CS 479 Pattern Recognition
Programming Assignment 2 Parameter Estimation and Skin-Color Classification
Julia Adamczyk, Brendan Aguiar

A shared private Github repository, Git, and Google docs were used to collaborate in this project. Both team members contributed equally to the project and clarification of work allocation was documented to further show transparency where appropriate. The source code has a list of version history. A further breakdown of the division of work is shown:

Julia - case 3 improvement to incorporate non-diagonals of covariance matrices, theory; calculate_mean function; randomizing the training data sets to specific sizes; deriving the test results for each parameter estimation, experiment 3 was mainly implemented together in the team session

report: implementation (modification to case 3; generating random sets, mean), discussed the result of experiment 1

Brendan - Collaborated with Julia on experiment 3. Implemented header/ source files for experiment 3. Created training model. Adjusted threshold values. Calculated g(x) for different threshold values.

report: theory: implementation (covariance), parameter estimation, discussed the result of experiment 2. Generated ROC Curves. Added content to experiment 3 report.

Theory

Parameter Estimation

The ML Parameter Estimation is used in this assignment to maximize the likelihood function (i.e. finding such values of mean and covariance for which the data can have the best fit.) Parameter Estimation can produce better results than results from Programming Assignment 1 because the mean and covariance are better estimated. In general, Maximum Likelihood Estimation (MLE) involves taking the values of mean and covariance θ that is most likely to give rise to the dataset D of p(D|θ). The expected mean and covariance can be found by taking the gradient of the likelihood function, setting it equal to zero, and solving for mean covariance. The sample mean and sample variance formulas result from the calculation. The sample mean is unbiased, but the sample variance is not, meaning they differ from the expected mean and covariance. To overcome this, the sample mean and sample covariance was used in ML Parameter Estimation to classify the samples.

Image Processing

For this project, the code for the image processing procedure that reads and writes ppm images was given. Processing ppm files is quite easy as the ppm file is a file that holds the values for each pixel in the image. In the grayscale images, the value of the pixel is an int whose value varies between 0 and 255 (0 represents black and 255 represents white). The RGB image has three different pixel components: Red, Green, Blue, and it is more complex than a greyscale. Ppm files consist of two parts (1) the header which contains the image type, width, height, and maximum pixel value, and (2) the image data. In order to read in a color image the 2D array of RGB values need to be created (one entry per pixel). Then the values from the binary file are converted to RGB integer representation of each pixel. That way the image can be stored in a 2D array and easily manipulated in the code. To see the results of classification, the write image function, which does the opposite transformation is used. This function converts the class of image type back to the ppm file which then can be converted to the human-readable form like jpg.

Skin Distribution Case Study

In their case study, Young and Weibel proposed a face recognition algorithm that detects and tracks faces based on skin color. They believed that this method would be faster than classifying faces based on the facial features because the skin color classifier could offer a faster and more robust algorithm. After testing data they discovered that skin color distribution forms a cluster that follows a Gaussian distribution. This fact significantly simplified the methodology. Experiment 3 is supposed to follow along with the case study and build a skin-color classifier that will detect faces in the provided images. The data from the color image is first collected and normalized into the chromatic space. Then using the fact that the skin color model follows a Gaussian Distribution the mean and covariance of two of the chromatic colors are estimated. Using these variables the value of the discriminant function specified in Eq. 1 is calculated.

$$g(\boldsymbol{x}) = -\frac{1}{2\pi|\Sigma|^{-1/2}} exp[-\frac{1}{2}(\boldsymbol{x} - \mu)^t \Sigma^{-1} (\boldsymbol{x} - \mu)] \quad (1)$$

Then, g(x) is compared to a set threshold and assigned to the correct class. If the value of g(x) is larger than the threshold the pixel is assigned to the skin color distribution and its value in the reference image is set to white (255). If it is smaller, then the value is set to black (non-skin color distribution). The optimal threshold can be chosen by testing the same image multiple times with different thresholds and calculating False Positives and False Negatives. The point on the ROC curve where FP = FN is called the Equal Error Rate and is the optimal threshold value.

Different Color Spaces

For Experiment 3 two different color spaces are specified: chromatic color space, and YC_bC_r color space. Normalized chromatic color space can be calculated from the following:

$$r = R / (R + G + B)$$
$$g = G / (R + G + B)$$

The b value is redundant since r + b + g = 1.

The YC_bC_r color space can be calculated using the following three equations:

$$Y = 0.299R + 0.587G + 0.114B$$
$$C_b = -0.169R - 0.332G + 0.500B$$
$$C_r = 0.500R - 0.419G - 0.081B$$

However, Y value can be omitted in skin color model for the model simplicity (it becomes a 2D Gaussian).

Implementation

The assignment is split into three test files, one for each experiment. To test set A use experiment1.cpp file, to test set B use experiment2.cpp, and to test the images use experiment3.cpp.

Non-Diagonal Covariance Matrices

Both the diagonal and non-diagonal covariances were calculated and compared when classifying the datasets. The code contains commented-out lines for both calculating the non-diagonal values as well as setting them to zero. When estimating the covariance, scaling factors between the two classes are not exactly equivalent for both the diagonal and non-diagonal matrices, making the appropriate discriminants for set A and set B case 3. It is important to note that while the covariance matrices are no longer diagonal, they are still symmetrical, but the negative values that appear as non-diagonal elements make the matrix non-diagonalizable.

Modification to Case 3 Function

Using parameter estimation in the assignment changed the way the samples are classified. All classification uses case 3 now because the diagonal entries are not equal anymore. This makes the model more complex as it extends the calculation and incorporates the correlation

between the variables. In addition, covariance matrices are not diagonal anymore. The case 3 function from the previous assignment needed to be adjusted in order to incorporate the newly included non-diagonal entries into classification. The following changes have been made in order to improve case 3 function: the inverse of the matrix is calculated differently using the full formula, the determinant of the matrix function is also updated with the non-diagonal entries, and the non-diagonal entries are included in calculating each addend of the case 3 formula.

Generating Random Sets

Generating random sets from the original set was challenging because the data from two different classes were stored in the same file. The way that this function is implemented in this code is as follows. The function takes in three arguments, the original set, the new set, and a new set size that the original sets should be converted to. The class 1 size and class 2 size are calculated in order to make sure that the new data set draws the specific number of samples from the original set. Then the set of indices is created in order to be randomized with the indices from the original set. The 'stl' set structure is used to keep the values inside the set unique. Then, the for loop goes through the set of indices and pushes the value corresponding to each index into the new data set. As the implementation from Programming Assignment 1 used a 'stl' list that prevents random access, the whole code needed to be adjusted to use 'stl' vector instead.

Mean and Covariance Estimation

In order to calculate the mean, the program needs to loop through the whole set, add the values to corresponding sums and then divide each sum by the sample size. For the covariance, the square roots were taken to normalize the diagonal values. The non-diagonal values were not normalized to prevent complex numbers from appearing in the code. The unbiased estimate was used in calculating the covariance matrix.

Experiment 3

Experiment 3 generates a classifier that classifies faces in the images by converting the color image into a black and white image with faces indicated in white and non-faces in black. The menu that displays to the screen has two options; (1) train data and (2) test data. Option 1 is used to generate the skin color distribution from two training images, one color and one black and white with face indication in white. In order to train data, both images are converted to ImageType using the image processing function. Then, both images are looped through and the values of the pixels in the reference image that are not black are added to the skin distribution model. (The pixels are first normalized to reduce the number of variables). Then the mean and covariance of the skin distribution are calculated similarly to experiment1 and experiment2. This is all done by the training() function. The mean and covariance calculated in the training part are then used in testing data. The testing function works as follows. The Reference image is allocated. This image is the black and white image that will be created using the skin color distribution. The code loops through the color image and takes the values of the pixels, normalizes them, calculates the discriminant function, and compares it to the threshold. If the value of the discriminant function is larger than the value in the reference image is set to white (it is considered a face), else it is set to black). Function model is the implementation of the discriminant function. In order to test different color spaces change the

c value and switch training and testing functions. In order to test the optimum thresholds change the values inside the testing function that is called in switch case 3. Optimum value for chromatic space image 3 is 140, optimum value for chromatic space image 6 is 130, optimum value for YCbCr color space image 3 is , and optimum value for YCbCr color space image 3 is . The optimum values in this code are calculated manually. The FP and FN for each threshold are collected and then graphed in order to find the Equal Error Rate.

Results and Discussion

Experiment 1

Experiment's 1 purpose was to test the Set A classification with parameters estimated with MLE. Table 1 indicates the test results for different training data set sizes and different covariance matrices. In the first case, non-diagonals of the covariance matrix are explicitly set to 0 which makes the features uncorrelated. In the second case, correlation is present. Given the data in the Tab 1 following statements about the results can be made. Firstly, the size of training data influences the classifier performance. As shown in Table 1, the overall performance of the classifier drops when the size of the training data sample decreases. This happens because the estimation of the mean and covariance are far off the values that the set was generated from. What is surprising, class 1 has a much higher misclassification rate than class 2, however, it aligns with the results from Programming Assignment 1 where the results for class 2 were better as well (misclassification rate: 1.048% for class 2 vs 2.805% for class 1).

Experiment 1 was expected to improve the performance of the classifier, however, it has not changed much. The best performance from experiment 1 is achieved when testing the parameters received from the training sample of size 2,000 using the covariance with non-diagonals equal to 0 and is indicated in bold in the table. The overall misclassification rate is equal to 1.573% where the previous assignment achieved 1.575%. It does not seem to be a significant improvement. Another thing is that the original covariance matrices, whose non-diagonal values are incorporated, produce worse classifier results than the ones set explicitly to 0. This happens because now we treat the variables as correlated which may make the classifier more sensitive when classifying. The results of this test show that using parameter estimation is only helpful when we do not know the real values of mean and covariance. When using MLE and incorporating the correlation of the features the performance might be a little reduced because (1) the complexity of the model is increased (using case 3 instead of case 1) and (2) correlation is increased.

Table 1: The error rates for set A decrease as the parameter estimation size increases.

Set A	**Non-diagonals Explicitly Set to Zero**			**Original Matrices**		
	Mean	*Covariance*	**Error Rates*	*Mean*	*Covariance*	**Error Rates*
	Parameter estimation size 200,000			Parameter estimation size 200,000		
Class 1	[1.00875, 1.00177]	[1.00252, 0, 0, 1.00075]	(1) 0.0279667 (2) 0.0105 (3) 0.01574 (4) 0.0490056	[1.00875, 1.00177]	[1.00252, 0.00022229, 0.00022229, 1.00075]	(1) 0.0279167 (2) 0.0105143 (3) 0.015735 (4) 0.0490049
Class 2	[3.99649, 4.001]	[1.00306, 0, 0, 1.00216]		[3.99649 4.001]	[1.00306, -0.0006638, -0.0006638, 1.00216]	
	Parameter estimation size 20			Parameter estimation size 20		

Class 1	[0.614205, 0.828851]	[0.973265, 0, 0, 0.800597]	(1) 0.0333667 (2) 0.00898571 (3) 0.01627 (4) 0.0182842	[0.614205, 0.828851]	[0.973265, -0.700792, -0.700792, 0.800597]	(1) 0.14985 (2) 0.000757143 (3) 0.045485 (4) 0.014238
Class 2	[3.78235, 4.57136]	[0.956608, 0, 0, 1.02571]		[3.78235, 4.57136]	[0.956608, 0.0731908, 0.0731908, 1.02571]	
	Parameter estimation size 200			Parameter estimation size 200		
Class 1	[1.03484, 0.940728]	[1.04706, 0, 0, 0.927048]	(1) 0.0361 (2) 0.00772143 (3) 0.016235 (4) 0.056594	[1.03484, 0.940728]	[1.04706, -0.148715, -0.148715, 0.927048]	(1) 0.03785 (2) 0.00734286 (3) 0.016495 (4) 0.0562211
Class 2	[3.89061, 3.81839]	[0.902248, 0, 0, 1.0576]		[3.89061, 3.81839]	[0.902248, 0, 0, 1.0576]	
	Parameter estimation size 2,000			Parameter estimation size 2,000		
Class 1	[0.964396, 1.012]	[1.03467, 0, 0, 0.96775]	(1) 0.0289833 (2) 0.01005 (**3) 0.01573** (4) 0.0481794	[0.964396, 1.012]	[1.03467, 0.00060498, 0.00060498, 0.96775]	(1) 0.02825 (2) 0.0104 (3) 0.015755 (4) 0.0481714
Class 2	[3.97198 3.98948]	[1.0002, 0, 0, 0.974536]		[3.97198 3.98948]	[1.0002, -0.0253665, -0.0253665, 0.974536]	
	Parameter estimation size 20,000			Parameter estimation size 20,000		
Class 1	[1.009 1.00297]	[1.01097, 0, 0, 0.998109]	(1) 0.0281833 (2) 0.01045 (3) 0.01577 (4) 0.0496914	[1.009 1.00297]	[1.01097, 0.00115545, 0.00115545, 0.998109]	(1) 0.0281167 (2) 0.0104786 (3) 0.01577 (4) 0.0496914
Class 2	[3.99439 3.9931]	[1.00559, 0, 0, 1.00355]		[3.99439 3.9931]	[1.00559, -0.0020876, -0.0020876, 1.00355]	

**Error Rates: 1 – misclassification of class 1, 2 - misclassification of class 1, 3 – total misclassification, 4 – Bhat Bound*

Experiment 2

The purpose of experiment 2 is to test the Maximum Likelihood estimation on set B. Set B has more variation between the two classes than set A. For that reason, it was expected that our estimation would not perform as accurately as set A. As the parameter size decreased, the error rates of both classes fluctuated. At first, class 1 had an underestimated error rate while class 2 had an overestimated error rate. The bhatt. bound was also underestimated. As the parameter size increased, the error rates started to get closer to the actual error rates of the entire population. Similarly, the estimations for the sample mean and covariances got closer to the population mean and covariance as the sample size increased.

The parameter estimation for non-zero non-diagonals and zero non-diagonals are also compared. As shown in Table 2, as the parameter estimation size increased, the non-diagonal values got closer to their true variance of zero. The error rates for set B are more accurate when using the covariance of zero non-diagoanls. The actual mean for set B, class 1 and 2 is [1,1] and [4,4] respectively. The actual covariances are [1,0,01] and [4,0,0,8]. Estimating the covariance and mean values of the entire population will be near the actual values and never off by more than a hundredth’s place value.

Table 2: The sample means and covariances of set B are estimated more closely to the expected mean and variance as the sample size increases.

Set B	**Non-diagonals Explicitly Set to Zero**			**Original Matrices**		
	Mean	*Covariance*	**Error rates*	*Mean*	*Covariance*	**Error rates*
	Parameter estimation size 200,000			Parameter estimation size 200,000		
Class 1	[1.0056, 0.999163]	[1.0032, 0, 0, 0.999268]	(1) 0.0855833 (2) 0.0641929 (3) 0.07061 (4) 0.162114	[1.0056, .999163]	[1.0032, -0.00339905, -0.00033905, 0.999268]	(1) 0.085216 (2) 0.064271 (3) 0.070555 (4) 0.162113
Class 2	[4.00041, 3.98646]	[4.0074, 0, 0, 7.98825]		[4.00041, 3.98646]	[4.0074, -0.0183193, -0.0183193, 7.98825]	
	Parameter estimation size 20			Parameter estimation size 20		
Class 1	[0.854012, 1.65432]	[1.00301, 0, 0, 0.830733]	(1) 0.04655 (2) 0.0769286 (3) 0.067815 (4) 0.111469	[.854012, 1.65432]	[1.00301, -.121353, -.121353, 0.830733]	(1) 0.035633 (2) 0.084978 (3) 0.070175 (4) 0.110241
Class 2	[3.04172, 6.96848]	[2.92582, 0, 0, 8.45447]		[3.04172, 6.96848]	[2.92582, -0.803986, -0.803986, 8.45447]	
	Parameter estimation size 200			Parameter estimation size 200		
Class 1	[1.13965, 0.950241]	[1.06787, 0, 0, 1.12162]	(1) 0.04065 (2) 0.0896214 (3) 0.07493 (4) 0.123586	[1.13216, 0.971673]	1.07747, 0.0234678, 0.0234678, 1.17636]	(1) 0.01695 (2) 0.132214 (3) 0.097635 (4) 0.099955
Class 2	[3.77674, 5.75824]	[3.64963, 0, 0, 7.80524]		[3.79051, -1.75212, -1.74212, 7.9174]	[3.94016, 6.36216]	
	Parameter estimation size 2,000			Parameter estimation size 2,000		
Class 1	[1.01117, 0.951178]	[0.971091, 0, 0, 1.01134]	(1) 0.0810167 (2) 0.0668929 (3) 0.07113 (4) 0.152254	[1.07584, 0.998925]	[1.02037, 0.0133035, 0.0133035, 1.01972]	(1) 0.05548 (2) 0.08634 (3) 0.07708 (4) 0.15512
Class 2	[4.08557, 4.09703]	[3.87561, 0, 0, 8.25385]		[4.07484, 4.0987]	[3.87231, -1.77279, -1.77279, 8.19491]	
	Parameter estimation size 20,000			Parameter estimation size 20,000		
Class 1	[1.00279, 0.995741]	[1.00775, 0, 0, 1.00894]	(1) 0.0869167 (2) 0.0640929 (3) 0.07094 (4) 0.164188	[1.02466, 0.988441]	[1.0015, 0.0139823, 0.0139823, 7.9151]	(1) 0.08348 (2) 0.06520 (3) 0.07069 (4) 0.16526
Class 2	[3.97052, 3.96698]	[3.98705, 0, 0, 7.92719]		[3.96721, 3.96677]	[3.98715, -0.167311, -0.167311, 7.9151]	

**Error Rates: 1 – misclassification of class 1, 2 - misclassification of class 1, 3 – total misclassification, 4 – Bhat Bound*

Overall the results show that sometimes it is better to have a simpler form of mean and covariance because it produces similar results while having a less complicated classifier. Especially when the model is based on the simpler mean and covariance. For proper

classification of set B, case 3 should be implemented, the parameter estimation size should be 20,000 or greater, and the non-diagonal rows should not be estimated.

Experiment 3

The skin color face recognition algorithm from the case study that was implemented in experiment 3 produced accurate results. The images with face classification are shown in Fig. 2 and Fig. 5 with their original corresponding images Fig. 3 and Fig. 6. The noise can still be noticed on both of these images, however all the faces seemed to be recognized at least partially. The noise in this case is considered as white pixels in non-faces spots and black pixels in face spots. The noise in both images comes from the fact that the Equal Error Rate is used to produce both images. As different thresholds were tested, the resulting images became lighter when the threshold was decreased and darker when the threshold was increased. This is because as the threshold rises, fewer pixels are classified to skin color and are, as a result, set to white in the reference image. Using Equal Error Rate (EER) allows to produce the most optimal image because the black and white pixels are then misclassified equally. Therefore, the images created using the EER as shown in Fig. 5 and Fig. 6 have the least amount of noise possible.

In order to calculate EER, False Positives and False Negatives for different thresholds are calculated and plotted superimposed on one another in order to find the point where the both curves cross. This point is the optimal threshold for the given image for the experiment 3 classifier. ROC curves for testing image 3 and testing image 6 are shown in Fig. 1 and Fig. 4. This part of the experiment tested the classifier that used the chromatic color space in order to classify the faces on both images (experiment 3 part a).

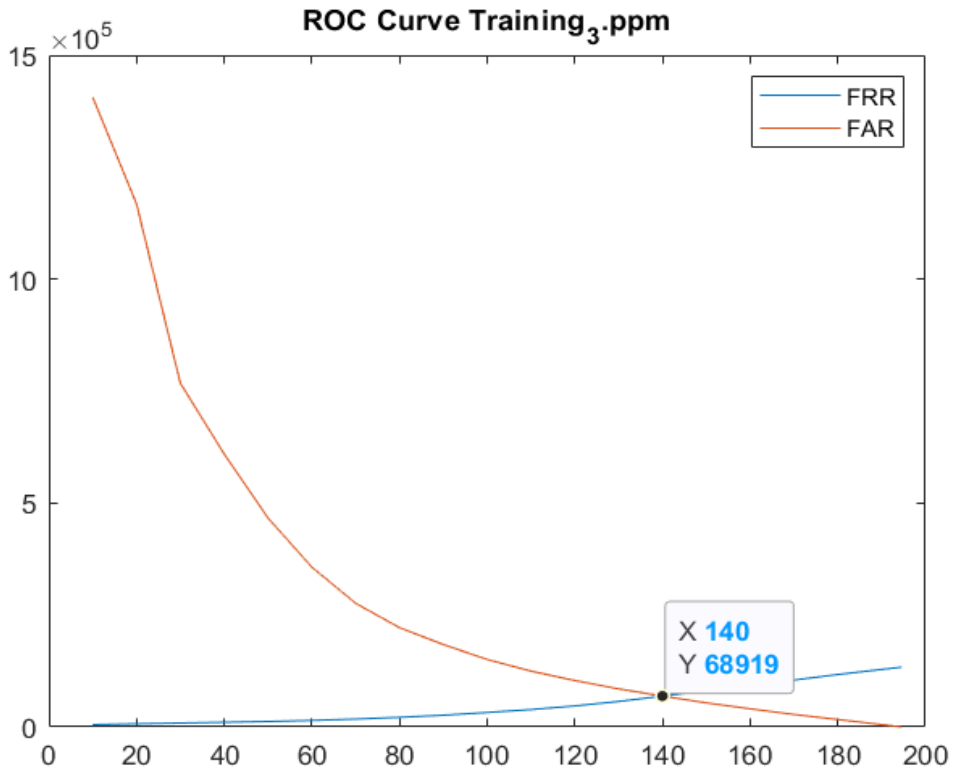


Fig 1: The ROC curve for test image 3 indicates that the optimal threshold for testing image 3 will be 140.

Fig 2: The reference image 3 produced by the experiment 3 classifier with a threshold of 140.

Fig 3: The original image 3 which was used as a testing image for skin color face recognition algorithm.

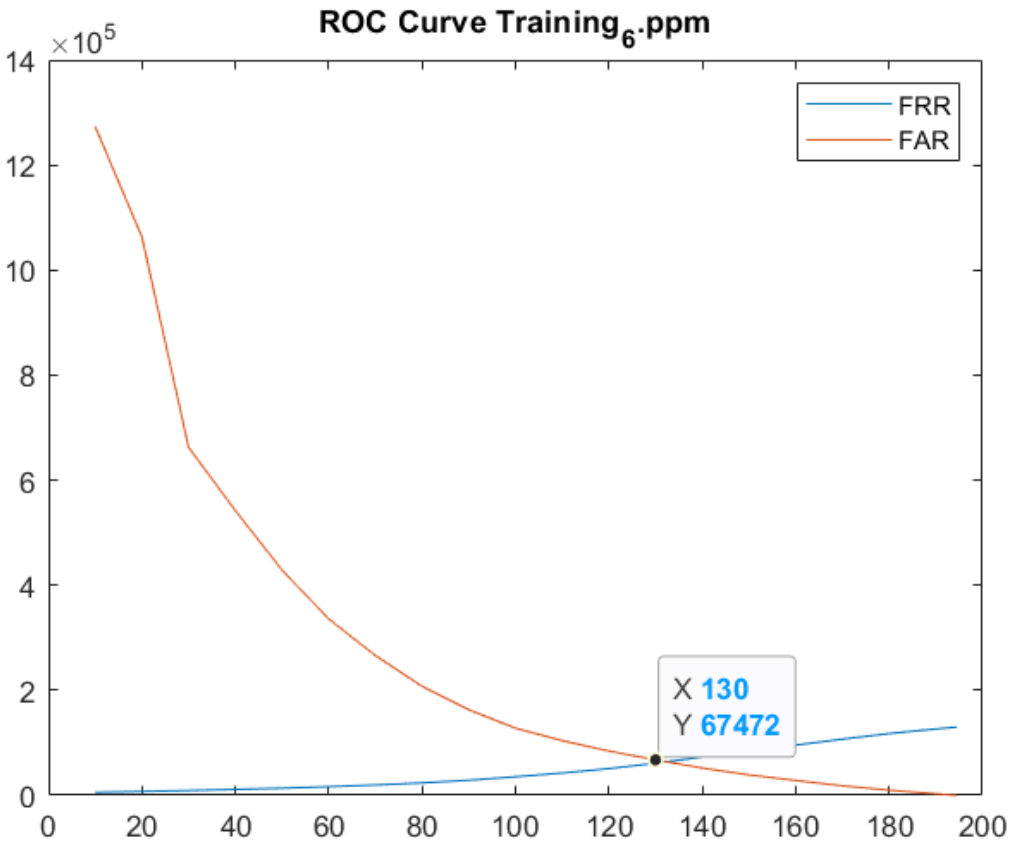


Fig 4: The ROC curve for test image 6 indicates that the optimal threshold for testing image 6 will be 130.

Fig 5: The reference image 6 produced by the experiment 3 classifier with a threshold of 130.

Fig 6: The original image 6 which was used as a testing image for skin color face recognition algorithm.

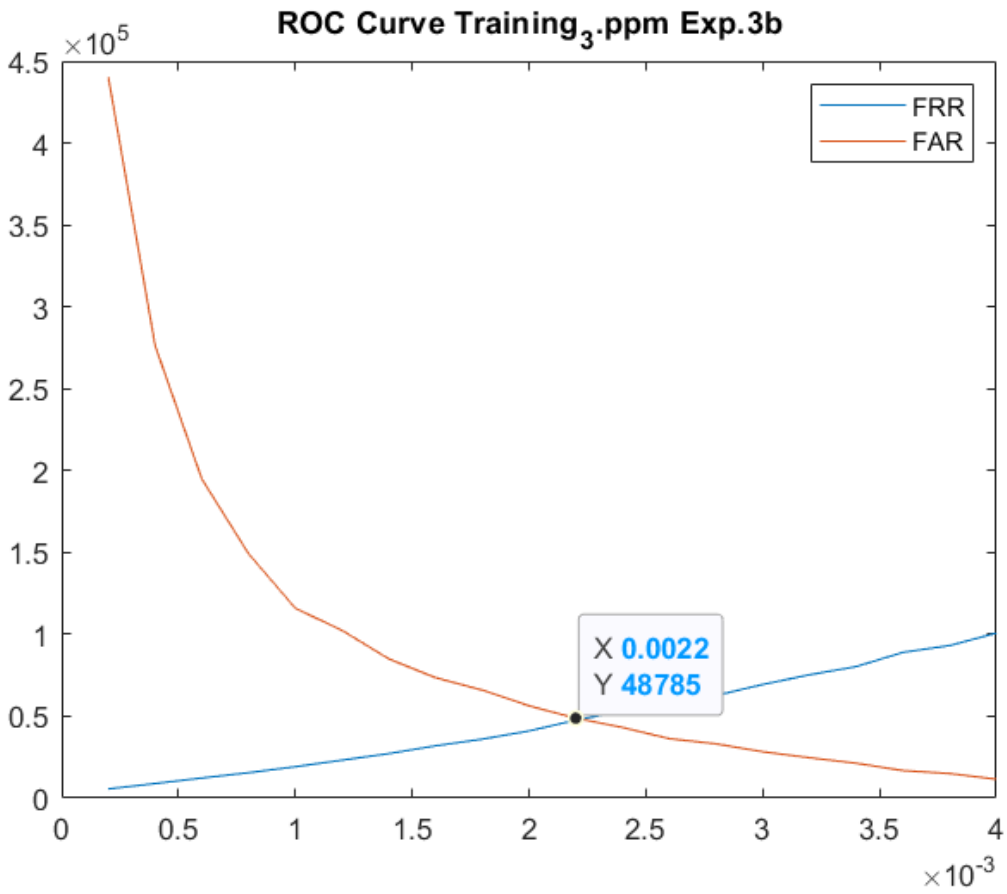


Fig 7: The ROC curve for test image 3 indicates that the optimal threshold for testing image 3 will be 0.0022.

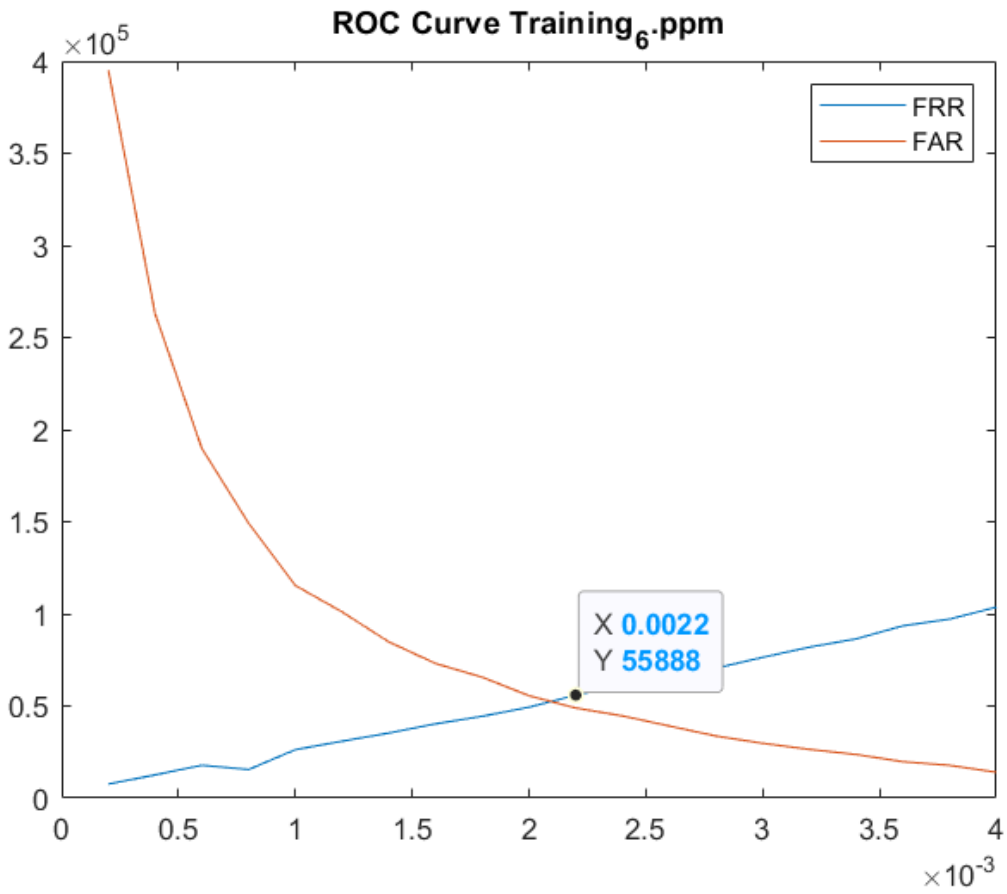
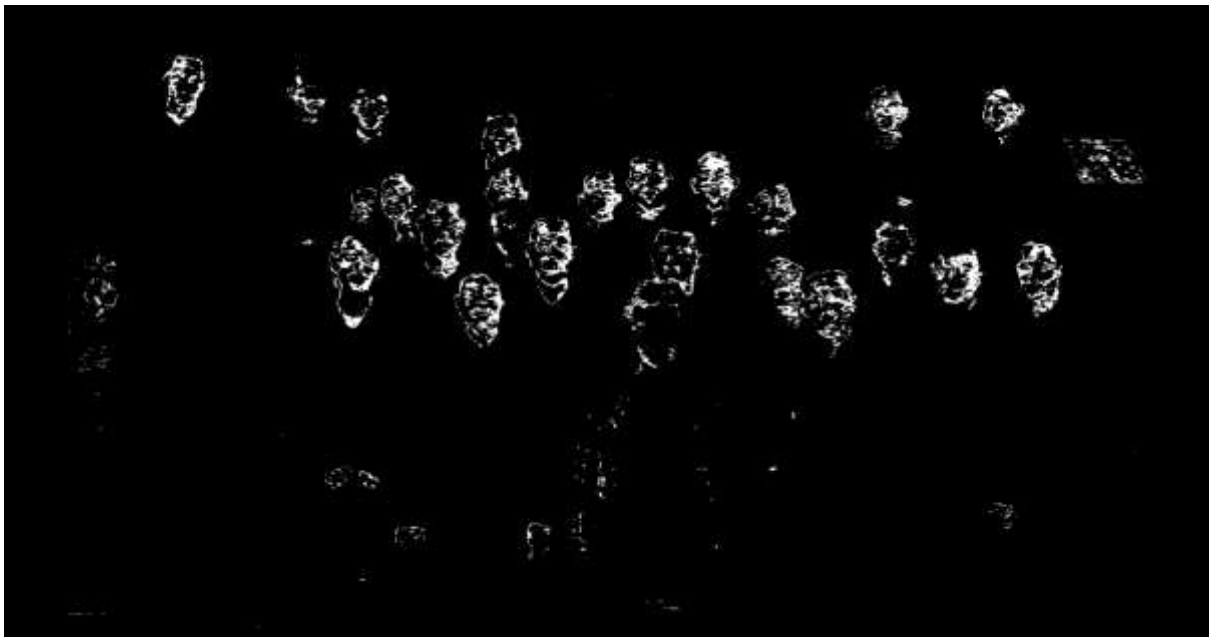


Fig 8: The ROC curve for test image 6 indicates that the optimal threshold for testing image 6 will be around 0.0021 (as the dot didn't want to focus on the dot).

Here are the reference images produced by our classifier using YCbCr color space:



These are closer to the references provided in the assignment than the reference images produced by the chromatic color space.

The misclassification rate for both images can be calculated using the y-axis of the ROC curve charts. The y-axis represents the number of errors from the given ERR threshold. In image 3 and image 6, the error is 68919 and 67472 respectively. The number of pixels in each image is the number of rows multiplied by the number of columns of each image, which is 2,583,552 pixels for both images. Taking the ratio of the errors to the total pixels gives us a misclassification rate of 2 percent for both images. The image classifier designed accurately classifies face-pixels from non-face pixels.