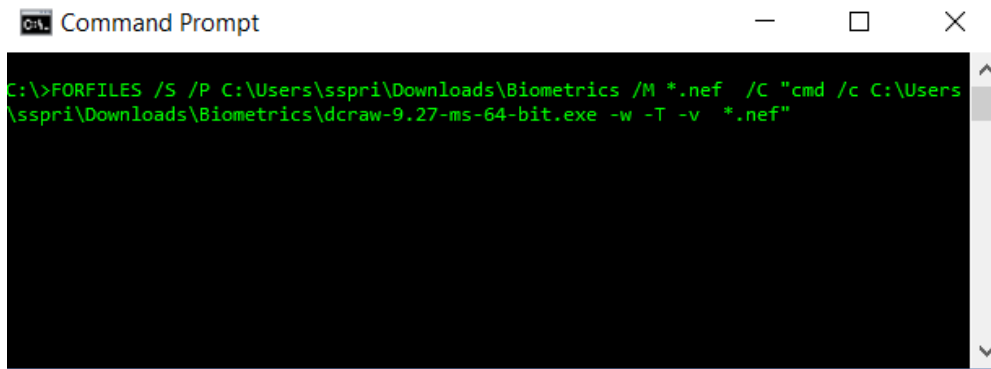


1. DESIGN

To begin, the raw Nikon NEF images were converted to the TIFF format using the *dcraw* executable. The following Windows command was used to recursively convert all of the raw images:



```
C:\>FORFILES /S /P C:\Users\sspri\Downloads\Biometrics /M *.nef /C "cmd /c C:\Users
\sspri\Downloads\Biometrics\dcraw-9.27-ms-64-bit.exe -w -T -v *.nef"
```

Figure 1: dcraw command

For the facial recognition system, the open source 'PCA-based FACE Recognition System' by Amir Omidvarnia was adapted and utilized. This software implements the Eigenface algorithm for authenticating subject images.

The wrapper function *runFacialRecognition.m*, invokes the following subfunctions:

- (1) *generateEigenfaces.m*
- (2) *performCustomRecognitions.m*
- (3) *generateGenuineAndImposterDistributionsFacial.m*
- (4) *generateROC_CurveFacial.m*
- (5) *generateCMC_CurveFacial.m*

First, *generateEigenfaces.m* reads a list gallery (train) images from a text file called *TestFiles.txt* and reads a list of probe (test) images from a text file called *TrainFiles.txt* and stores the two lists internally.

TestFiles.txt	
1	90003d1.tiff
2	90003d2.tiff
3	90003d3.tiff
4	90003d11.tiff
5	90004d1.tiff
6	90004d2.tiff
7	90004d3.tiff
8	90004d11.tiff
9	90006d11.tiff
10	90006d12.tiff
11	90006d13.tiff
12	90006d39.tiff
13	90008d43.tiff
14	90010d19.tiff
15	90012d22.tiff
16	90014d66.tiff
17	90016d20.tiff
18	90018d64.tiff
19	90020d17.tiff
20	90022d138.tiff
21	90024d14.tiff
22	90026d21.tiff
23	90028d18.tiff
24	90030d115.tiff
25	90032d22.tiff
26	90034d22.tiff
27	90036d15.tiff
28	90038d16.tiff
29	90040d19.tiff
30	90042d23.tiff
31	90044d18.tiff
32	90046d14.tiff
33	

TrainFiles.txt	
1	90003d4.tiff
2	90003d5.tiff
3	90003d6.tiff
4	90004d5.tiff
5	90004d6.tiff
6	90004d7.tiff
7	90004d8.tiff
8	90006d13.tiff
9	90006d21.tiff
10	90008d23.tiff
11	90008d24.tiff
12	90010d23.tiff
13	90010d24.tiff
14	90010d40.tiff
15	90012d21.tiff
16	90012d22.tiff
17	90012d23.tiff
18	90014d13.tiff
19	90014d14.tiff
20	90014d15.tiff
21	90016d20.tiff
22	90016d21.tiff
23	90016d22.tiff
24	90016d23.tiff
25	90018d23.tiff
26	90018d24.tiff
27	90022d19.tiff
28	90022d20.tiff
29	90022d21.tiff
30	90024d23.tiff
31	90024d24.tiff
32	90026d14.tiff

Figure 2: TestFiles.txt and TrainFiles.txt

Next, it creates T , a two-dimensional matrix containing all one-dimensional image vectors. T is then input to the function *EigenfaceCore.m*, which returns m , the mean of the training database, *Eigenfaces*, Eigen vectors of the covariance matrix, and A , a matrix of centered image vectors.

Algorithm for computing Eigenfaces:

- “Eigenfacecore.m” - Uses Principle Component Analysis (PCA) to determine the most discriminating features between images of faces.
- This function gets a 2D matrix, containing all training image vectors and returns 3 outputs which are extracted from training database.
- Computes the average face image $m = (1/P) * \sum(T_j's) \quad (j = 1 : P)$
- Computes the difference image for each image in the training set $A_i = T_i - m$
- Merges all centered images
- L is the surrogate of covariance matrix $C = A * A'$.
- Diagonal elements of D are the eigenvalues for both $L = A' * A$ and $C = A * A'$.
- Sort and eliminate eigenvalues :All eigenvalues of matrix L are sorted and those who are less than a specified threshold, are eliminated. So the number of non-zero eigenvectors may be less than $(P-1)$.

- Calculate the eigenvectors of covariance matrix 'C'
- Eigenvectors of covariance matrix C (or so-called "Eigenfaces")
- Can be recovered from L's eigenvectors.

Second, *performCustomRecognitions.m*, obtains, for each test images, a row vector containing its Euclidean distances, its minimum Euclidean distance, and the index of the image corresponding to the lowest Euclidean distance. All of the Euclidean distances are stored in the matrix *euclideanDistances*. To obtain this information, it repeatedly invokes the subfunction *customRecognitions.m* for each individual test image, taking the values of *m*, *A*, and *Eigenfaces* outputted by *generateEigenfaces.m* as inputs.

Algorithm for customRecognition :

- “ customRecognition.m “- This function compares two faces by projecting the images into facespace and measuring the Euclidean distance between them.
- Argument: Image - Path of the input test image
- m- ($M \times N \times 1$) Mean of the training database, which is output of 'EigenfaceCore' function.
- Eigenfaces- ($M \times N \times (P-1)$) Eigen vectors of the covariance matrix of the training database, which is output of 'EigenfaceCore' function.
- A - ($M \times N \times P$) Matrix of centered images vectors, which is output of 'EigenfaceCore' function.
- Projecting centered image vectors into facespace.
- All centered images are projected into facespace by multiplying in Eigenface basis's.
- Projected vector of each face will be its corresponding feature vector.
- Image is supposed to have minimum distance with its corresponding image in the training database.

2. GENUINE AND IMPOSTER DISTRIBUTIONS

Approach:

The genuine and imposter distributions are calculated and plotted by the function *generateGenuineAndImposterDistributionsFacial.m*. Initially, this function determines whether or not a Euclidean distance corresponds to a true match or a false match if the corresponding file names of the test and train files start with the same string. For example, if the test image is 90003d1.tiff and the corresponding train image is 90003d2.tiff, the algorithm would determine that there is a true match because both file names start with “90003d2”. If, however, the test image is 90004d1.tiff and the corresponding train image is 90006d11.tiff, the algorithm would determine that there is a false match because the first file name starts with “90004” and the second starts with “90006”. If there is a true match, the Euclidean distance is considered is placed in the vector *euclideanDistancesGenuine*, which contains the genuine Euclidean distances. If, however, there is a false match, the Euclidean distance is placed in the vector *euclideanDistancesImposter*, which contains the imposter Euclidean distances. Next the imposter Euclidean distances are placed into histogram bins and the number of imposter Euclidean distances for a given range of Euclidean distances are determined. The resulting histogram is converted to a probability distribution by dividing the number of imposter Euclidean distances in every bin by the total number of imposter Euclidean distances. The result is a vector *probabilitiesImposter*, which

contains the imposter probability distribution. The analogous vector *probabilitesGenuine* is calculated using the same algorithm. Finally, the *probabilitiesImposter* and *probabilitesGenuine* are plotted.

Results:

The figure below contains the genuine and imposter distributions outputted by *generateGenuineAndImposterDistributionsFacial.m*.

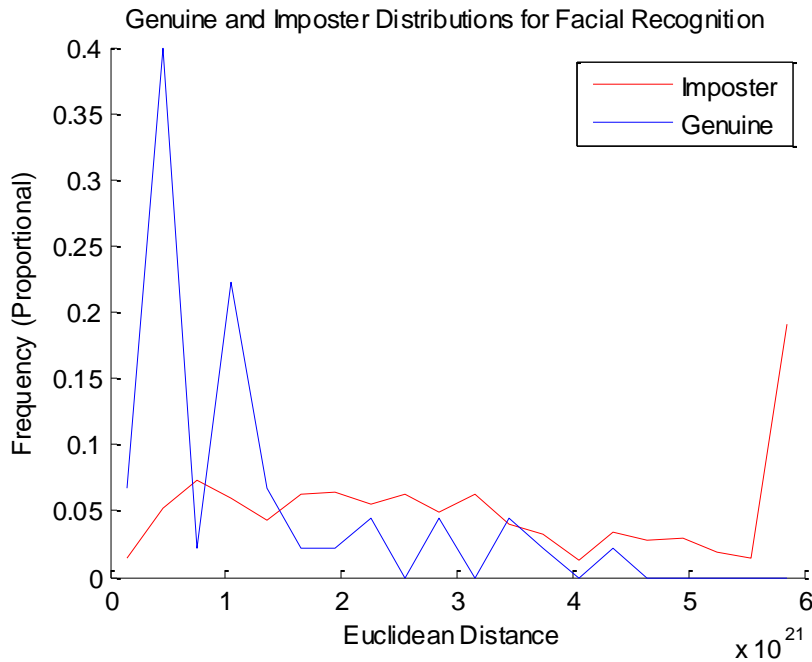


Figure 3: Genuine and Imposter Distributions for the Facial Recognition System

Analysis:

The results strongly indicate that the typical Euclidean distance for the genuine distribution is less than that of the typical Euclidean distance for the imposter distribution as evidenced by the large number of genuine Euclidean distances less than 2.0×10^{21} and the large number of imposter Euclidean distances greater than 5.0×10^{21} . In particular, all the genuine Euclidean distances are less than 5.0×10^{21} .

3. ROC CURVE

Approach:

The preceding function *generateGenuineAndImposterDistributionsFacial.m* calculates the probability functions of the imposter and genuine distributions and stores them in the vectors *probabilitiesImposter* and *probabilitiesGenuine*.

The *generateROC_CurveFacial.m* function generates and plots the ROC curve. For a given threshold, the false match rate is determined by taking the cumulative sum of *probabilitiesImposter* up to that point.

By analogy, the true match rate is determined by taking the cumulative sum of *probabilitiesGenuine* up to that point.

Results:

The figure below contains the genuine and imposter distributions outputted by *generateROC_CurveFacial.m*.

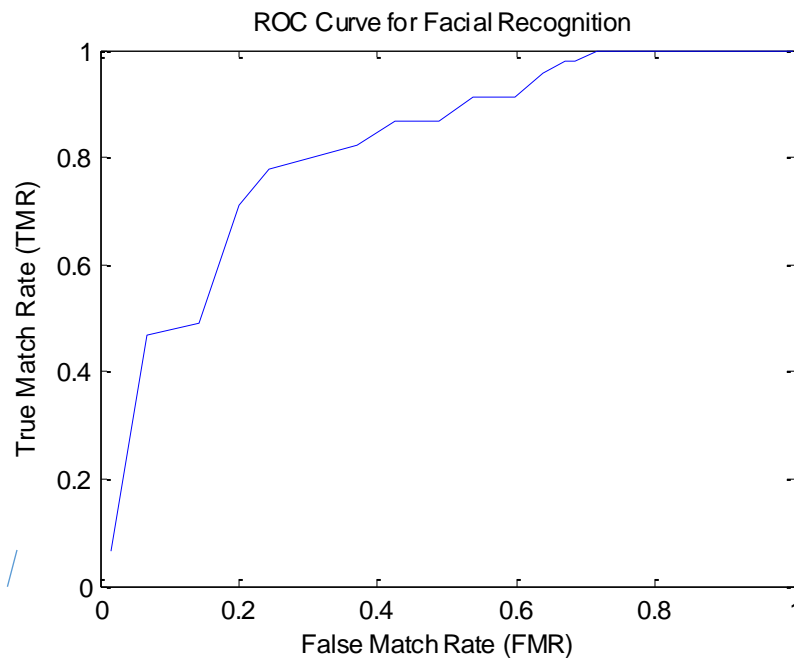


Figure 4: ROC Curve for the Facial Recognition System

Analysis:

The ROC curve indicates that the optimal threshold will result in a False Match Rate of about 0.2 and a True Match Rate of about 0.8. Overall, the ROC seems to indicate the plausibility of the biometric system as it appears to be markedly different from the diagonal line from (0,0) to (1,1), the worst-case scenario.

4. CMC CURVE

Approach:

The function *generateCMC_CurveFacial* calculates the rank of the correct identity for every given test image. For each test image, the Euclidean distance between that test image and every train image in the gallery is recorded. The Euclidean distances and subject name (ex. "90003") for the train images are stored in an array called *matchInfo*. Next *matchInfo* is sorted by Euclidean Distance. The subject name of the test image is compared against the subject names of the entries in *matchInfo*. As the comparisons are made, a vector called *currentIdentities* stores the distinct elements that are encountered up to the

point where the correct element is found. After the correct element is found the length of *currentIdentities* is computed and is stored as *currentRankOfCorrectIdentity*. The ranks of the correct identity for each test image are stored in an array called *ranksOfCorrectIdentities*. Next, the function converts *ranksOfCorrectIdentities* into *rank_t_IdentificationRates*, an array of the identification rates starting from $k=1$. Finally, *rank_t_IdentificationRates* are plotted.

Results:

The figure below contains the CMC curve outputted by *generateCMC_CurveFacial*.

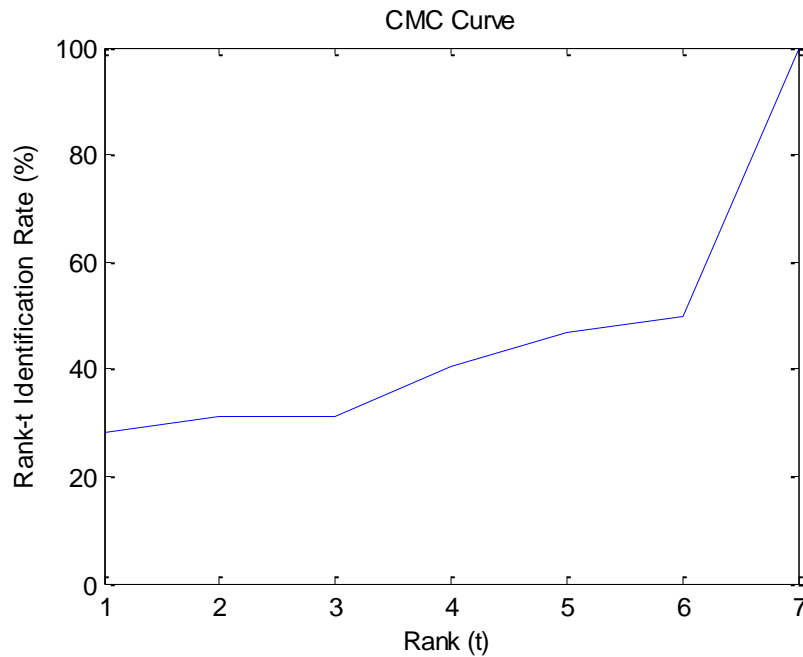


Figure 5: CMC Curve for the Facial Recognition System

Analysis:

The CMC curve reveals that the rank-1 identification rate is approximately 30%, meaning that the correct identity is returned first about a third of the time. Ideally, this figure would be higher. More problematic is the finding that there are a large number of subjects for which the correct identity is returned 7th. Looking at the data, it appears that these subjects correspond to faces with a small number of images in the gallery. Therefore, it appears that that the biometric system could be improved by adding more images into the gallery.