based on Adaboost to build efficient classifiers from a small number of critical visual features, and (3) by employing a cascade method that uses simple classifiers to reject most of the true negatives in the first stages and then uses more complex classifiers in the later stages to detect faces in an image [2,3]. These are described in more detail next.

2.2.2.1 Simple Features

The five types of simple Haar-like features that are used by the Viola-Jones algorithm are shown in Figure 6. The width and height of these features are scaled up until the feature no longer fits in a 24-by-24 detection window. For each scale, the feature is placed in all possible locations in the filter, which leads to having a total of 162,336. When calculating the feature, the pixels in the image under the shaded regions are added while the pixels in the white regions are subtracted.

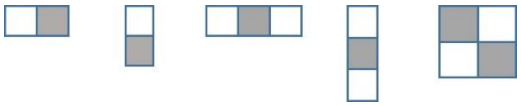

Figure 6. Five Types of Simple Features

2.2.2.2 Integral Image

In order to increase the speed of computation of the simple features, an integral image is used. As described in [2,3], the integral image at location (x, y) is given by the sum of the original image pixels above and to the left of (x, y). This makes computing the simple features very efficient as the sum of the pixels within a specific area is given by a simple computation that involves four elements in the integral image. For example, if the sum of the pixels in the shaded area in Figure 7 needs to be computed, this can be done by simply calculating (D-C-B-A) in the integral image. Please refer to [2,3] for more details of this calculation.

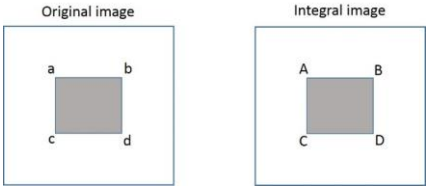

Figure 7. Five Types of Simple Features

Therefore, for the simple features shown in Figure 6, since they have adjacent sides, they can be calculated with six, eight, and nine array references of the integral image. For example, for the feature with three adjacent rectangles, as shown in Figure 8, the feature is equal to 2B + D + E + G - A - 2C - 2F - H.

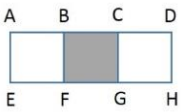

Figure 8. Three Rectangle Feature

2.2.2.3 AdaBoost

Viola-Jones uses a modification of the machine learning algorithm AdaBoost to select small number of features from the 162,336 and combines them into a very effective classifier. In machine learning terms, the single features that are selected are called weak classifiers. They are thresholded features that don't lead to great performance results and are only expected to classify 50% of the images correctly. [2,3] On the other side, the very effective classifier is called a strong classifier and it is a weighted combination of the weak classifiers with a threshold.

2.2.2.4 Attentional Cascade

Attentional cascade, as described in [2,3], is an algorithm that achieves increased detection performance while drastically reducing computational time. The main idea behind this method is that simple, small, and fast classifiers can be used at the beginning of the detection process to reject the majority of the non-face images. Then, for the images that survive this classification, more complex classifiers can be used to identify the real face images. Therefore, very little time is spent on the images that are definitely not faces and more time is spent on the images that are more difficult to classify. This process is broken up into stages where each stage is a strong classifier, and only the positively-classified images are passed as the input to the next stage, while the others are rejected. Figure 9 illustrates this process. The number of stages is determined by the desired final detection and false positive rates.

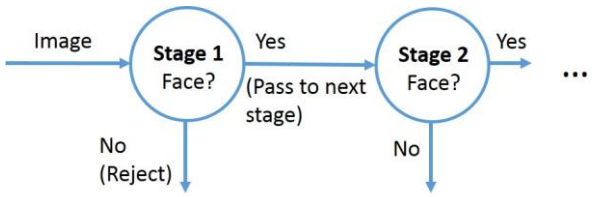

Figure 9: Attentional Cascade

2.3 Face Recognition

2.3.1 Eigenfaces

In this algorithm, face images are projected onto a feature space, called the face space, which spans the variations among a set of known face images [4]. Eigenfaces, which are the eigenvectors of the principal components of the face images, represent significant features that are not necessarily commonly known face features, such as eyes, noses, and mouths. Eigenfaces are used to characterize individual faces and can be used for both detection and recognition. The characterization of an individual face is achieved by projecting the image onto the set of eigenfaces, which yields the weighted sum of the eigenfaces that resembles the individual face the most. Recognition is achieved by comparing this weighted sum with the weighted sum of known individuals.

2.3.2 Neural Network

The methodology proposed for this algorithm has two stages of separate Neural Networks. The first stage is used to determine whether the image in question is a face or not. The second stage is then used to identify the face. The first stage uses a Convolutional Neural Network (CNN). A CNN is a type of feed-forward neural network used in deep learning which utilizes spatial locality of data for classification. All human faces share features which can be taken advantage of to classify whether the image is a face. The first stage of our approach utilizes a CNN trained using mini-batch gradient descent. The architecture of the first stage of this methodology can be seen in Figure 10 below.

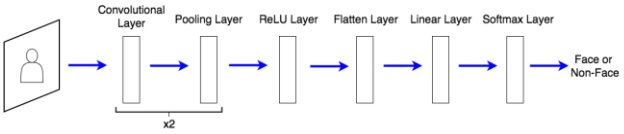

Figure 10. CNN Architecture to determine Face or Non-Face

The second stage is an Artificial Neural Network (ANN) used to determine the identity of the face in the image. The neural network is used to use the pixel data to classify the image and determine the person the face belongs to. The architecture of the second stage of this methodology can be seen in Figure 11 below.

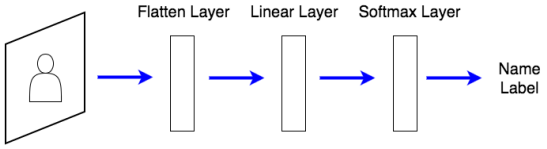

Figure 11. NN used for Face Identification

3. Implementation

3.1 Face Detection

3.1.1 Skin-Color Blob Detector

The Skin-Color Blob Detection algorithm was built from scratch apart from the blob detector itself which was built on top of the homework 3 code of EECS 442 [1]. The algorithm performs the following steps in order to determine the locations of the faces present in the image:

Filter the image to keep only skin-color regions:

1. Convert the image from the RGB spectrum to the YC_bC_r spectrum as follows [1]:

$$\begin{bmatrix} Y \\ C_b \\ C_r \end{bmatrix} = \begin{bmatrix} .257 & .504 & .098 \\ -.148 & -.291 & .439 \\ .439 & -.368 & -.071 \end{bmatrix} \cdot \begin{bmatrix} R \\ G \\ B \end{bmatrix} + \begin{bmatrix} 16 \\ 128 \\ 128 \end{bmatrix}$$

2. Use the bounds of C_b and C_r that denote skin color to detect regions that skin appears. The thresholds used are $150 < C_b < 185$ and $145 < C_r < 165$.
3. Apply morphological operation of erosion using a disk filter of small size, radius ~2.
4. Apply morphological operation of dilution using a disk filter of larger size, radius ~10.
5. Perform blob detection by cascading a Laplacian of Gaussian filter across the entire image. The filter is scaled to the size of faces expected in the image.
6. Extract the regions where blobs were detected.
7. Filter skin-color segments on the extracted potential face regions once again using more refined bounds of C_b and C_r.
8. Perform blob detection once again.
9. Extract the images that were confirmed as faces.

3.1.2 Viola Jones Algorithm

We implemented the Viola Jones algorithm from scratch which accomplishes the following:

Perform AdaBoost to construct strong classifier:

1. Read faces and non-face images in the training set.
2. Evaluate features on all images and determine threshold and polarity to perform classification.
3. Calculate classification error for all features and determine the feature that produces the minimum error.
4. Add feature to strong classifier.
5. Repeat 1-4 for a specified number of weak classifiers, changing the weights assigned to each sample after every iteration.

Build Attentional Cascade:

1. Based on desired detection and false positive rates for each stage, construct a strong classifier by adding one weak classifier at a time and evaluating its performance on the training set until desired rates are achieved.
2. Add stages to the attentional cascade until desired overall detection and false positive rates are achieved, or the number of specified stages are computed.

Evaluate strong classifier in test image:

1. Apply a strong classifier to sub-windows of a test image to determine whether they are faces or not.
2. Draw boxes on top of the sub-windows that are classified as faces.

3.2 Face Recognition

3.2.1 Eigenfaces

The eigenfaces algorithm that we implemented from scratch does the following:

1. Perform principal component analysis (PCA) on the training set of images to determine eigenfaces.
2. For each subject in the training dataset, determine its corresponding components in the face space by averaging the components of all the face images of this subject.
3. For each new face image that needs to be recognized, project the image into the face space and finds its Mahalanobis distance to the components that characterize the known subjects.
4. For the smallest distance found, if it is below a tuned threshold, recognize the new face image as the subject that yielded such distance.

3.2.2 Neural Network

The two stages of our Neural Network system are optimized version for face recognition purposes built on the framework written for homework 5 of EECS 442. Initially, the training data and the testing data from the University of Washington database [5] as well as the data from the AT&T Laboratories Cambridge [6] are pre-processed. In addition, weight decay was implemented to make the training of the CNN and ANN faster. The process for this is as follows:

Determine whether the image is a face or not:

1. Pre-process training and testing images from the University Washington database [5] by normalizing and subsampling down to 24x24 image.
2. Initialize stage 1 CNN to contain the following layers
    - Conv.: 5 Filters of Size 2, Depth 1
    - Pooling: Filter of Size 2, Stride 2
    - Conv.: 5 Filters of Size 2, Depth 5
    - Pooling: Filter of Size 2, Stride 2
    - ReLU
    - Flatten
    - Linear: 125 Inputs, 2 Outputs
    - Softmax
3. Train model using Mini-Batch Gradient Descent with weight decay
4. Test model every 100 iterations

Identify the person in the image

1. Pre-process training and testing images from the AT&T face database [6] by normalizing and subsampling down to 24x24 image.
2. Initialize stage 2 ANN to contain the following layers
    - Flatten
    - Linear: 10304 Inputs, 40 Outputs
    - Softmax
3. Train model using Mini-Batch Gradient Descent with weight decay
4. Test model every 100 iterations

4. Experiments

4.1 Face Detection

4.1.1 Skin-Color Blob Detector

In order to detect the bounds for which skin-color should be detected, there is some calibration involved due to the differences in lighting for each classroom. It's worth noting that while changing the color spectrum from RGB to YC_bC_r certainly allow for some filtering of the luminescence in the environment, the lighting conditions may still have an effect on the skin-filtering and it is necessary to make sure that the bounds specified are applicable to the classroom in question. It is also worth noting that the range of skin colors across the world varies greatly. In order to include skin colors of all kinds, the bounds of C_b and C_r are calibrated with four subjects with vastly different skin tones.

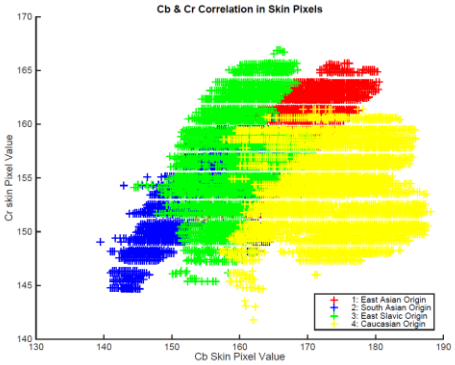

Figure 12: C_b and C_r Correlation in Skin Pixels

From Figure 12, we can see that the range of values lie between 140 and 190 for C_b and 145 and 165 for C_r. In order to keep the background regions from also being included in the filter, the range can be constrained to be $150 < C_b < 185$ and $145 < C_r < 165$ which was able to produce the Skin-Color Filtered Image as shown in Figure 4. After producing the Skin-Color Filtered Image and executing the binary morphological operations, the image was sent through the blob detector. The results of the overall algorithm can be seen in Figure 13 below.

Figure 13: Results of Skin-Color Blob Detector

Through the testing of various versions of the classroom image shown in Figure 13, the results are excellent and quire resilient to pose change of the students, camera placement, as well as feature occlusion due to other body parts. While the test data for students within a classroom is somewhat limited, the results are promising and confirm that this is a feasible methodology for determining the location of faces through the use of skin-color. The fallbacks of this method lie in the fact that other body parts and regions that resemble the circular shape of a face may also be caught, which would then need to be filtered out and classified as not a face in the face recognition step of the overall system.

4.1.2 Viola Jones Algorithm

Our Viola Jones algorithm was trained using images from a database was put together by the University of Washington [5]. This database contains approximately 13,000 faces and 10,000 background images. Some of the images in this database are shown in Figure 14 below.

Figure 14: Sample Database Images

We first implemented just one strong classifier containing a total of 3 weak classifiers. After experimenting with a larger number of weak classifiers when testing the strong classifier with our test images, we noticed decreased performance and larger false positive rates. Unfortunately, this does not agree with the findings of Viola and Jones [2]; this might be due to the relatively narrow training set we used. Due to slow training time and project time constraints, we used a total of 2500 images to train the classifier, 500 faces and 2000 background images. The detection rate that was obtained for this strong classifier was 94% and the false positive rate was 11%.

The top 2 weak classifiers that we obtained by our algorithm are shown in Figure 15 below:

Figure 15: Top 2 Weak Classifiers

It can be noted how the first weak classifier (pictured right) focuses on the eyes-forehead region of the faces, while the second one (pictured left) focuses on the eyes-cheek region. Figure 16 shows how the classifier is able to detect most of the faces in the image with a small amount of false positives.

Figure 16: Detected faces using Strong Classifier

We then constructed the attentional cascade classifier. For training, we started with 6000 images, from which 2000 were faces and 4000 were background images. The positive samples stayed constant for all stages, while the negative samples, after the first stage, were the false positives from the previous stage. After conducting many experiments with different number of stages and weak classifiers, we got best results with 3 stages, with 2, 10, and 15 weak classifiers per stage, respectively. The detection and false positive rates that were achieved were 78% and 5% respectively. However, our results as shown in Figure 17, do not agree with our expectations of fewer false positives and lower face detection when compared to Figure 16. This may be due to the decreasing number of negative training samples as we move forward through the cascade.

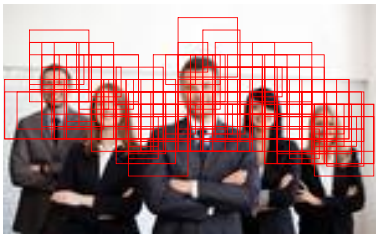

Figure 17: Detected faces using Attentional Cascade

Unfortunately, we were not able to accomplish great performance results as we ran into issues achieving the desired detection and false positive rate for each stage and the overall performance was decreased when more stages were added. We conducted a significant amount of experiments, but were unable to get optimal results. With

a larger amount of training data that resembled our actual testing images, we would expect the algorithm to achieve better results.

4.2 Face Recognition

4.2.1 Eigenfaces

In order to train our algorithm for face recognition using eigenfaces, a training set comprised of 100 images picturing 20 subjects (5 images per subject) from the AT&T Database was used. The images were taken at different times and with variations in lighting, facial expressions, and facial details. All the training images were used to determine the eigenfaces and form the face space. In this experiment, the eigenfaces were chosen to be the 30 largest principal components of the data as they led to the best performance results. By looking at the singular values associated with the principal components of the data and conducting experiments, we were able to determine that no significant improvement were observed when more than 30 eigenfaces were used. The top 9 eigenfaces for this dataset are shown in Figure 18. Then, all the subjects were characterized in the face space by taking the average of the projection of all their pictures onto the face space.

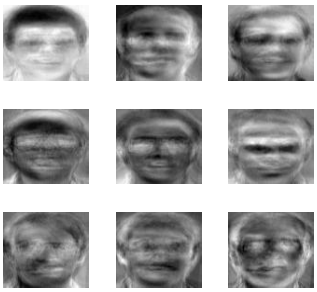

Figure 18: Top 9 Eigenfaces

In order to test the algorithm, a test set was constructed by using a different set of 100 images picturing the same 20 subjects, also from the AT&T Database. The performance obtained for this dataset was equal to 73%. When the test dataset was modified to also contain the images used for training, a performance of 80% was obtained. An example of how a test image is projected onto the face space and then reconstructed using the eigenfaces is shown in Figure 19.

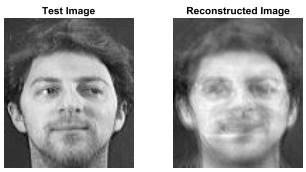

Figure 19. Reconstructed Image

Figure 20 shows the Mahalanobis distances of the test image with respect to the characterized subjects. The algorithm was able to successfully identify the subject as subject 7.

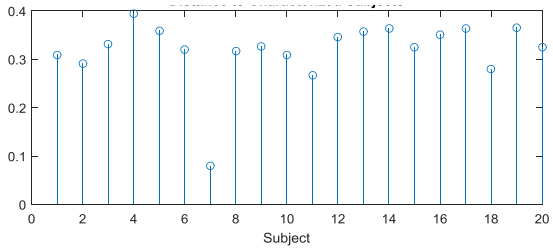

Figure 20. Distance to Characterized Subject

Although the results obtained for this algorithm give a fairly good performance, using eigenfaces for face recognition has many drawbacks. During our experiments, we observed that this method is not very robust against large changes in pose, lighting conditions, backgrounds, face size, and partial occlusion. In addition, we had difficulties setting up the threshold that divides the faces from the non-faces.

4.2.2 Neural Network

For the first stage of the neural network implementation, the CNN is trained with the University of Washington dataset [5]. For this first stage, the CNN is trained and tested using 2000 training images and 2000 test images consisting of half face and half non-face images. The trained model was able to determine whether the image input was a face or not a face with 96% accuracy as can be seen in Figure 21. The percentage of the testing data correct as a function of the training iterations can be seen in the following graph:

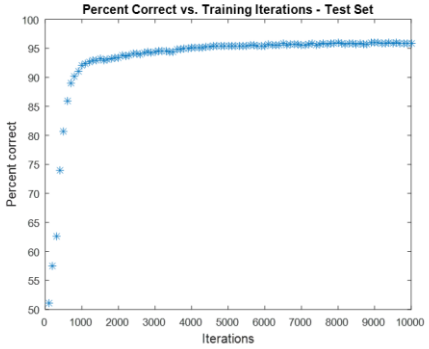

Figure 21. Percentage Correct vs. Training Iterations for Face vs. Non-face Detection

With further training and an expanded training and testing dataset, the results could certainly be further improved. The model was trained using Mini-Batch Gradient Descent with a learning rate of 0.0025 and a weight decay of 0.3 for 100 iterations of batches with 100 specimens each.

For the second stage of the neural network implementation, the ANN is trained with the AT&T Face

Database [6]. This database contains 40 people with 10 images of each person. This neural network was able to achieve an accuracy of 93.25% as can be seen in Figure 22. While a more complicated neural network may be used to achieve a marginally higher accuracy, we found that often led to overfitting due to the limited data set and did not make any significant improvements. The percentage of the testing data correct as a function of the training iterations can be seen in the following graph:

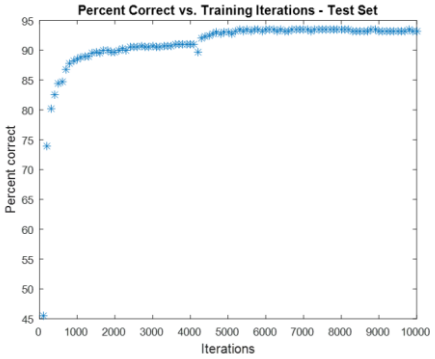

Figure 22. Percentage Correct vs. Training Iterations for Face Identification

In addition to the two stage system above, we also attempted to use the stage 1 face or non-face classifier to cascade through a test image and determine whether the pixel range contained a face or not. This methodology failed to yield satisfactory results. We believe this is due to the substantial differences between the University of Washington database used to train and the classroom image input.

5. Conclusion

In this project, we were able to deeply investigate several methods for face detection and recognition. After reading a large amount of journal papers, we decided to focus our efforts on four algorithms that we found interesting and considered would allow us to obtain a deep understanding of how these tasks have been approached in the past decades. The four methods that we implemented were skin-color blob detector, Viola-Jones, eigenfaces, and neural networks. After conducting experiments with all these methods, we were able to identify their advantages, as well as disadvantages, and draw comparisons between them.

With our skin-color blob detection algorithm implementation, we were able to accomplish great results for face detection with great robustness properties against pose changes and occlusion. However, as discussed previously, this algorithm must be modified depending on the specific classroom and is not ideal for large-scale roll-out. On the other side, even though we ran into many issues and were not able to accomplish optimal results with our Viola-Jones implementation, this algorithm has been proven to be very fast with great detection and false positive rates. [2,3] However, under large face rotation, harsh backlighting, and occlusion, this algorithm fails. For face recognition, the eigenfaces method is relatively easy to implement and provides fairly good performance. However, it has very poor robustness properties against small variations in face size, lighting, and pose. The NN method provides fairly accurate performance; however, it requires extensive training data which resembles the expected test.

As we experienced many difficulties while implementing some of these algorithms, we were able to learn a lot from our mistakes and get an excellent understanding of the methods we implemented. Looking back, we could have achieved more refined end results had we reduced the amount of research that we initially did and picked just one method to implement from the beginning. Nonetheless, we were able to learn a lot about computer vision topics, such as color spaces, image segmentation, principal component analysis, machine learning algorithms, and convolutional neural networks. We are very satisfied with our learnings from this project and hope to apply the knowledge learned in the future.

6. References

[1] S. Phung, A. Bouzerdoum, and D. Chai, "Skin segmentation using color pixel classification: analysis and comparison," IEEE Transactions on Pattern Analysis and Machine Intelligence IEEE Trans. Pattern Anal. Machine Intell., vol. 27, no. 1, pp. 148–154, 2005.

[2] P. Viola and M. Jones, "Robust real-time face detection," *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*.

[3] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*.

[4] M. Turk and A. Pentland, "Eigenfaces for Recognition," Journal of Cognitive Neuroscience, vol. 3, no. 1, pp. 71–86, 1991.

[5] "Assignment 4: Face Detection," *University of Washington - Computer Science Department*, 06-Jun-2011. [Online]. Available at: https://courses.cs.washington.edu/courses/csep576/11sp/assignment4.htm. [Accessed: 10-Apr-2016].

[6] "The Database of Faces," *AT&T Laboratories Cambridge* April-1995. [Online]. Available at: http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html [Accessed: 10-Apr-2016].