

# Face Detection Using Color Thresholding, and Eigenimage Template Matching

Diedrick Marius, Sumita Pennathur, and Klint Rose

## 1. Introduction

The goal of this project is to take a color digital image with over 20 faces and indicate the location of the center of each face in that image. Detection of faces in a digital image has gained much importance in the last decade, with applications in fields such as law enforcement and security. Although facial detection is an extremely simple task to the human eye, automating the process to a computer requires the use of various image-processing techniques.

Many different algorithms have been implemented for facial detection, including the use of color information, template matching, neural networks, edge detection and Hough transforms[1]. The approach used in this report follows one similar to a rejection scheme algorithm [2]. The first step is to reject regions in the image that are not faces based on color thresholding and skin segmentation. The next step is to use binary image processing to create clearer delineations in these regions. Template matching with both training image faces and eigenfaces using the Sirovich-Kirby [3] method is used to detect faces from the non-faces, and also used to find the position of faces. Other sub-routines including the separation of overlapping faces and removal of high aspect ratio and low standard deviation non-face images are iterated against the main detection scheme before a final output is created. The algorithm was implemented in MATLAB and is outlined in figure 1.

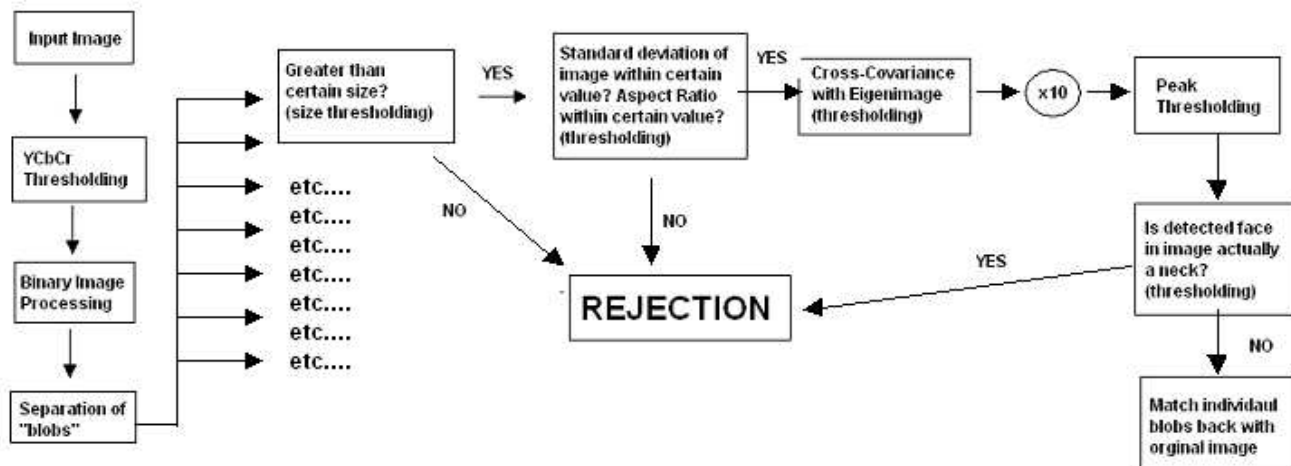


Figure 1: Block Diagram for entire face detection algorithm

The following sections describe the various techniques used to detect the faces. Section 2 covers the skin segmentation based rejection, section 3 describes the binary image processing done after skin segmentation, section 4 describes template matching using eigenimages, and section 5 discusses various attempts at troubleshooting issues such as overlapping faces. The result of the algorithm, presented in section 6, consisted of a 94.3% detection accuracy and 1minute 40 second run time on a 689MHz AMD Athlon laptop. Finally, sections 7 and 8 discuss possible improvements and conclusions.

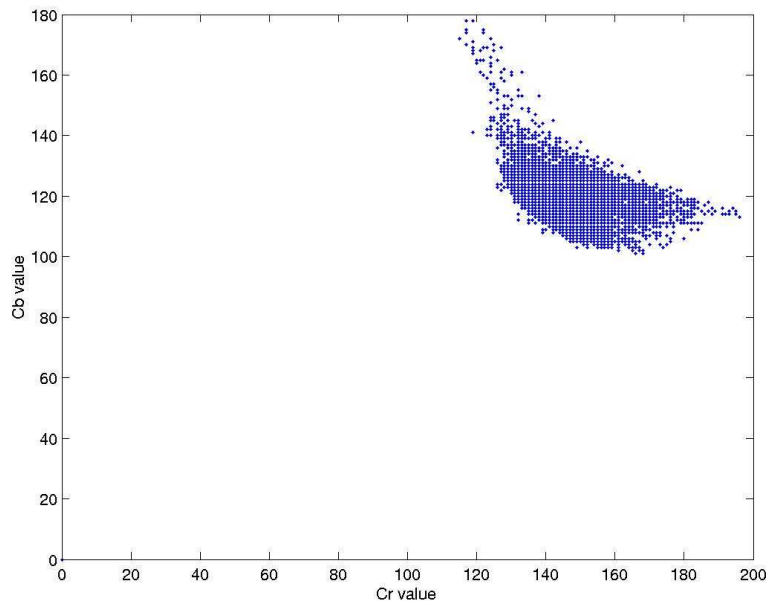
## 2. Skin segmentation

The first step in the face detection algorithm is using skin segmentation to reject as much “non-face” of the image as possible, since the main part of the images consists of non-skin color pixels. There are two ways of segmenting the image based on skin color: converting the RGB picture to YCbCr space or to HSV space. A YCbCr space segments the image into a luminosity component and color components, whereas an HSV space divides the image into the three components of hue, saturation and color value.

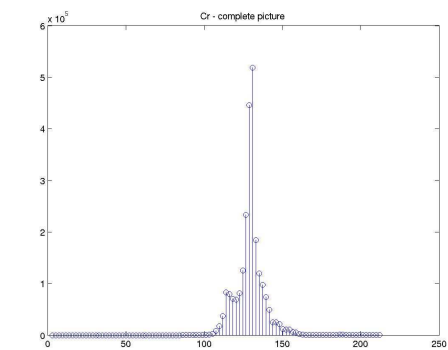
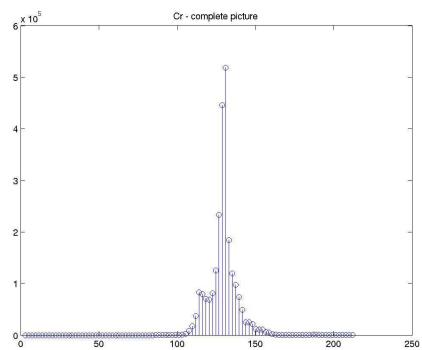
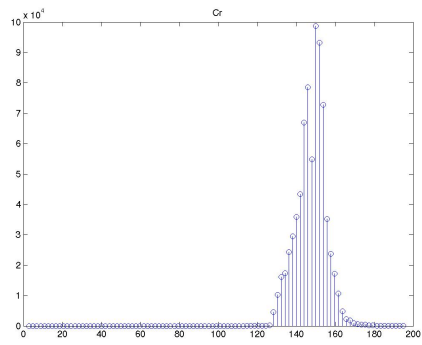
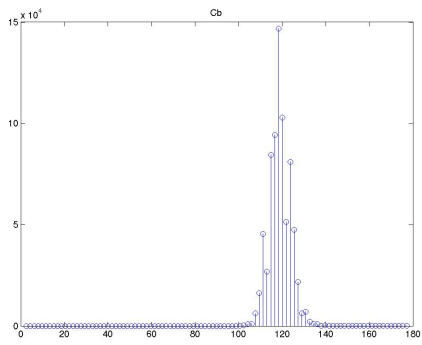
The main advantage of converting the image to the YCbCr domain is that influence of luminosity can be removed during our image processing. In the RGB domain, each component of the picture (red, green and blue) has a different brightness. However, in the YCbCr domain all information about the brightness is given by the Y-component, since the Cb (blue) and Cr (red) components are independent from the luminosity. The following conversions are used to segment the RGB image into Y, Cb and Cr components:

$$\begin{aligned} Y &= 0.257 * R + 0.504 * G + 0.098 * B + 16 \\ Cb &= 0.148 * R - 0.291 * G + 0.439 * B + 128 \\ Cr &= 0.439 * R - 0.368 * G - 0.071 * B + 128 \end{aligned} \tag{1}$$

The Cb and Cr components give a good indication on whether a pixel is part of the skin or not. This can clearly be seen in Figure 3, which are the Cb and Cr values of all the pixels that are part of the faces in the first five training images. (Only five images were used so that two images would be left to use as test images.) The faces were differentiated using the “ground truth data” on the project website. There is a strong correlation between the Cb and Cr values of skin pixels, as shown in figure 2. Figure 3 shows the Cb and Cr components of the entire training image, to reveal the comparison between faces and nonfaces in the YCbCr space. From these figures, it is apparent that background and faces can be distinguished by applying maximum and minimum threshold values for both Cb and Cr components. Note, however, that even with proper thresholds, images containing other parts of the body, such as exposed arms, legs, and other skin, will be captured, and most be removed in following image processing steps.



**Figure 2: Strong correlation between Cr and Cb values for skin pixels**



**Figure 2b: Cb and Cr values of the faces (top row) and the entire image (bottom row)**

The thresholds that were chosen based on the histograms in figure 2 and experiments with data are as following:

$$140 < Cr < 165$$

$$105 < Cb < 135$$

Narrowing down these thresholds increases the probability that the accepted pixels are actually part of the skin and that most of the background is rejected. The main disadvantage is that also a lot of actual skin pixels are rejected, which might be a problem in future steps in the algorithm. Broadening the thresholds obviously detects most of the skin pixels, but will also include a lot of non-skin pixels. The values above gave us the best results.

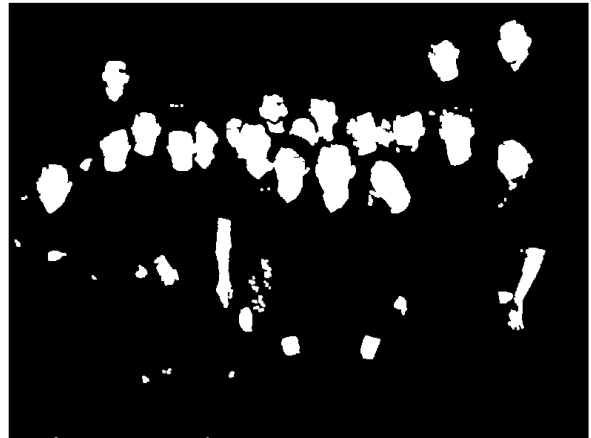
An alternative to thresholding in the YCbCr space is to use the HSV domain (color hues, saturations and values). Appendix B presents thresholding based on the hue and saturation values in a similar method to that used previously for the YCbCr space.

### 3. Binary Image Processing

Based on the Cb and Cr thresholding, a resulting black and white "mask" is obtained with all the faces in addition to some artifacts (body parts, background). This mask is then refined through binary morphological operations to reduce the background contribution and remove holes within faces. The image is first eroded with a 9x9 pixel face-shaped kernel to eliminate small background objects and separate individual faces. This eroded image is then dilated with a 10x10 pixel face-shaped kernel to refill gaps within the faces. To supplement the dilation, a hole-filling algorithm is used to fill any remaining holes inside faces.

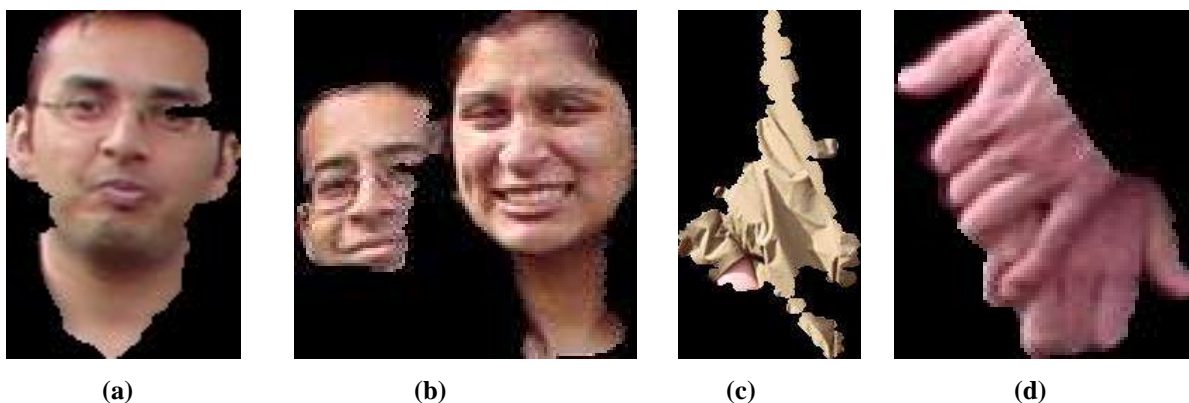


(a)



(b)

Figure 4: (a) thresholding based on Cb and Cr values. (b) resulting image after the dilution/erosion/hole-filling process



**Figure 5: Examples of “blobs” detected after binary image processing. (a) Face, (b) overlapping faces, (c) a non-face in the form of a coat, (d) a non-face as hands**

The effects of dilution and erosion were to effectively “remove” areas of the images where the “blob” size was less than 17 pixels and to create separations between individual faces. Once these morphological operations are performed, the face “blobs” are separated into individual images and the blobs smaller than 3000 total pixels are removed since the smallest training face was 60x60 pixels. The outputs of image segmentation using binary image processing are images that represent faces and non-faces (such as background and hands) as shown in Figure 5.

After the above morphological processing was performed, a few thresholding techniques were applied in attempt to remove non-faces. Typically, images that are faces are not all the same color, since they have different color features such as eyes, teeth and lips. For other body parts such as arms, these colors tend towards the same. Therefore, a threshold can be applied to take only those images that have a standard deviation, or variation of color, above a certain number. By the same token, an image such as a jacket in the wind will have a much higher standard deviation than a typical face. Thus, it was noticed that faces images fit within the certain standard deviation of 60 and 100. Furthermore, images such as arms and jackets have a much higher or lower aspect ratio than faces, so that applying jointly a thresholding of aspect ratio between 0.5 and 1.8 resulted in the rejection of many non-faces, such as in figure 5c.

#### **4. Template Matching/ Eigenfaces**

Once individual candidate face images are separated, template matching is used as not only a final detection scheme for faces, but also for locating the centroid of the face. The idea of template matching is to perform cross-covariances with the given image and a template that is representative of the image. Therefore, in application to face detection, the template should be a representative face - being either an average face of all the faces in the training images, or an eigenface. In our case, both templates were created. The first step was to crop out the faces from each training image posted on the website. Using these faces as our set of training images was justified since the same people would be present in the actual test image- otherwise a larger and more diverse set of training faces

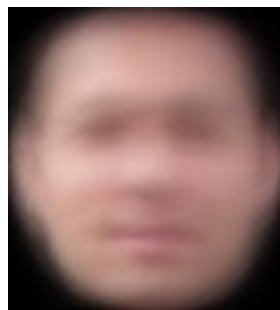
would have been used. 98 of 142 faces were acquired from the first 6 training images, choosing only full faces and those that were facing forward. (The 7<sup>th</sup> training image was to be used as a test.) After the images were acquired, they were resized to the average size, and then the mean of the faces were calculated, using the standard definition of average value:

$$mean = \langle m \rangle = \frac{1}{N} \sum_{n=1}^N m^n$$

where N is the number of samples. Examples of average faces and the mean face are shown below (Figure 5, Figure 6).



**Figure 5: Examples of average resized faces**



**Figure 6: average face**

The resized faces were also used to find eigenfaces using the Sirovich-Kirby method [1]. Detection using eigenfaces was the test used to determine whether or not a portion of the image was a face (see figure 1). The basis behind using this method is the assumption that the space of faces can be spanned by eigenvectors. The method used to find eigenimages starting with the subtraction of the mean from each of 98 resized faces

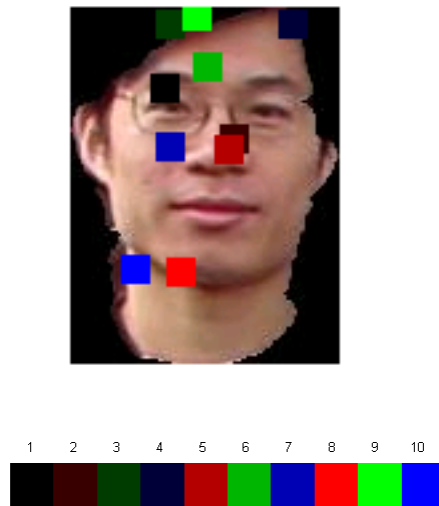
(152x152 matrix) and converting each image to gray scale. (Gray scale was chosen since it drastically increased the speed of the algorithm at a low cost of performance accuracy in the detection routine). Next each image was stored as a column vectors in larger matrix, S, with the size of S being (152\*152) x 98. Following the Sirovich-Kirby method, the matrix  $L=S^T S$  is computed, from which a set of eigenvalues and eigenvectors are computed. Thus we have:

$$S^T S v_i = \lambda_i v_i$$

where  $v_i$  are the eigenvectors, and  $\lambda_i$  are the corresponding eigenvalues. From this result, it is evident that multiplying this equation by S will give us the eigenmatrices of  $SS^T$ , as

$$SS^T S v_i = \lambda_i S v_i$$

Therefore, the eigenfaces,  $U_i=S v_i$ . In the algorithms used for the final face detection, 10 eigenimages were calculated corresponding to the 10 largest eigenvalues, and presented in Appendix C. Figure 7 shows a comparison of where each eigenimage (1-10) would detect the centroid of the face. From figure 7 and other similar face images, it becomes apparent that eigenimage 2 is the most accurate. Therefore, this image was used as the kernel for cross covariance. Figure 8 displays the actual eigenimage 2.



**Figure 7: Accuracy of first 10 eigenimages in detecting centroid of face**



**Figure 8: Eigenimage 2 used in face detection routine**

Using eigenimages for a template kernel proved to be the most accurate in not only detecting faces, but also detecting location of faces.

## **5. Post-Processing**

As mentioned previously, several smaller images with a high probability of faces were gathered from the input training image after the binary image-processing step. Other skin parts such as hands, arms and legs were removed based on their size (hands are smaller than faces) or shape (e.g. arms) from standard deviation and aspect ratio thresholding.

In order to detect individual faces as well as multiple faces within a single blob, the detection algorithm systematically found ten potential face locations within each blob. In each of ten iterations, the peak of the cross-covariance between the blob and the eigenimage was marked as a potential face centroid. A 120x120 pixel (average size of a face) area around this point was then set to 0's to remove any contribution into the next cross-covariance calculation and the process was repeated. After ten iterations, ten potential centroids were located within the blob irrespective of the number of actual faces.

Using a threshold value determined from cross-covariances with 132 single faces, centroids whose cross-covariance peak value fell below this point were disregarded. The remaining points were tested to determine whether they were on or near the same vertical axis as another centroid. Those that were on-axis and also near the point in the horizontal direction were assumed to be necks since the neck was the second most highly correlated area other than the face. If a point fit this "neck" criterion, it was removed. The remaining points were assumed to be face centers and their location with respect to the full image was recorded.

Other methods attempted to delineate between overlapping faces include edge detection and creating smaller kernel such as nose or eye kernels to identify a face. These methods proved to be inadequate for separating faces, however. Edge detection would at times remove too much of the overlapping face, resulting in an image which would be impossible to detect using a kernel. Nose and eye kernels were difficult to implement because too many background images looked like a nose, and given the outdoor light in the images, many people were squinting, making their eyes hard to recognize.

## **6. Performance**

In this experiment, the detection rate of the system is 95% with 4.3% false positives. Moreover, the result demonstrates that the system works extremely well for the images whose faces are full, upright, and facing towards the front. The false positives primarily occurred on large non-face body parts such as arms and legs. The false negatives were typically due to an obstructed face or variations that caused separations in the face such as sunglasses. Examples of the image with detected faces marked are shown in figure 9 and in the appendix D.

Since the final demonstration of the face detection algorithm includes a time limit, the average run time for the code was calculated for the laptop computer used. The run time for the code averaged 100 seconds with approximately 60 seconds for the skin segmentation and the remaining time for the actual detection.

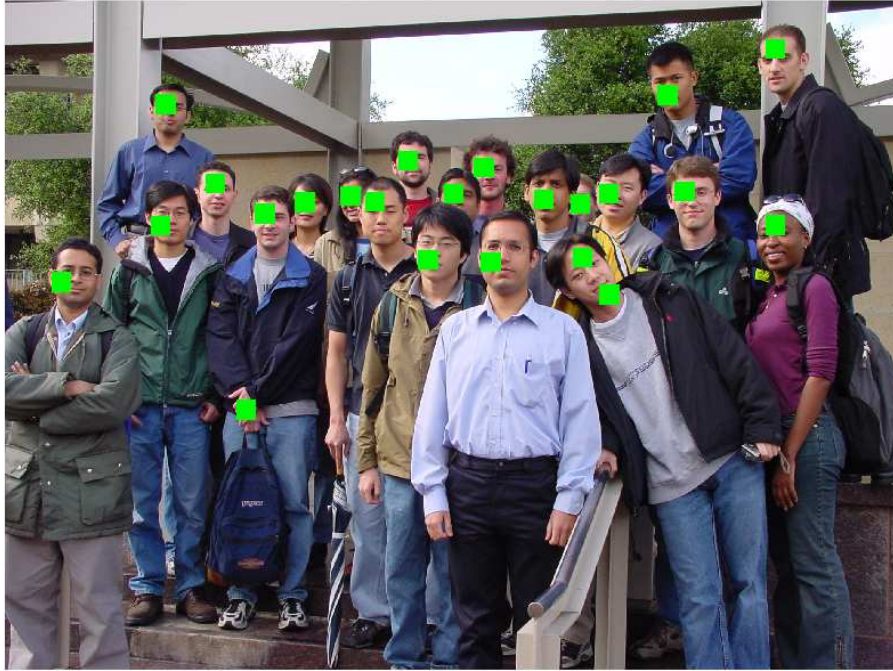


Figure 9: Face detection results for training image 1

	# faces	# Correct	False Positive	False Negative	Percentage
Trainingimage1	21	21	1	0	100
Trainingimage2	23	21	5	2	91
Trainingimage3	23	22	0	1	96
Trainingimage4	24	22	0	2	92
Trainingimage5	24	23	0	1	96
Trainingimage6	24	23	0	1	96
Trainingimage7	22	19	1	3	86
<b>Total</b>	<b>161</b>	<b>151</b>	<b>7</b>	<b>10</b>	<b>95</b>

## 7. Possible Improvements

Improvements in coverage and accuracy can be made by using a larger set of training images that includes more diverse lighting situations, and by creating a more exact alignment of both the various training faces and the possible face regions. Finding a way to better account for separation of overlapping faces, such as creating more accurate eye and nose kernel can possible help with this. Employing different strategies in parallel,

such as neural networks, may also increase the accuracy of the problem. However, in the scope of this project, these solutions seemed unreasonable.

## **8. Conclusions**

A face detection algorithm is proposed that combines pixel thresholding with eigenface decomposition. The algorithm is able to detect faces at different scales in a given image, as well as slightly tilted images. However, it has a hard time breaking apart overlapping faces if they are too near each other vertically. Future work could include the use of rotated eigenimages or the implementation of a neural network or a linear classifier as a secondary detection scheme.

## **REFERENCES:**

- [1] Ming-Hsuan Yang and Narendra Ahuja “Detecting Human Faces in Color Images”. Beckman Institute and Department of Electrical and Computer Engineering University of Illinois at Urbana-Champaign.
- [2] Elad, M. *Rejection Based face Detection* EE368:Digital Image Processing Lecture, Stanford University, Stanford, CA. May 19, 2003
- [3] L. Sirovich and M. Kirby, “Low-dimensional procedure for the characterization of human faces”, *Journal of Optical Society of America*, vol. 4, pp. 519, 1987.

## **APPENDIX A** – Division of Labor

Color Thresholding – Diedrick

Eigenimages – Sumita

Binary Image Processing – Klint

Std. Dev. and Aspect Ratio Thresholding – Sumita

Template Matching – Klint

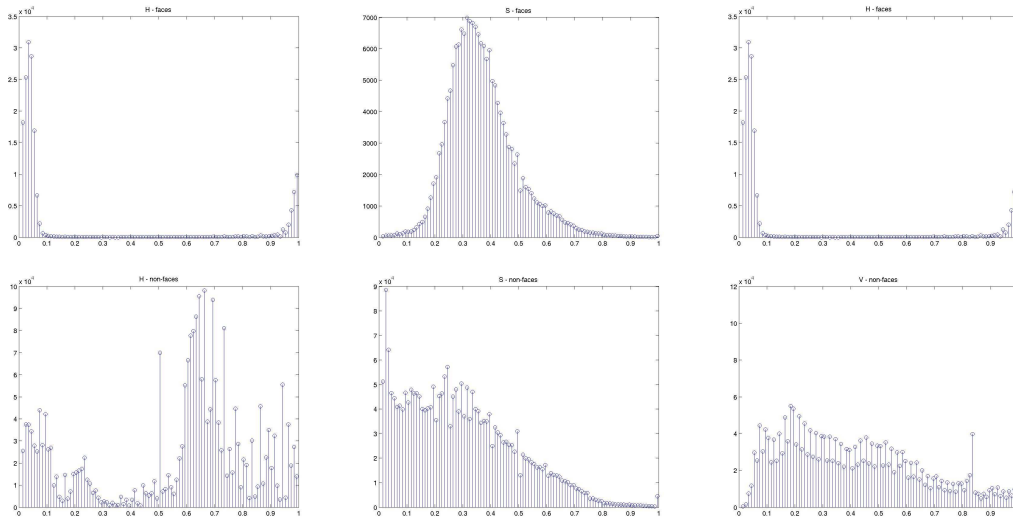
Post-Processing – Klint

Many other trials and tribulations of code – Diedrick, Klint, Sumita

Report – Klint, Sumita , Diedrick

## APPENDIX B – HSV values

An alternative to thresholding in the YCbCr space is to use the HSV domain (color hues, saturations and values). Appendix B presents thresholding based on the hue and saturation values in a similar method to that used previously for the YCbCr space. The color values cannot be used due to their slowly varying distribution. When applying both the YCbCr and HSV technique, the same results were achieved, and thus the conclusion was drawn that there is no advantage in using both techniques in our image processing algorithm.

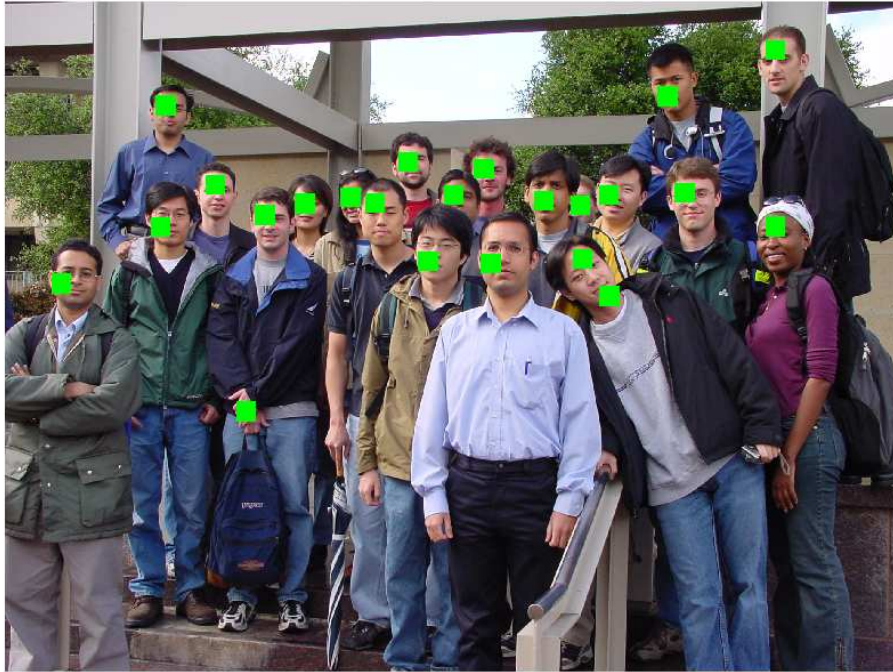


**Figure B1: Color Hue, Color Saturation and Color Values of the faces (top row) and that of an entire image (bottom row).**

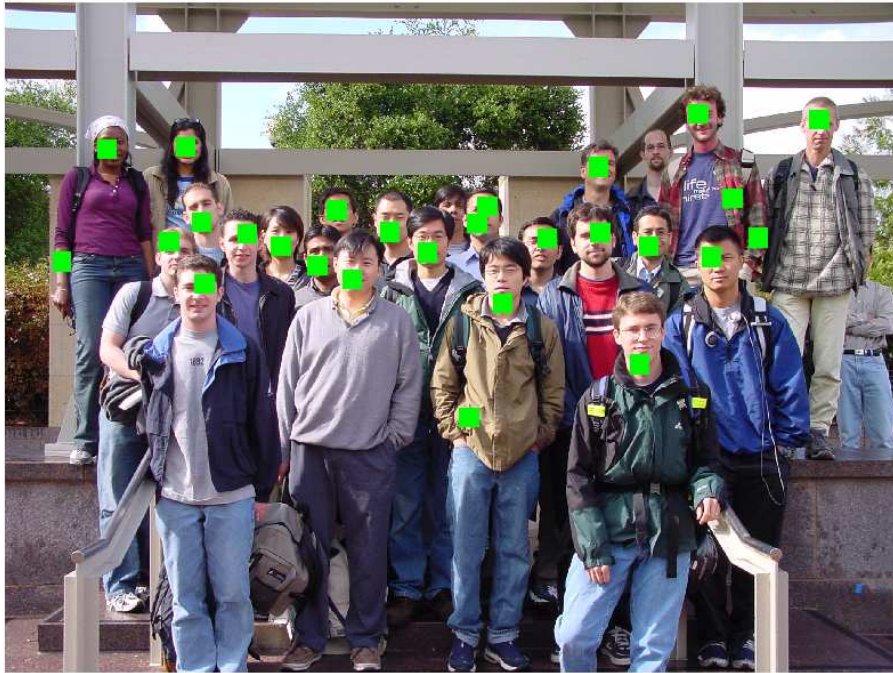
APPENDIX C– Eigenimages



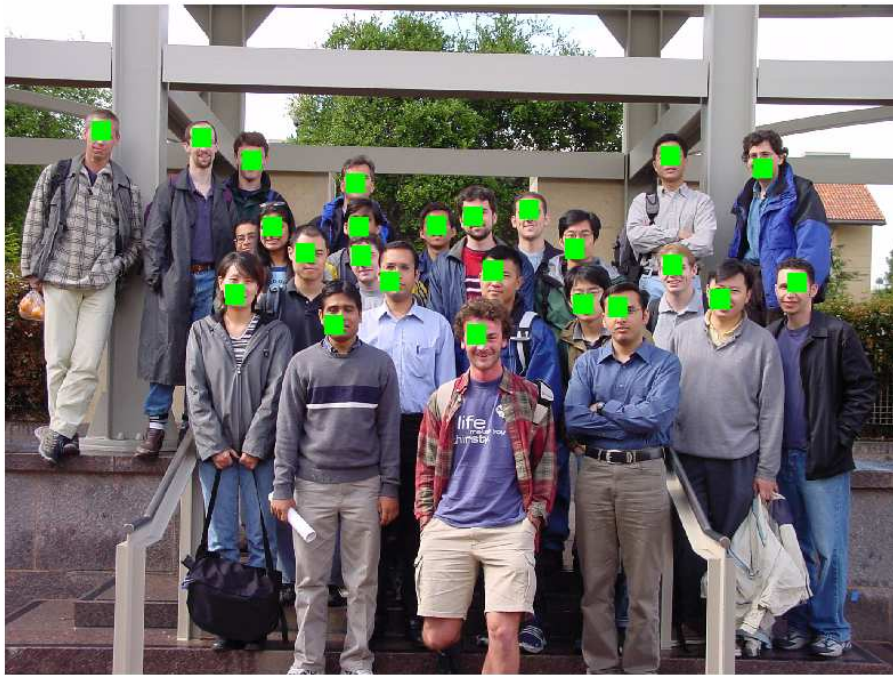
**APPENDIX D**– Detection results



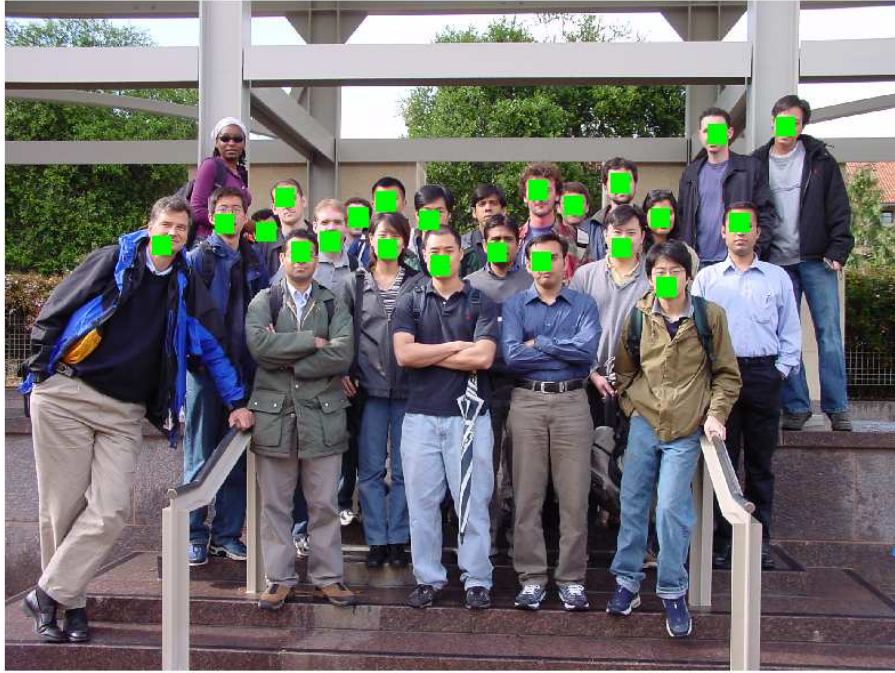
Training 1



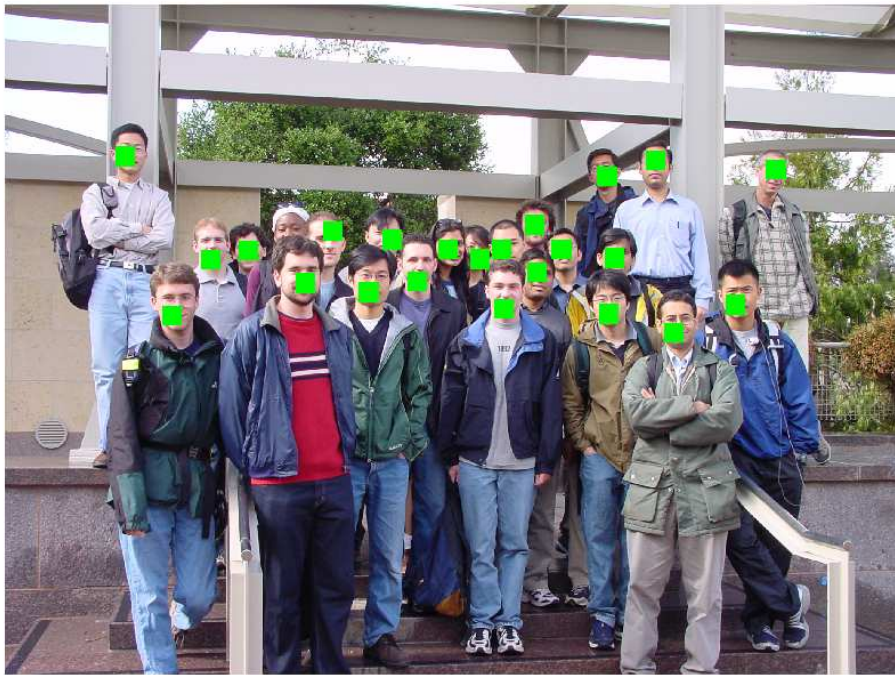
Training 2



Training 3



Training 4



Training 5



Training 6



Training 7