Project 3

CSE 402 - Biometrics and Pattern Recognition
Instructor: Dr. Arun Ross
Due Date: Dec 10, 2022 (11:00pm ET)
Total Points: 100

Note:

1. **You are permitted to discuss the following questions with others in the class. However, you *must* write up your *own* solutions to these questions. Any indication to the contrary will be considered an act of academic dishonesty. Copying from *ANY source* constitutes academic dishonesty.**
2. **A neatly typed report is expected (alternately, you can neatly handwrite the report and then scan it). The report, in PDF format, must be uploaded in D2L by December 10, 11:00 pm. Late submissions will not be graded. In your submission, please include the names of individuals you discussed this assignment with and the list of external resources (e.g., websites, other books, articles, etc.) that you used to complete the assignment (if any).**
3. **When solving equations or reducing expressions you must explicitly show every step in your computation and/or include the code that was used to perform the computation. Missing steps or code will lead to a deduction of points. Including the code without including the outputs will lead to deduction of points.**
4. **Code developed as part of this assignment must be (a) included as an appendix to your report or inline with your solution (in the report), and also (b) archived in a single zip file and uploaded in D2L. Including the code without the outputs or including the outputs without the code will result in deduction of points.**
5. **Please submit the report (PDF) and the code (Zip file) as two separate files in D2L.**

---

1. Select one of your own frontal face photo ("selfie"). Crop this photo so that only the face is seen. Next, rescale the cropped face photo to a size of 100 × 100. Finally, convert the cropped and scaled photo from color to grayscale. Use this 100 × 100 grayscale image to do the following.

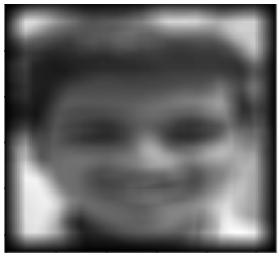
   (a) [5 points] Apply the following operations to the grayscale image: (a) increase brightness; (b) improve contrast; (c) smoothen the image using a Gaussian Filter. You can use any set of parameter values for each of these operations. Display the original grayscale image (say, F_o) along with the 3 modified images (say, F_B, F_C , F_G) in your report.

   Ans) Grayscale image:

Increased brightness:

Improved Contrast:

Smoothen the image using a Gaussian Filter:



(b) [10 points] For each of the 4 images, compute the corresponding LBP image with $P=8$, $R=1$. You can set the border pixels of each LBP image to 0. Display the 4 LBP images (say, L_0, L_B, L_C , L_G) in your report.

Ans) L_0:

L_B:

L_C:

L_G:

(c) [5 points] Write a program that compares L_0 with each of L_B, L_C and L_G (so there will be 3 comparison scores). This must be done as follows: compare each pixel in one image with the corresponding pixel in the other image, take the absolute difference between the two, sum all the absolute differences, and then divide this sum by the total number of pixels. Report the 3 scores.

Ans)

```
L_0_vs_L_b = pixel_comparator(L_0,L_b)
L_0_vs_L_b
```

```
19.0068
```

```
L_0_vs_L_c = pixel_comparator(L_0,L_c)
L_0_vs_L_c
```

```
53.7961
```

```
L_0_vs_L_g = pixel_comparator(L_0,L_g)
L_0_vs_L_g
```

```
92.7782
```

(d) [5 points] Also report the scores obtained when comparing F_0 with each of F_B, F_C and F_G. Use the same procedure as above to compute the comparison scores.

Ans)

```
F_0_vs_F_b = pixel_comparator(F_0,F_b)
F_0_vs_F_b
```

```
/opt/anaconda3/lib/python3.7/site-packages
erflow encountered in ubyte_scalars
  if __name__ == '__main__':
```

```
212.9101
```

```
F_0_vs_F_c = pixel_comparator(F_0,F_c)
F_0_vs_F_c
```

```
/opt/anaconda3/lib/python3.7/site-packages
erflow encountered in ubyte_scalars
  if __name__ == '__main__':
```

```
107.8967
```

```
F_0_vs_F_g = pixel_comparator(F_0,F_g)
F_0_vs_F_g
```

```
34.11228765432111
```

(e) [5 points] Discuss your observations. In particular, does applying the LBP operator reduce

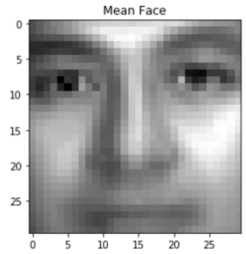
the intra-class variation due to changes in pixel intensity?

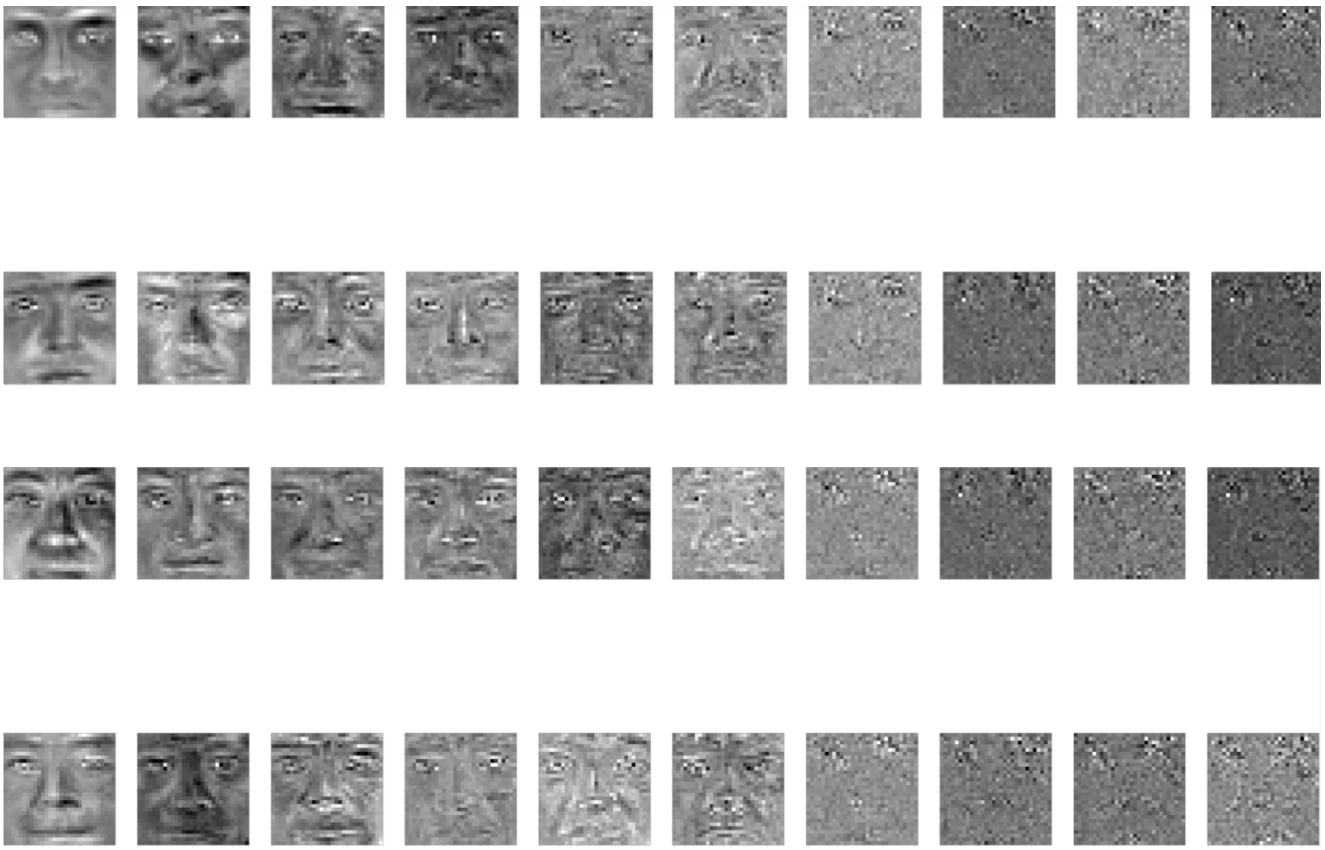
Ans) The intra-class variances are really lessened by the LBP operator. An illustration would be to contrast a grayscale image with a brighter one. The F_0 and F_B 's differential score is 212.9101. The difference between L_0 and L_B, on the other hand, is 19.0068. This considerable difference demonstrates that LBP greatly lowers intra-class variation. LBP lowers intra-class variance brought on by variations in pixel intensity. It does this by encoding the local structure around each pixel, which may be very helpful in lowering intra-class variance.

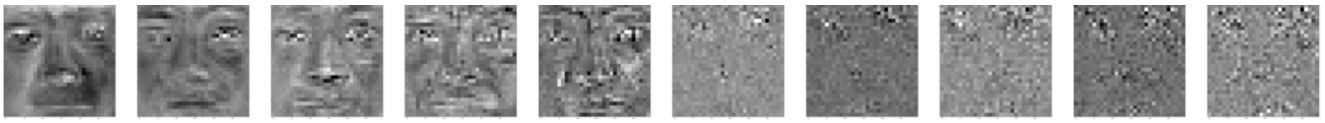
2. You are given a set of 50 face images pertaining to 10 different subjects (5 images per subject).

(a) [10 points] Compute the eigen-faces (i.e., basis images) using 30 face images (first 3 images per subject). **Show** the mean face as well as the eigen-faces corresponding to the top 50 eigen-values.

Ans)







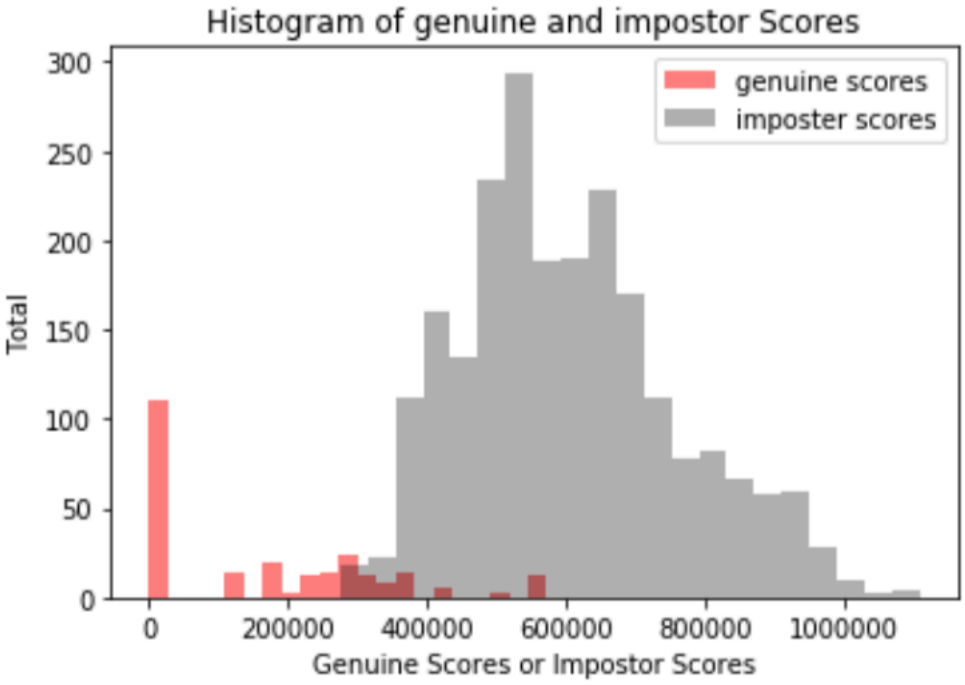
(b) [10 points] Using the mean face and the top 50 eigen-faces (i.e., eigen-vectors), compute the eigen-coefficients (i.e., the 50-dimensional feature vector) for each of the 50 images in the dataset, including the 30 face images you had used in 2a.

Ans)

```
[[[-1.65062019e+02  5.59122139e+01  3.87395719e+02 ... -1.77635684e-14
   -2.13162821e-14 -2.13162821e-14]]

 [[-2.38963202e+02  3.24997001e+01  3.39177800e+02 ... -3.01980663e-14
   -1.77635684e-14 -1.77635684e-14]]

 [[-2.33919502e+02  6.66753226e+01  2.12357545e+02 ... -2.30926389e-14
   -3.19744231e-14 -3.19744231e-14]]

 ...

 [[-2.73088075e+01 -1.27073964e+02 -2.17346337e+02 ... -5.32907052e-15
    2.48689958e-14  2.48689958e-14]]

 [[-2.73088075e+01 -1.27073964e+02 -2.17346337e+02 ... -5.32907052e-15
    2.48689958e-14  2.48689958e-14]]

 [[-2.73088075e+01 -1.27073964e+02 -2.17346337e+02 ... -5.32907052e-15
    2.48689958e-14  2.48689958e-14]]]
```

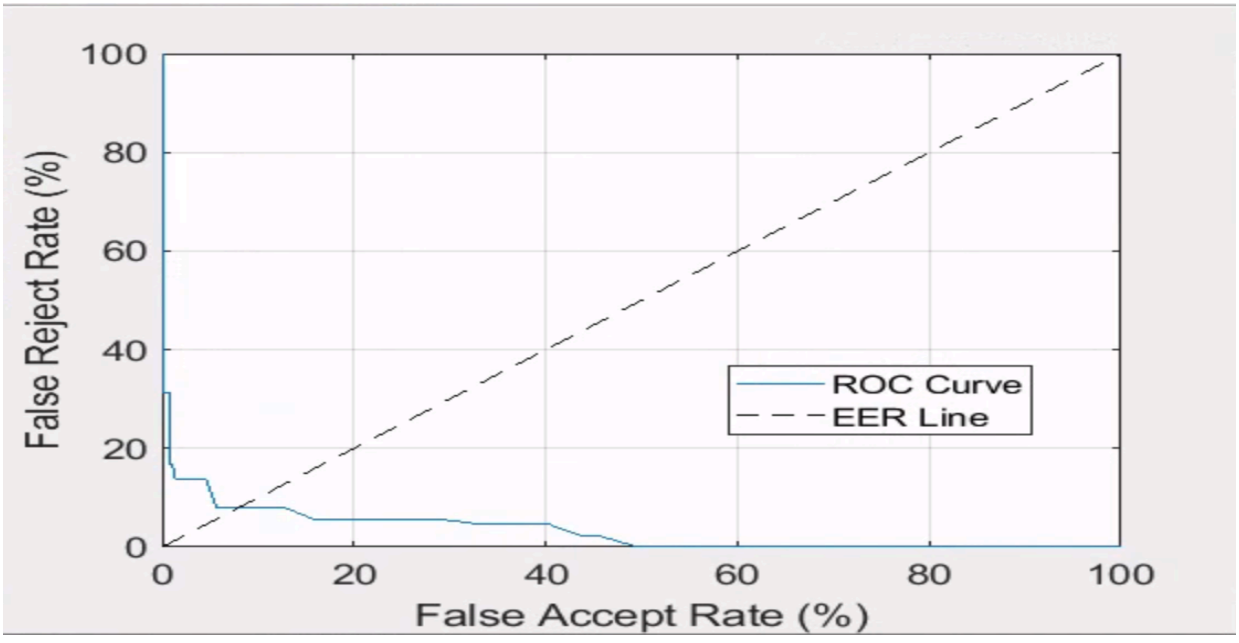
(c) [10 points] Generate genuine scores and impostor scores by computing the Euclidean distance between the feature vectors of every pair of face images. **Plot** the histograms of genuine and impostor scores in the same graph. Use a different color for each histogram.

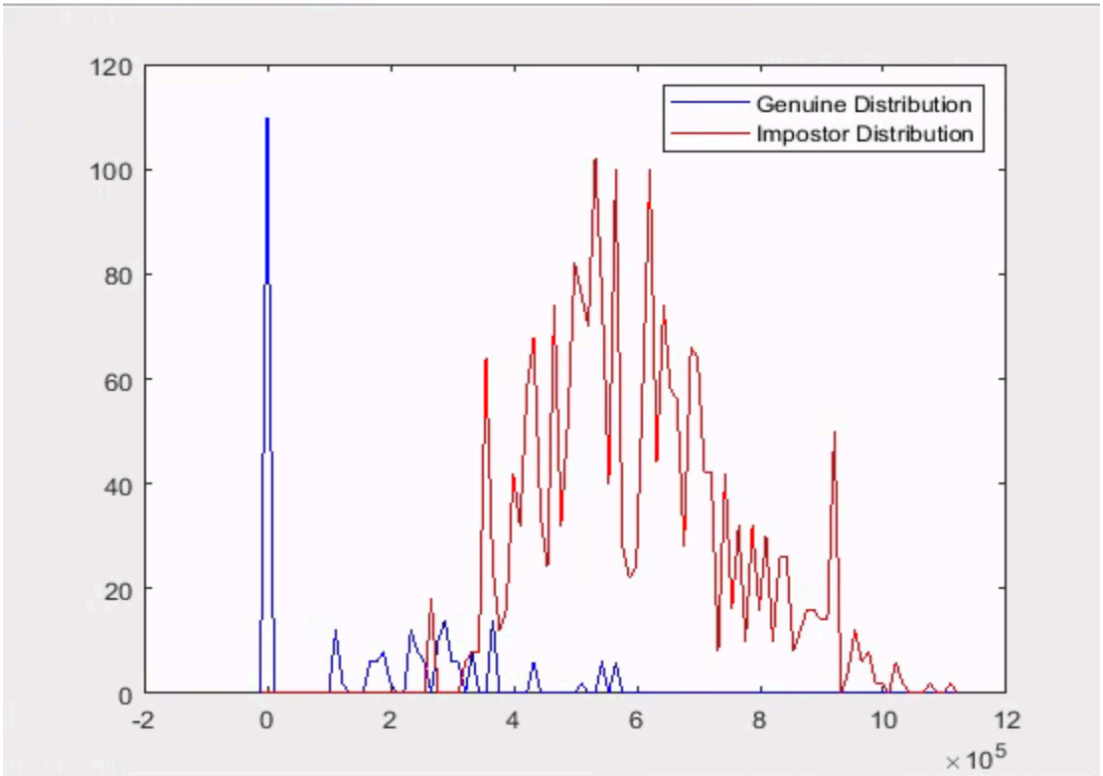
Ans)



(d) [5 points] **Plot** the ROC curve summarizing the matching performance using these scores. You can use the matlab code available here to plot the ROC curve. The code can be invoked as **roc(gen, imp, 'd')**. Here, **gen** and **imp** are the set of genuine and impostor scores, respectively. 'd' denotes that the scores are distance (or dissimilarity) scores.

Ans)



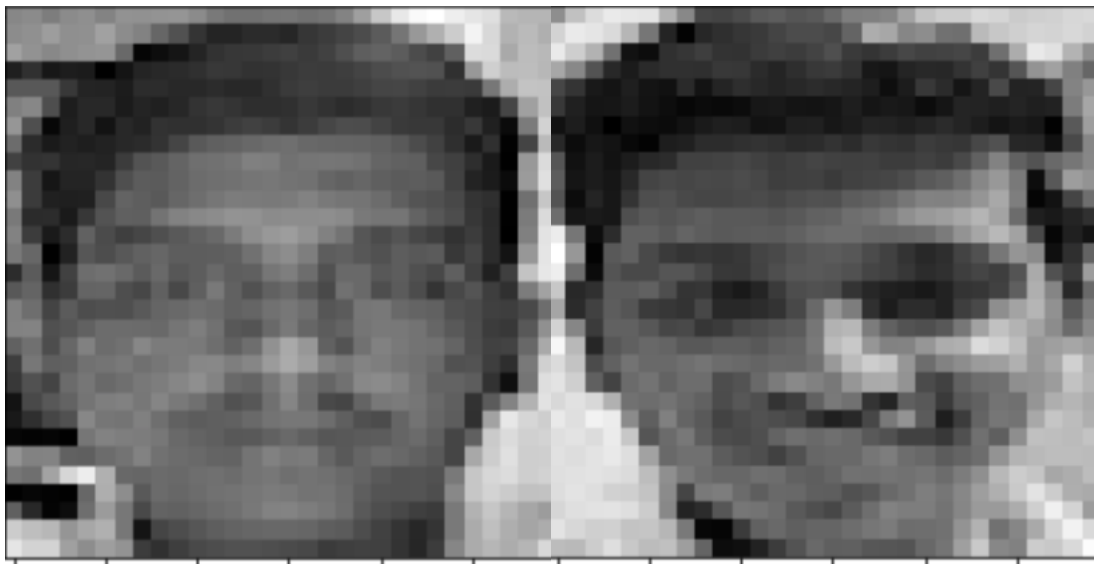


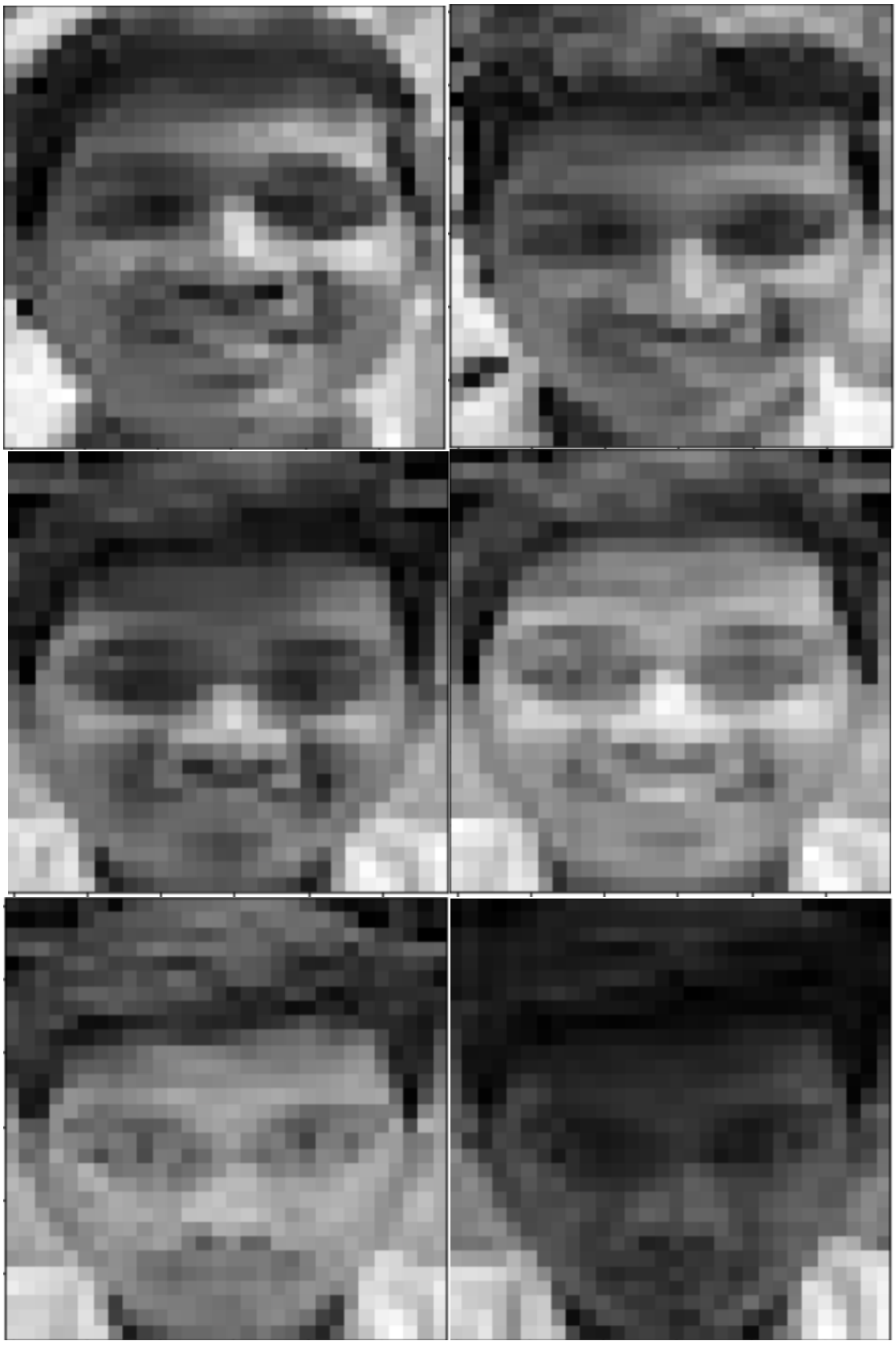
(e) [10 points] Repeat the above after selecting the top (i) 10, (ii) 20, (iii) 40 eigen-faces. **Plot** the ROC curves for each case. **Comment** on the change in matching performance as you vary the number of eigen-faces used to generate the feature vector.

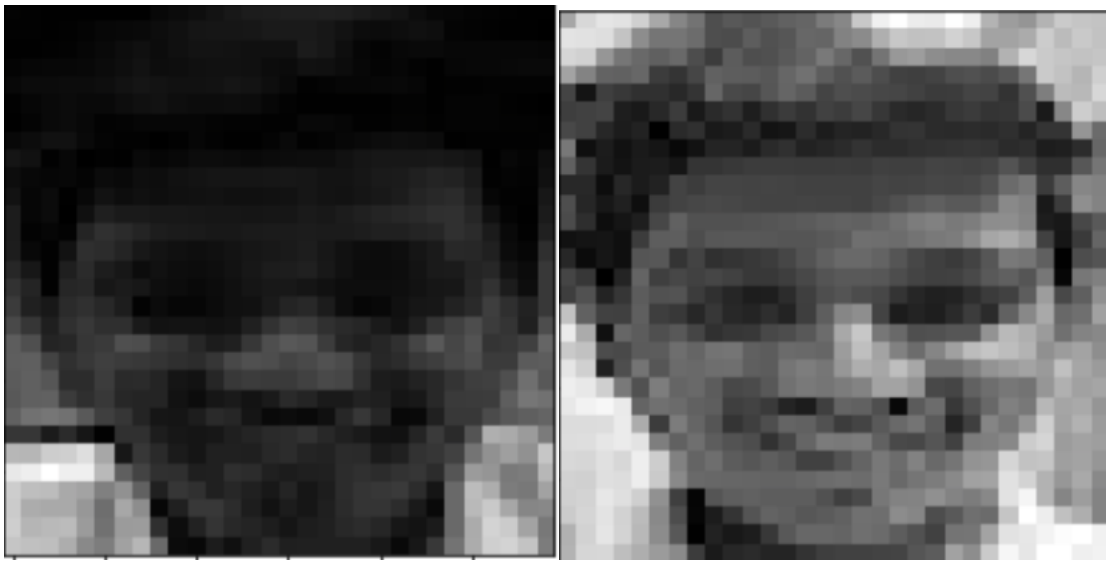
3. Select 10 of your own **frontal** face photos ("selfies"). Crop each photo so that only the face is seen. Next, rescale each cropped face photo to a size of 30×30. Finally, convert the cropped and scaled photos from color to grayscale.

   - [5 points] Display the 10 grayscale images.

   Ans)







- [5 points] Compute the eigen-coefficients of the 10 faces using the top 25 eigen-faces (eigen-vectors) computed in2a.

Ans)

```
[array([[ 561.34436746,  534.76905278,  -25.72407189, -427.66847029,
         794.01096208,  667.3253231 ,  216.89865483, -128.31072475,
          78.46172945,   56.48423781,  -15.88645646,  -15.88645646,
          21.03180401,    1.11464486,    1.11464486,    2.30889196,
           2.30889196,  -14.64173439,  -14.64173439,   56.63606648,
          59.79960625,   72.83849433,   72.83849433,   49.0285998 ,
          49.0285998 ,  -88.47647763,    4.15950647,    4.15950647,
         -23.18613829,  -23.18613829,   26.32133481,   26.32133481,
          -1.57998777,   -1.57998777,  -31.90273494,  -31.90273494,
         -37.71291452,    3.75672367,    3.75672367,  -18.3830851 ,
         -18.3830851 ,   -1.50616744,   -1.50616744,  -55.10781024,
         -34.18236618,  -34.18236618,  -51.90186135,  -51.90186135,
          21.7802648 ,   21.7802648 ]]), array([[ 2.05090510e+02,  4.02070533e+02, -1.
33452199e+03,
         1.73727249e+02,  3.91200767e+02,  7.33447790e+02,
        -1.68541169e+02,  3.19822942e+02,  1.02429905e+02,
         5.19956372e+01, -1.58864565e+01, -1.58864565e+01,
         2.10318040e+01,  1.11464486e+00,  1.11464486e+00,
         2.30889196e+00,  2.30889196e+00, -1.46417344e+01,
        -1.46417344e+01,  5.66360665e+01,  5.97996063e+01,
         7.28384943e+01,  7.28384943e+01,  4.90285998e+01,
                  4.90285998e+01, -8.84764776e+01,  4.15950647e+00,
                  4.15950647e+00, -2.31861383e+01, -2.31861383e+01,
                  2.63213348e+01,  2.63213348e+01, -1.57998777e+00,
                 -1.57998777e+00, -3.19027349e+01, -3.19027349e+01,
                 -3.77129145e+01,  3.75672367e+00,  3.75672367e+00,
                 -1.83830851e+01, -1.83830851e+01, -1.50616744e+00,
                 -1.50616744e+00, -5.51078102e+01, -3.41823662e+01,
                 -3.41823662e+01, -5.19018614e+01, -5.19018614e+01,
                  2.17802648e+01,  2.17802648e+01]]), array([[ 266.40921252,  482.12099196, -49
        3.72512246, 1025.39157982,
                  592.67114774,  312.59463837,  319.52471927,  -56.51842205,
                   94.92966867,   62.30893707,  -15.88645646,  -15.88645646,
                   21.03180401,    1.11464486,    1.11464486,    2.30889196,
                    2.30889196,  -14.64173439,  -14.64173439,   56.63606648,
                   59.79960625,   72.83849433,   72.83849433,   49.0285998 ,
                   49.0285998 ,  -88.47647763,    4.15950647,    4.15950647,
                  -23.18613829,  -23.18613829,   26.32133481,   26.32133481,
                   -1.57998777,   -1.57998777,  -31.90273494,  -31.90273494,
                  -37.71291452,    3.75672367,    3.75672367,  -18.3830851 ,
                  -18.3830851 ,   -1.50616744,   -1.50616744,  -55.10781024,
                  -34.18236618,  -34.18236618,  -51.90186135,  -51.90186135,
```

```
          21.7802648 ,   21.7802648 ]]), array([[ 2.87017024e+02, -9.50464950e+01, -1.
11942192e+03,
        -9.04984153e+01, -6.48061455e+01,  5.05896411e+02,
         6.62829834e+02,  5.46388198e+01,  2.66993081e+01,
         7.66744627e+01, -1.58864565e+01, -1.58864565e+01,
         2.10318040e+01,  1.11464486e+00,  1.11464486e+00,
         2.30889196e+00,  2.30889196e+00, -1.46417344e+01,
        -1.46417344e+01,  5.66360665e+01,  5.97996063e+01,
         7.28384943e+01,  7.28384943e+01,  4.90285998e+01,
         4.90285998e+01, -8.84764776e+01,  4.15950647e+00,
         4.15950647e+00, -2.31861383e+01, -2.31861383e+01,
         2.63213348e+01,  2.63213348e+01, -1.57998777e+00,
        -1.57998777e+00, -3.19027349e+01, -3.19027349e+01,
        -3.77129145e+01,  3.75672367e+00,  3.75672367e+00,
        -1.83830851e+01, -1.83830851e+01, -1.50616744e+00,
        -1.50616744e+00, -5.51078102e+01, -3.41823662e+01,
        -3.41823662e+01, -5.19018614e+01, -5.19018614e+01,
         2.17802648e+01,  2.17802648e+01]]), array([[ 808.86394749, -664.58054228, -85
5.8681215 ,  273.66881833,
         270.75228486,  760.11713066,  105.94180001, -350.22046855,
         367.39253854,   63.34991577,  -15.88645646,  -15.88645646,
          21.03180401,    1.11464486,    1.11464486,    2.30889196,
           2.30889196,  -14.64173439,  -14.64173439,   56.63606648,
          59.79960625,   72.83849433,   72.83849433,   49.0285998 ,
          49.0285998 ,  -88.47647763,    4.15950647,    4.15950647,
         -23.18613829,  -23.18613829,   26.32133481,   26.32133481,
          -1.57998777,   -1.57998777,  -31.90273494,  -31.90273494,
         -37.71291452,    3.75672367,    3.75672367,  -18.3830851 ,
         -18.3830851 ,   -1.50616744,   -1.50616744,  -55.10781024,
         -34.18236618,  -34.18236618,  -51.90186135,  -51.90186135,
          21.7802648 ,   21.7802648 ]]), array([[ 126.47768283, -675.42310659, -220.78
51291 ,  367.42790036,
          64.69494522,  711.24754472,  -12.4776206 ,  -96.60923672,
        -195.46609004,   -5.32130392,  -15.88645646,  -15.88645646,
          21.03180401,    1.11464486,    1.11464486,    2.30889196,
           2.30889196,  -14.64173439,  -14.64173439,   56.63606648,
          59.79960625,   72.83849433,   72.83849433,   49.0285998 ,
          49.0285998 ,  -88.47647763,    4.15950647,    4.15950647,
         -23.18613829,  -23.18613829,   26.32133481,   26.32133481,
          -1.57998777,   -1.57998777,  -31.90273494,  -31.90273494,
         -37.71291452,    3.75672367,    3.75672367,  -18.3830851 ,
         -18.3830851 ,   -1.50616744,   -1.50616744,  -55.10781024,
         -34.18236618,  -34.18236618,  -51.90186135,  -51.90186135,
          21.7802648 ,   21.7802648 ]]), array([[ 282.37593966, -838.05489155, -523.23
204883,  -58.2920197 ,
         478.81636671,  -21.86127008,   16.79941488,  129.13186146,
         129.20582102,  143.17759279,  -15.88645646,  -15.88645646,
          21.03180401,    1.11464486,    1.11464486,    2.30889196,
           2.30889196,  -14.64173439,  -14.64173439,   56.63606648,
          59.79960625,   72.83849433,   72.83849433,   49.0285998 ,
          49.0285998 ,  -88.47647763,    4.15950647,    4.15950647,
         -23.18613829,  -23.18613829,   26.32133481,   26.32133481,
          -1.57998777,   -1.57998777,  -31.90273494,  -31.90273494,
         -37.71291452,    3.75672367,    3.75672367,  -18.3830851 ,
         -18.3830851 ,   -1.50616744,   -1.50616744,  -55.10781024,
         -34.18236618,  -34.18236618,  -51.90186135,  -51.90186135,
          21.7802648 ,   21.7802648 ]]), array([[ 1.59493867e+03, -6.51396398e+02, -1.
22827139e+03,
         1.34509420e+02,  9.58474729e+02,  4.63371403e+02,
         2.72177601e+02, -5.93700566e+01,  6.51749664e+00,
        -1.73392344e+02, -1.58864565e+01, -1.58864565e+01,
         2.10318040e+01,  1.11464486e+00,  1.11464486e+00,
         2.30889196e+00,  2.30889196e+00, -1.46417344e+01,
```

```
                -1.46417344e+01,  5.66360665e+01,  5.97996063e+01,
                 7.28384943e+01,  7.28384943e+01,  4.90285998e+01,
                 4.90285998e+01, -8.84764776e+01,  4.15950647e+00,
                 4.15950647e+00, -2.31861383e+01, -2.31861383e+01,
                 2.63213348e+01,  2.63213348e+01, -1.57998777e+00,
                -1.57998777e+00, -3.19027349e+01, -3.19027349e+01,
                -3.77129145e+01,  3.75672367e+00,  3.75672367e+00,
                -1.83830851e+01, -1.83830851e+01, -1.50616744e+00,
                -1.50616744e+00, -5.51078102e+01, -3.41823662e+01,
                -3.41823662e+01, -5.19018614e+01, -5.19018614e+01,
                 2.17802648e+01,  2.17802648e+01]]), array([[ 2.38884608e+03, -4.49252702e+02,
        -1.16829573e+03,
                 2.55916282e+02,  8.33600125e+02,  6.81935747e+02,
                 2.05325471e+02, -1.55762638e+02, -9.57308195e+01,
                 2.93923131e+02, -1.58864565e+01, -1.58864565e+01,
                 2.10318040e+01,  1.11464486e+00,  1.11464486e+00,
                 2.30889196e+00,  2.30889196e+00, -1.46417344e+01,
                -1.46417344e+01,  5.66360665e+01,  5.97996063e+01,
                 7.28384943e+01,  7.28384943e+01,  4.90285998e+01,
                 4.90285998e+01, -8.84764776e+01,  4.15950647e+00,
                 4.15950647e+00, -2.31861383e+01, -2.31861383e+01,
                 2.63213348e+01,  2.63213348e+01, -1.57998777e+00,
                -1.57998777e+00, -3.19027349e+01, -3.19027349e+01,
                -3.77129145e+01,  3.75672367e+00,  3.75672367e+00,
                -1.83830851e+01, -1.83830851e+01, -1.50616744e+00,
                -1.50616744e+00, -5.51078102e+01, -3.41823662e+01,
                -3.41823662e+01, -5.19018614e+01, -5.19018614e+01,
                 2.17802648e+01,  2.17802648e+01]]), array([[ 9.99201037e+01,  4.52881371e+02,
        -1.30974335e+03,
                -1.25943931e+01,  2.62327185e+02,  1.94625534e+02,
                -4.08694945e+01, -4.69778733e+02, -4.42417978e+01,
                 4.23294168e+01, -1.58864565e+01, -1.58864565e+01,
                 2.10318040e+01,  1.11464486e+00,  1.11464486e+00,
                 2.30889196e+00,  2.30889196e+00, -1.46417344e+01,
                -1.46417344e+01,  5.66360665e+01,  5.97996063e+01,
                 7.28384943e+01,  7.28384943e+01,  4.90285998e+01,
                 4.90285998e+01, -8.84764776e+01,  4.15950647e+00,
                 4.15950647e+00, -2.31861383e+01, -2.31861383e+01,
                 2.63213348e+01,  2.63213348e+01, -1.57998777e+00,
                -1.57998777e+00, -3.19027349e+01, -3.19027349e+01,
                -3.77129145e+01,  3.75672367e+00,  3.75672367e+00,
                -1.83830851e+01, -1.83830851e+01, -1.50616744e+00,
                -1.50616744e+00, -5.51078102e+01, -3.41823662e+01,
-3.41823662e+01, -5.19018614e+01, -5.19018614e+01,
 2.17802648e+01,  2.17802648e+01]])]
```

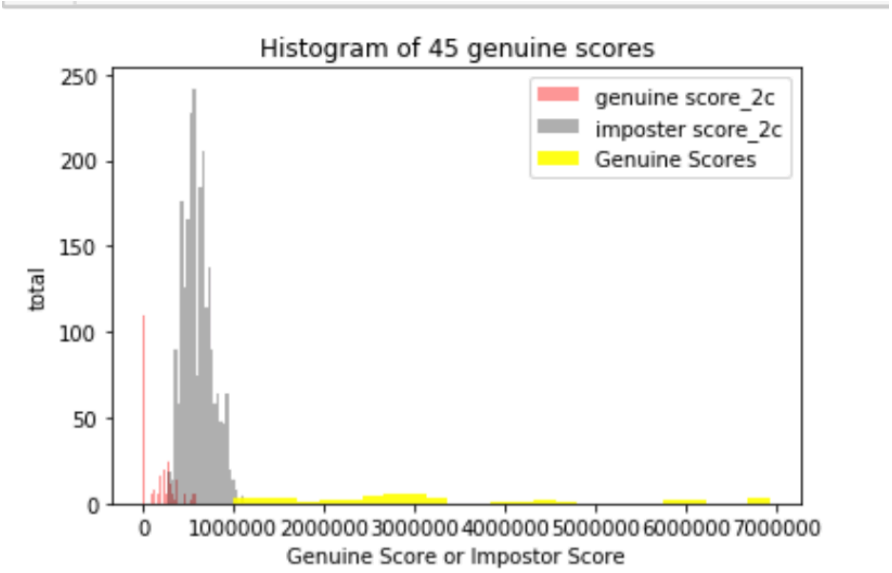
- [5 points] Compute the genuine scores between every pair of faces (there will be 45 genuine scores).

Ans)

```
genuine scores =  [2735764.999999998, 2602529.9999999986, 2780819.999999999, 3108211.9
99999998, 2990206.9999999986, 3037154.9999999986, 4371942.999999995, 6170207.999999997
, 2745044.0, 2040150.9999999984, 1397486.9999999986, 2351237.000000001, 2842000.000000
0037, 2913822.0000000023, 3847058.0, 6190051.0, 1017286.999999999, 2574972.000000001,
2815439.999999999, 2505320.999999999, 3177007.0000000005, 4605248.999999999, 6726679.9
99999999, 2215354.0000000037, 1566284.0000000023, 1973791.0000000035, 1920451.00000000
12, 3360121.000000001, 5819428.000000001, 1357846.0000000019, 1322605.0, 1483739.00000
00014, 1635663.0000000012, 3278884.0000000014, 2564856.0000000023, 1211690.0000000028,
4237980.000000001, 6822181.0, 3075181.0000000037, 2973457.999999998, 5914446.999999999
, 2816503.0000000023, 995451.0000000002, 4355179.000000002, 6934126.000000004]

Total number of genuine scores =  45
```

- [5 points] Plot the histogram of 45 genuine scores on the same graph as2c. For this part, you can re-plot the histograms from the previous question and use them here.

Ans)



- [5 points] Comment on the accuracy of the face matcher.

Ans) The PCA model served as the foundation for this face matcher. The new genuine scores significantly overlap with the Q2 genuine score. Having some of the face real scores fall into the impostor score range is an issue. Therefore, the threshold where the False Match Rate is low has an issue. The model was also trained using a small sample of input data. The model was only trained on 50 photos in total. So, while this model may be utilized for other reasons, it cannot be used for learning. A more intricate PCA model must be created.

4. **[Bonus Question: 10 Points]** Select 10 of your own **non-frontal** face photos (5 right profile and 5 left profile). Crop each photo so that only the face/head is seen. Next, rescale each cropped face photo to a size of 30×30. Finally, convert the cropped and scaled photos from color to grayscale.

- Display the 10 grayscale images.

Ans)

- Compute the eigen-coefficients of the 10 faces using the top 25 eigen-faces (eigen-vectors) computed in2a.

Ans)

```
[array([[1161.64669284,  967.89280501,  -99.55454012,  753.88389404,
        -354.30152531,  -73.37412875,  513.96036741,  209.34663384,
        -279.48492395,  -40.18947441,  -33.92815358,  -45.34747203,
         -45.34747203,   12.4568171 ,   12.4568171 ,   77.52378142,
          77.52378142,   -1.56846456,   -1.56846456,   -5.61804295,
         -18.99325624,  -18.99325624,   25.66238177,   25.66238177,
          58.73964939,   58.73964939,   36.84463211,   36.84463211,
          37.51281242,   37.51281242,    5.50985845,    5.50985845,
          18.3107492 ,  -31.23114162,  -31.23114162,   62.27484438,
          62.27484438,  -12.77530046,   14.44069482,   14.44069482,
          55.8102783 ,   55.8102783 ,  -26.09683834,  -26.09683834,
         -21.65339354,  -21.65339354,    9.18113812,    9.18113812,
         -12.4691749 ,  -12.4691749 ]]), array([[  798.75716001, -1412.51388623,   610
.77566718,   578.76130758,
        -1182.86462285,    55.21281718,  -105.79192609,  -281.52962062,
          -90.47775833,   -28.15921336,   -33.92815358,   -45.34747203,
          -45.34747203,    12.4568171 ,    12.4568171 ,    77.52378142,
           77.52378142,    -1.56846456,    -1.56846456,    -5.61804295,
          -18.99325624,   -18.99325624,    25.66238177,    25.66238177,
           58.73964939,    58.73964939,    36.84463211,    36.84463211,
           37.51281242,    37.51281242,     5.50985845,     5.50985845,
```

```
          18.3107492 ,   -31.23114162,   -31.23114162,    62.27484438,
          62.27484438,   -12.77530046,    14.44069482,    14.44069482,
          55.8102783 ,    55.8102783 ,   -26.09683834,   -26.09683834,
         -21.65339354,   -21.65339354,     9.18113812,     9.18113812,
         -12.4691749 ,   -12.4691749 ]]), array([[1047.23894729,  786.52224899, -394.
22281474, -531.40009245,
        -517.85598769,  320.64639218, -127.70042618, -185.95461291,
          89.63414441,  -60.59784921,  -33.92815358,  -45.34747203,
         -45.34747203,   12.4568171 ,   12.4568171 ,   77.52378142,
          77.52378142,   -1.56846456,   -1.56846456,   -5.61804295,
         -18.99325624,  -18.99325624,   25.66238177,   25.66238177,
          58.73964939,   58.73964939,   36.84463211,   36.84463211,
          37.51281242,   37.51281242,    5.50985845,    5.50985845,
          18.3107492 ,  -31.23114162,  -31.23114162,   62.27484438,
          62.27484438,  -12.77530046,   14.44069482,   14.44069482,
          55.8102783 ,   55.8102783 ,  -26.09683834,  -26.09683834,
         -21.65339354,  -21.65339354,    9.18113812,    9.18113812,
         -12.4691749 ,  -12.4691749 ]]), array([[ 1.14775546e+03, -1.38904023e+03,  8.
20113491e+02,
        -6.70613262e+01,  2.11035134e+02, -2.65111774e+02,
         1.17817508e+01,  8.51626639e+01,  1.00028403e+02,
        -9.70902881e-01, -3.39281536e+01, -4.53474720e+01,
        -4.53474720e+01,  1.24568171e+01,  1.24568171e+01,
         7.75237814e+01,  7.75237814e+01, -1.56846456e+00,
        -1.56846456e+00, -5.61804295e+00, -1.89932562e+01,
        -1.89932562e+01,  2.56623818e+01,  2.56623818e+01,
         5.87396494e+01,  5.87396494e+01,  3.68446321e+01,
         3.68446321e+01,  3.75128124e+01,  3.75128124e+01,
         5.50985845e+00,  5.50985845e+00,  1.83107492e+01,
        -3.12311416e+01, -3.12311416e+01,  6.22748444e+01,
         6.22748444e+01, -1.27753005e+01,  1.44406948e+01,
         1.44406948e+01,  5.58102783e+01,  5.58102783e+01,
        -2.60968383e+01, -2.60968383e+01, -2.16533935e+01,
        -2.16533935e+01,  9.18113812e+00,  9.18113812e+00,
        -1.24691749e+01, -1.24691749e+01]]), array([[1381.58690347,  856.31000407,  69
2.5805484 ,  751.61699992,
         -50.38952656,  703.25740084, -443.34230433,   25.00914128,
        -103.81716247,  -89.00064475,  -33.92815358,  -45.34747203,
         -45.34747203,   12.4568171 ,   12.4568171 ,   77.52378142,
          77.52378142,   -1.56846456,   -1.56846456,   -5.61804295,
         -18.99325624,  -18.99325624,   25.66238177,   25.66238177,
          58.73964939,   58.73964939,   36.84463211,   36.84463211,
          37.51281242,   37.51281242,    5.50985845,    5.50985845,
          18.3107492 ,  -31.23114162,  -31.23114162,   62.27484438,
          62.27484438,  -12.77530046,   14.44069482,   14.44069482,
          55.8102783 ,   55.8102783 ,  -26.09683834,  -26.09683834,
         -21.65339354,  -21.65339354,    9.18113812,    9.18113812,
         -12.4691749 ,  -12.4691749 ]]), array([[ 1474.52674103, -1375.52347467,  -478
.76604602,   651.00964469,
         -356.34627522,   156.9520568 ,  -369.15165887,   312.10150016,
           89.71002167,   397.66219462,   -33.92815358,   -45.34747203,
          -45.34747203,    12.4568171 ,    12.4568171 ,    77.52378142,
           77.52378142,    -1.56846456,    -1.56846456,    -5.61804295,
          -18.99325624,   -18.99325624,    25.66238177,    25.66238177,
           58.73964939,    58.73964939,    36.84463211,    36.84463211,
           37.51281242,    37.51281242,     5.50985845,     5.50985845,
           18.3107492 ,   -31.23114162,   -31.23114162,    62.27484438,
           62.27484438,   -12.77530046,    14.44069482,    14.44069482,
           55.8102783 ,    55.8102783 ,   -26.09683834,   -26.09683834,
          -21.65339354,   -21.65339354,     9.18113812,     9.18113812,
          -12.4691749 ,   -12.4691749 ]]), array([[1526.09747041, 1019.11340694,   78.
65747917,  415.00331233,
```

```
                -414.34936547, -605.3358783 , -700.9779866 ,   13.37006474,
               -250.87107874,  -72.78840541,  -33.92815358,  -45.34747203,
                -45.34747203,   12.4568171 ,   12.4568171 ,   77.52378142,
                 77.52378142,   -1.56846456,   -1.56846456,   -5.61804295,
                -18.99325624,  -18.99325624,   25.66238177,   25.66238177,
                 58.73964939,   58.73964939,   36.84463211,   36.84463211,
                 37.51281242,   37.51281242,    5.50985845,    5.50985845,
                 18.3107492 ,  -31.23114162,  -31.23114162,   62.27484438,
                 62.27484438,  -12.77530046,   14.44069482,   14.44069482,
                 55.8102783 ,   55.8102783 ,  -26.09683834,  -26.09683834,
                -21.65339354,  -21.65339354,    9.18113812,    9.18113812,
                -12.4691749 ,  -12.4691749 ]]), array([[ 1424.32600681, -1142.87151707,  -476
.19222957,   842.80514222,
                 106.26896794,    -9.19140884,  -159.92471772,  -545.49955871,
                 -13.4278299 ,  -242.89802552,   -33.92815358,   -45.34747203,
                 -45.34747203,    12.4568171 ,    12.4568171 ,    77.52378142,
                  77.52378142,    -1.56846456,    -1.56846456,    -5.61804295,
                 -18.99325624,   -18.99325624,    25.66238177,    25.66238177,
                  58.73964939,    58.73964939,    36.84463211,    36.84463211,
                  37.51281242,    37.51281242,     5.50985845,     5.50985845,
                  18.3107492 ,   -31.23114162,   -31.23114162,    62.27484438,
                  62.27484438,   -12.77530046,    14.44069482,    14.44069482,
                  55.8102783 ,    55.8102783 ,   -26.09683834,   -26.09683834,
                 -21.65339354,   -21.65339354,    9.18113812,    9.18113812,
                 -12.4691749 ,   -12.4691749 ]]), array([[1423.98009827,  869.37306212,  179.
61613325,  811.5022867 ,
                -500.43046875, -151.30842155, -128.29673897,   36.38703263,
                 676.39290273, -155.61843362,  -33.92815358,  -45.34747203,
                 -45.34747203,   12.4568171 ,   12.4568171 ,   77.52378142,
                  77.52378142,   -1.56846456,   -1.56846456,   -5.61804295,
                 -18.99325624,  -18.99325624,   25.66238177,   25.66238177,
                  58.73964939,   58.73964939,   36.84463211,   36.84463211,
                  37.51281242,   37.51281242,    5.50985845,    5.50985845,
                  18.3107492 ,  -31.23114162,  -31.23114162,   62.27484438,
                  62.27484438,  -12.77530046,   14.44069482,   14.44069482,
                  55.8102783 ,   55.8102783 ,  -26.09683834,  -26.09683834,
                 -21.65339354,  -21.65339354,    9.18113812,    9.18113812,
                 -12.4691749 ,  -12.4691749 ]]), array([[ 1442.46374563, -1359.6796919 ,  -156
.39252594,   370.84838097,
                -495.00703454,   128.64027941,  -279.89960026,   462.07705886,
                 -23.68660299,  -582.79645477,   -33.92815358,  -45.34747203,
                 -45.34747203,    12.4568171 ,    12.4568171 ,    77.52378142,
                  77.52378142,    -1.56846456,    -1.56846456,    -5.61804295,
-18.99325624,   -18.99325624,    25.66238177,    25.66238177,
 58.73964939,    58.73964939,    36.84463211,    36.84463211,
 37.51281242,    37.51281242,     5.50985845,     5.50985845,
 18.3107492 ,   -31.23114162,   -31.23114162,    62.27484438,
 62.27484438,   -12.77530046,    14.44069482,    14.44069482,
 55.8102783 ,    55.8102783 ,   -26.09683834,   -26.09683834,
-21.65339354,   -21.65339354,    9.18113812,    9.18113812,
-12.4691749 ,   -12.4691749 ]])]
```

- Compute the genuine scores between every pair of faces (there will be 45 genuine scores).

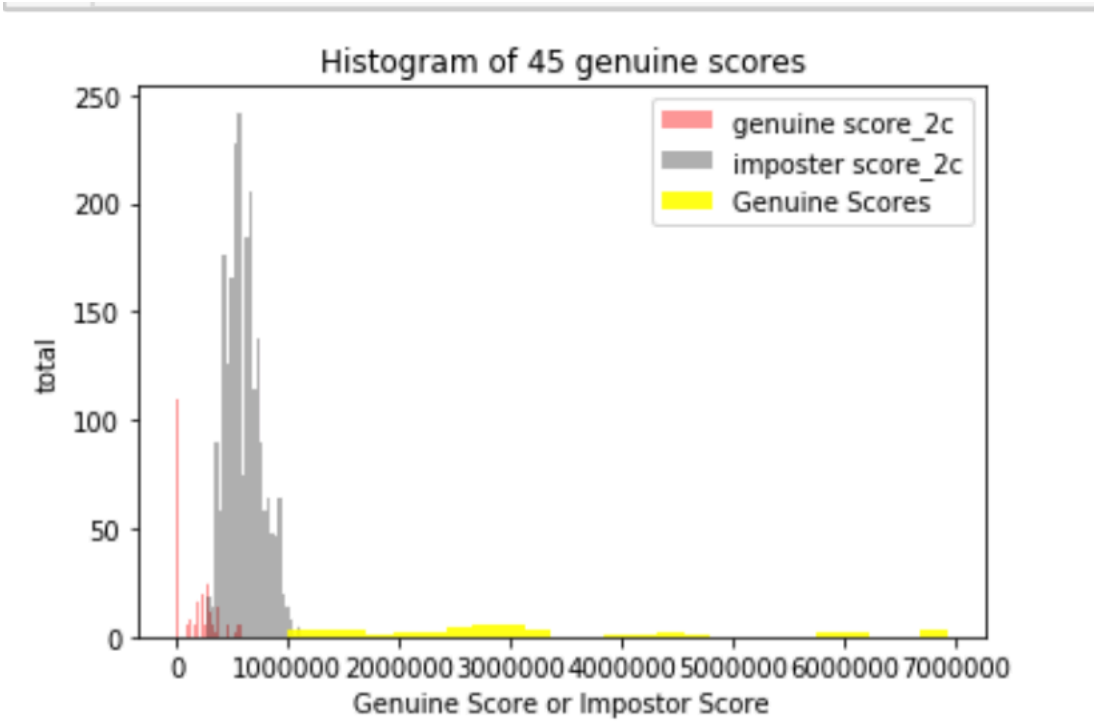
Ans)

```
genuine scores =  [7697233.999999996, 2671427.999999999, 7844609.0, 2367476.9999999986
, 6915396.999999999, 2085000.0000000007, 6026125.000000001, 1556644.0, 6760947.9999999
99, 7695782.000000002, 2814143.000000002, 7438070.999999999, 2979384.999999998, 824767
1.999999996, 3505934.999999998, 7052259.999999997, 2422812.0, 7404416.999999996, 34912
14.9999999977, 6831499.000000001, 2755326.000000003, 6432404.999999998, 2907271.999999
9995, 6399032.000000002, 7048913.999999996, 3165362.0000000014, 7872817.000000001, 322
0446.0000000014, 7256088.999999999, 2467549.0, 7126172.000000001, 2470881.0000000005,
6342684.0, 1914054.999999999, 6775922.999999996, 7224481.0, 1535266.0000000005, 640147
1.0, 1208338.999999999, 6492231.000000001, 1608978.0000000002, 6958441.999999997, 5691
537.000000001, 1898120.9999999977, 6231283.999999996]

Total number of genuine scores =  45
```

- Plot the histogram of 45 genuine scores on the same graph as2c. For this part, you can

re-plot the histograms from the previous question (2c) and use them here.

Ans)



- Comment on the accuracy of the face matcher.

Ans) The histogram demonstrates that the real values are excessively distributed and even exceed the fictitious scores from Q2, hence this face matcher will not work. One of the potential explanations is the model's training on frontal images while the data was gathered using non-frontal images. This is mostly the blame for this trainer's low rating. This is not a good match for the side-face photos.

Appendix

In [1]:
```
import numpy as np
import math
from matplotlib import pyplot as plt
from skimage import color
from skimage import io
from scipy import signal
from skimage.feature import local_binary_pattern
from PIL import Image, ImageEnhance, ImageFilter
import sys
import os
```

In [2]:
```
#1
img = Image.open('frontal_img.jpg')
img
```

Out[2]:

In [3]:
```
F_0 = img.convert('L')
F_0
```

Out[3]:

In [4]:
```
brightness = ImageEnhance.Brightness(F_0)
F_b = brightness.enhance(1.5)
F_b
```

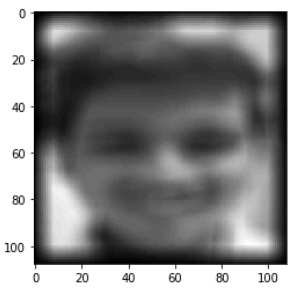
Out[4]:

In [5]:
```
contrast = ImageEnhance.Contrast(F_0)
F_c = contrast.enhance(3.0)
F_c
```

Out[5]:

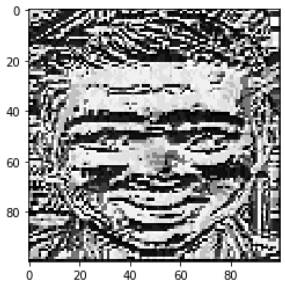
In [6]:
```
F_g = signal.convolve(F_0,np.ones((9,9))/(9*9))
plt.imshow(F_g, cmap = 'gray')
```

Out[6]: <matplotlib.image.AxesImage at 0x7f87181a4cd0>



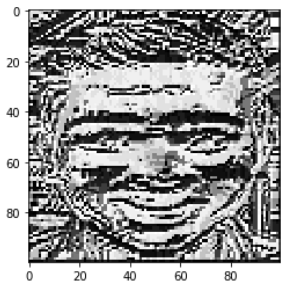
In [7]:
```
L_0 = local_binary_pattern(F_0, 8, 1)
plt.imshow(L_0, cmap = 'gray')
```

Out[7]: <matplotlib.image.AxesImage at 0x7f8719e7e290>



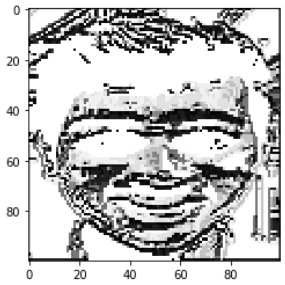
In [8]:
```
L_b = local_binary_pattern(F_b, 8, 1)
plt.imshow(L_0, cmap = 'gray')
```

Out[8]: <matplotlib.image.AxesImage at 0x7f8719f06910>



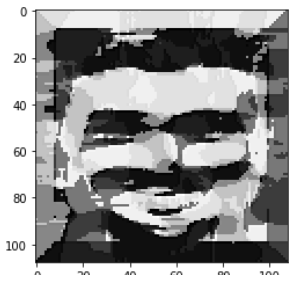
In [9]:
```
L_c = local_binary_pattern(F_c, 8, 1)
plt.imshow(L_c, cmap = 'gray')
```

Out[9]: <matplotlib.image.AxesImage at 0x7f871a875f10>



In [10]:
```
L_g = local_binary_pattern(F_g, 8, 1)
plt.imshow(L_g, cmap = 'gray')
```

Out[10]: <matplotlib.image.AxesImage at 0x7f871a99e710>



In [11]:
```
def pixel_comparator(image_1,image_2):
    image_1 = np.array(image_1)
    image_2 = np.array(image_2)
    sum_difference = 0
    i =0
    while i < 100:
        j=0
        while j<100:
            difference = abs(image_1[i][j]-image_2[i][j])
            sum_difference += difference
            j +=1
        i+=1

    result = sum_difference/(image_1.shape[0]*image_1.shape[1])
    return result
```

In [12]:
```
L_0_vs_L_b = pixel_comparator(L_0,L_b)
L_0_vs_L_b
```

Out[12]: 19.0068

In [13]:
```
L_0_vs_L_c = pixel_comparator(L_0,L_c)
L_0_vs_L_c
```

Out[13]: 53.7961

In [14]:
```
L_0_vs_L_g = pixel_comparator(L_0,L_g)
L_0_vs_L_g
```

Out[14]: 92.7782

In [15]:
```
F_0_vs_F_b = pixel_comparator(F_0,F_b)
F_0_vs_F_b
```

```
/opt/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:9: RuntimeWarning: ov
erflow encountered in ubyte_scalars
  if __name__ == '__main__':
```

Out[15]: 212.9101

In [16]:
```
F_0_vs_F_c = pixel_comparator(F_0,F_c)
F_0_vs_F_c
```

```
/opt/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:9: RuntimeWarning: ov
erflow encountered in ubyte_scalars
  if __name__ == '__main__':
```

Out[16]: 107.8967

In [17]:
```
F_0_vs_F_g = pixel_comparator(F_0,F_g)
F_0_vs_F_g
```

Out[17]: 34.11228765432111

In [18]:

```
#2
all_images = []
thirty_images = []
i = 1
while i < 11:
    j = 1
    while j < 6:

        if j<4:

            images_path = f'proj03_face_images/user{i:02}_{j:02}.bmp'
            image = np.asarray(Image.open(images_path))
            thirty_images.append(image.flatten())


        all_images_path = f'proj03_face_images/user{i:02}_{j:02}.bmp'
        all_image = np.asarray(Image.open(all_images_path))
        all_images.append(image.flatten())
        j += 1
    i+=1
```

In [19]:

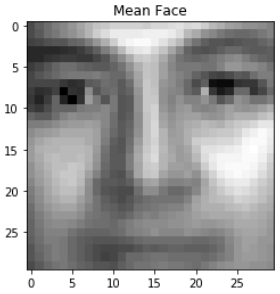
```
mean_list = []
i = 0
while i < len(thirty_images[0]):
    mean = []
    for j in thirty_images:
        mean.append(int(j[i]))
    m = (sum(mean)/30)
    mean_list.append(m)
    i+=1

mean_list
```

Out[19]:

```
[97.2,
 105.0,
 112.1,
 120.23333333333333,
 124.76666666666667,
 131.7,
 138.76666666666668,
 148.2,
 153.33333333333334,
 156.83333333333334,
 160.4,
 161.0,
 161.9,
 162.03333333333333,
 160.26666666666668,
 163.26666666666668,
 163.9,
 166.16666666666666,
 167.7,
 163.53333333333333,
 165.43333333333334,
```

In [20]:
```
mean_matrix = np.matrix(mean_list).T
mean_img = mean_matrix.reshape(30, 30)
plt.imshow(mean_img, cmap="gray")
plt.title("Mean Face")
```

Out[20]:
```
Text(0.5, 1.0, 'Mean Face')
```



In [180]:
```
pca_data = []
i = 0
while i < len(thirty_images):
    difference = np.array(thirty_images[i]) - np.array(mean_list)
    pca_data.append(difference)
    i+=1
pca_data_matrix = np.matrix(pca_data)
pca_data_matrix
```

Out[180]:
```
matrix([[ -7.2       ,  -4.        ,  -9.1       , ...,  -5.23333333,
          -4.26666667,  -5.16666667],
        [ -7.2       , -10.        , -16.1       , ...,  -5.23333333,
           0.73333333,  -1.16666667],
        [ -8.2       ,  -6.        , -10.1       , ...,  17.76666667,
          19.73333333,  15.83333333],
        ...,
        [ -6.2       ,  -9.        ,  -6.1       , ...,  -0.23333333,
          -7.26666667,  -8.16666667],
        [-13.2       , -13.        ,  -3.1       , ...,  -4.23333333,
          -4.26666667,  -5.16666667],
        [-19.2       , -23.        , -13.1       , ..., -10.23333333,
          -3.26666667, -12.16666667]])
```

In [22]:
```
covariance = np.dot(pca_data_matrix.T, pca_data_matrix)
covariance.shape
```

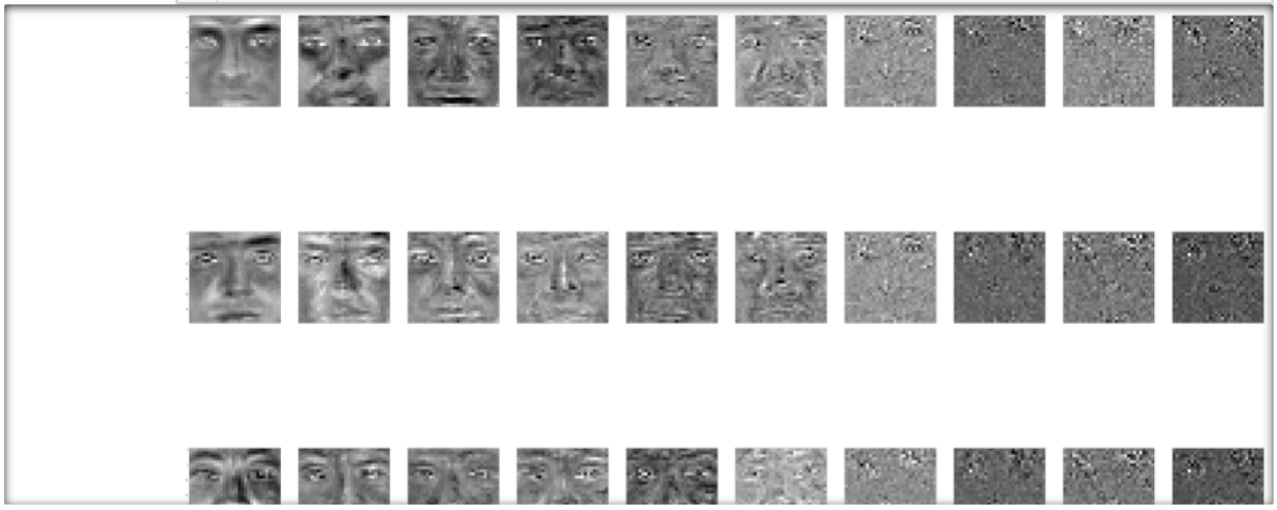
Out[22]:
```
(900, 900)
```

In [ ]:
```

```

In [191]:
```
def eigen(covariance, vec):
    eigval, eigvec = np.linalg.eig(covariance)
    eigval = eigval.real
    eigval_sort = np.argsort(eigval)[-50:][::-1]
    eigval = eigval[eigval_sort]
    eigvec = eigvec[:,eigval_sort].real

    return eigval,eigvec


```

In [192]:

```
eigval, eigvec = eigen(covariance, 50)
fig, axes = plt.subplots(5,10,sharex=True,sharey=True,figsize=(100,100))
i = 0
while i < 50:
    axes[i%5][i//5].imshow((eigvec[:,i].reshape(30,30)), cmap="gray")
    i+=1

```



In [30]:

```
def eigencoefficients(vector):
    e_c = []
    i = 0
    while i<len(all_images):
        eigen_coefficients = np.dot(vector.T, all_images[i] - (mean_list))
        e_c.append(np.array(eigen_coefficients))
        e_c_array = np.array(e_c)
        i+=1
    return e_c_array
```

In [198]:

```
value, vector = eigen(covariance,50)
print(eigencoefficients(vector))
```

```
[[[-1.65062019e+02  5.59122139e+01  3.87395719e+02 ... -1.77635684e-14
   -2.13162821e-14 -2.13162821e-14]]

 [[-2.38963202e+02  3.24997001e+01  3.39177800e+02 ... -3.01980663e-14
   -1.77635684e-14 -1.77635684e-14]]

 [[-2.33919502e+02  6.66753226e+01  2.12357545e+02 ... -2.30926389e-14
   -3.19744231e-14 -3.19744231e-14]]

 ...

 [[-2.73088075e+01 -1.27073964e+02 -2.17346337e+02 ... -5.32907052e-15
    2.48689958e-14  2.48689958e-14]]

 [[-2.73088075e+01 -1.27073964e+02 -2.17346337e+02 ... -5.32907052e-15
    2.48689958e-14  2.48689958e-14]]

 [[-2.73088075e+01 -1.27073964e+02 -2.17346337e+02 ... -5.32907052e-15
    2.48689958e-14  2.48689958e-14]]]
```

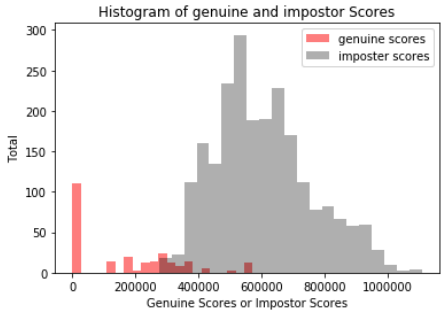
In [223]:
```
def score_calculator(x):

    eigen_coeff = eigencoefficients(vector)

    genuine_scores = []
    imposter_scores = []

    i=0
    while i<(len(eigen_coeff)):
        j = 0
        while j<(len(eigen_coeff)):
            difference = eigen_coeff[i] - eigen_coeff[j]
            absolute = np.abs(difference)
            comparison = pow(absolute, 2)
            comparison = np.sum(comparison)
            if (i//5) == (j//5):
                genuine_scores.append(comparison)
            else:
                imposter_scores.append(comparison)
            j+=1
        i+=1

    return genuine_scores, imposter_scores


```

In [224]:
```
score_calculator_temp = score_calculator(50)
print('\n',len(score_calculator_temp[0]))

```

```
 250
```

In [219]:
```
x, y = score_calculator(50)
plt.hist(x,bins = 21, label = 'genuine scores', alpha = 0.5, color = 'red')
plt.hist(y,bins = 21, label = 'imposter scores', alpha = 0.3, color = 'black')
plt.xlabel('Genuine Scores or Impostor Scores')
plt.ylabel('Total')
plt.legend()
plt.title("Histogram of genuine and impostor Scores")
```

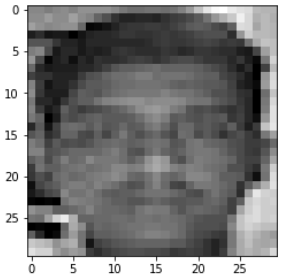
Out[219]:
```
Text(0.5, 1.0, 'Histogram of genuine and impostor Scores')
```



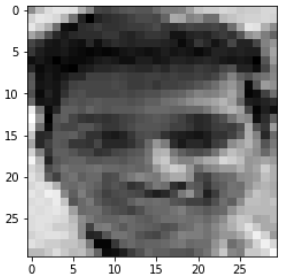
In [220]:
```
np.savetxt("Genuine_scores.mat", x)
np.savetxt("Imposter_scores.mat", y)
```

In [141]:
```
#3
img_1 = Image.open('IMG_1.jpg')
img_2 = Image.open('IMG_2.jpg')
img_3 = Image.open('IMG_3.jpg')
img_4 = Image.open('IMG_4.jpg')
img_5 = Image.open('IMG_5.jpg')
img_6 = Image.open('IMG_6.jpg')
img_7 = Image.open('IMG_7.jpg')
img_8 = Image.open('IMG_8.jpg')
img_9 = Image.open('IMG_9.jpg')
img_10 = Image.open('IMG_10.jpg')

img_gray = []
```

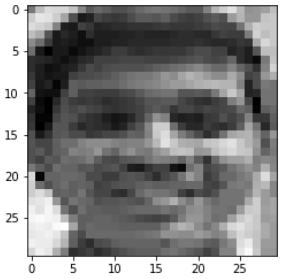
In [142]:
```
img_1_gray = img_1.convert('L')
plt.imshow(img_1_gray, cmap = 'gray')
img_gray.append(np.asarray(img_1_gray).flatten())
```



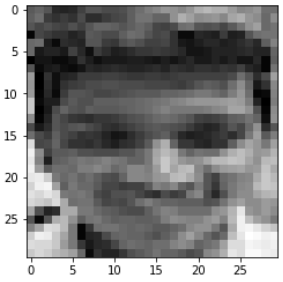
In [143]:
```
img_2_gray = img_2.convert('L')
plt.imshow(img_2_gray, cmap = 'gray')
img_gray.append(np.asarray(img_2_gray).flatten())
```



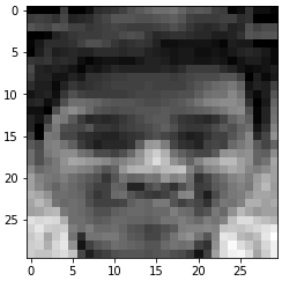
In [144]:
```
img_3_gray = img_3.convert('L')
plt.imshow(img_3_gray, cmap = 'gray')
img_gray.append(np.asarray(img_3_gray).flatten())
```



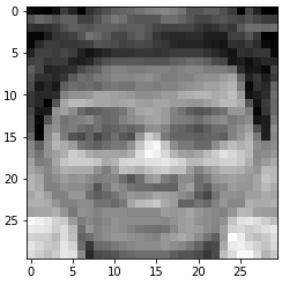
In [145]:
```
1 img_4_gray = img_4.convert('L')
2 plt.imshow(img_4_gray, cmap = 'gray')
3 img_gray.append(np.asarray(img_4_gray).flatten())
```



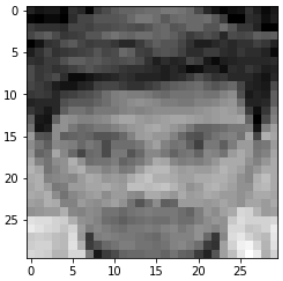
In [146]:
```
1 img_5_gray = img_5.convert('L')
2 plt.imshow(img_5_gray, cmap = 'gray')
3 img_gray.append(np.asarray(img_5_gray).flatten())
```



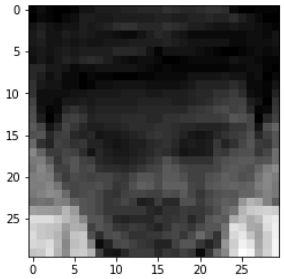
In [147]:
```
1 img_6_gray = img_6.convert('L')
2 plt.imshow(img_6_gray, cmap = 'gray')
3 img_gray.append(np.asarray(img_6_gray).flatten())
```



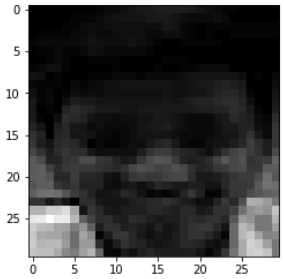
In [148]:
```
1 img_7_gray = img_7.convert('L')
2 plt.imshow(img_7_gray, cmap = 'gray')
3 img_gray.append(np.asarray(img_7_gray).flatten())
```



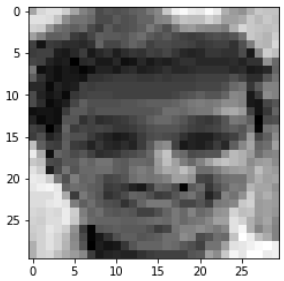
In [149]:
```
1 img_8_gray = img_8.convert('L')
2 plt.imshow(img_8_gray, cmap = 'gray')
3 img_gray.append(np.asarray(img_8_gray).flatten())
```



In [150]:
```
1 img_9_gray = img_9.convert('L')
2 plt.imshow(img_9_gray, cmap = 'gray')
3 img_gray.append(np.asarray(img_9_gray).flatten())
```



In [151]:
```
1 img_10_gray = img_10.convert('L')
2 plt.imshow(img_10_gray, cmap = 'gray')
3 img_gray.append(np.asarray(img_10_gray).flatten())
```



In [153]:
```
mean_list_q3 = []
i = 0
while i < len(img_gray[0]):
    mean = []
    for j in img_gray:
        mean.append(int(j[i]))
    m = (sum(mean)`/30)
    mean_list_q3.append(m)
    i+=1

mean_list_q3
```

Out[153]:
```
[23.566666666666666,
 29.366666666666667,
 30.066666666666666,
 29.533333333333335,
 31.866666666666667,
 28.133333333333333,
 23.866666666666667,
 22.0,
 22.9,
 23.6,
 24.133333333333333,
 25.433333333333334,
 27.066666666666666,
 28.833333333333332,
 31.5,
 33.666666666666664,
 37.2,
 38.1,
 37.43333333333333,
 37.2,
 39.7,
```

In [154]:
```
pca_data_q3 = []
i = 0
while i < len(img_gray):
    difference = np.array(img_gray[i]) - np.array(mean_list_q3)
    pca_data_q3.append(difference)
    i+=1
pca_data_matrix_q3 = np.matrix(pca_data_q3)
pca_data_matrix_q3.shape
```

Out[154]:
```
(10, 900)
```

In [155]:
```
covariance_q3 = np.dot(pca_data_matrix_q3.T, pca_data_matrix_q3)
covariance_q3.shape
```

Out[155]:
```
(900, 900)
```

In [156]:

```
eig_val, eig_vec = eigen(covariance_q3,25)

eigen_coefficients = []
i = 0
while i < len(img_gray):
    e_c = np.dot(eig_vec.T, img_gray[i]-mean_list)
    e_c = np.array(e_c)
    eigen_coefficients.append(e_c)
    i+=1
print(eigen_coefficients)
```

```
[array([[ 561.34436746,  534.76905278,  -25.72407189, -427.66847029,
         794.01096208,  667.3253231 ,  216.89865483, -128.31072475,
          78.46172945,   56.48423781,  -15.88645646,  -15.88645646,
          21.03180401,    1.11464486,    1.11464486,    2.30889196,
           2.30889196,  -14.64173439,  -14.64173439,   56.63606648,
          59.79960625,   72.83849433,   72.83849433,   49.0285998 ,
          49.0285998 ,  -88.47647763,    4.15950647,    4.15950647,
         -23.18613829,  -23.18613829,   26.32133481,   26.32133481,
          -1.57998777,   -1.57998777,  -31.90273494,  -31.90273494,
         -37.71291452,    3.75672367,    3.75672367,  -18.3830851 ,
         -18.3830851 ,   -1.50616744,   -1.50616744,  -55.10781024,
         -34.18236618,  -34.18236618,  -51.90186135,  -51.90186135,
          21.7802648 ,   21.7802648 ]]), array([[ 2.05090510e+02,  4.02070533e+02, -1.
33452199e+03,
         1.73727249e+02,  3.91200767e+02,  7.33447790e+02,
        -1.68541169e+02,  3.19822942e+02,  1.02429905e+02,
         5.19956372e+01, -1.58864565e+01, -1.58864565e+01,
         2.10318040e+01,  1.11464486e+00,  1.11464486e+00,
         2.30889196e+00,  2.30889196e+00, -1.46417344e+01,
        -1.46417344e+01,  5.66360665e+01,  5.97996063e+01,
         7.28384943e+01,  7.28384943e+01,  4.90285998e+01,
```

In [157]:

```
def genuine_score_calculator(x):

    genuine_scores = []

    i = 0
    while i < 10:
        j = i+1
        while j<10:
            difference = eigen_coefficients[i] - eigen_coefficients[j]
            result = difference**2
            genuine_scores.append(np.sum(result))
            j+=1
        i+=1
    return genuine_scores
```

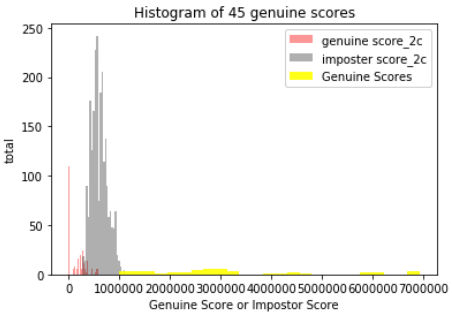
In [158]:

```
scores_length = genuine_score_calculator(25)
print("genuine scores = ",scores_length)
print('\n'"Total number of genuine scores = ",len(scores_length))
```

```
genuine scores =  [2735764.999999998, 2602529.9999999986, 2780819.999999999, 3108211.9
99999998, 2990206.9999999986, 3037154.9999999986, 4371942.999999995, 6170207.999999997
, 2745044.0, 2040150.9999999984, 1397486.9999999986, 2351237.000000001, 2842000.000000
0037, 2913822.0000000023, 3847058.0, 6190051.0, 1017286.999999999, 2574972.000000001,
2815439.999999999, 2505320.999999999, 3177007.0000000005, 4605248.999999999, 6726679.9
99999999, 2215354.0000000037, 1566284.0000000023, 1973791.0000000035, 1920451.00000000
12, 3360121.000000001, 5819428.000000001, 1357846.0000000019, 1322605.0, 1483739.00000
00014, 1635663.0000000012, 3278884.0000000014, 2564856.0000000023, 1211690.0000000028,
4237980.000000001, 6822181.0, 3075181.0000000037, 2973457.999999998, 5914446.999999999
, 2816503.0000000023, 995451.0000000002, 4355179.000000002, 6934126.000000004]

Total number of genuine scores =  45
```

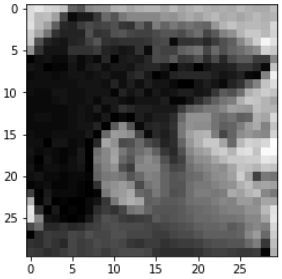
In [159]:

```
plt.hist(x, bins=25, label='genuine score_2c', alpha=0.4, color = 'red')
plt.hist(y, bins=25, label='imposter score_2c', alpha=0.3, color = 'black')
plt.hist(scores_length,bins=25, label="Genuine Scores", alpha=0.9, color = 'yellow'
plt.xlabel('Genuine Score or Impostor Score')
plt.ylabel('total')
plt.legend()
plt.title("Histogram of 45 genuine scores")
plt.show()
```

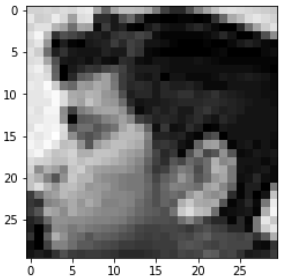


In [160]:
```
#4
side_img_1 = Image.open('side_image_1.jpg')
side_img_2 = Image.open('side_image_2.jpg')
side_img_3 = Image.open('side_image_3.jpg')
side_img_4 = Image.open('side_image_4.jpg')
side_img_5 = Image.open('side_image_5.jpg')
side_img_6 = Image.open('side_image_6.jpg')
side_img_7 = Image.open('side_image_7.jpg')
side_img_8 = Image.open('side_image_8.jpg')
side_img_9 = Image.open('side_image_9.jpg')
side_img_10 = Image.open('side_image_10.jpg')

side_img_gray = []
```

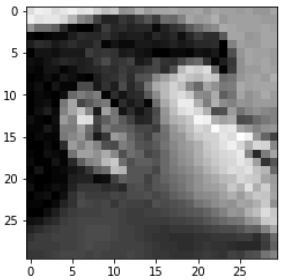
In [161]:
```
side_img_1_gray = side_img_1.convert('L')
plt.imshow(side_img_1_gray, cmap = 'gray')
side_img_gray.append(np.asarray(side_img_1_gray).flatten())
```



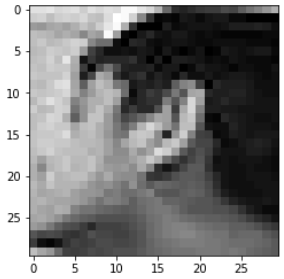
In [162]:
```
side_img_2_gray = side_img_2.convert('L')
plt.imshow(side_img_2_gray, cmap = 'gray')
side_img_gray.append(np.asarray(side_img_2_gray).flatten())
```



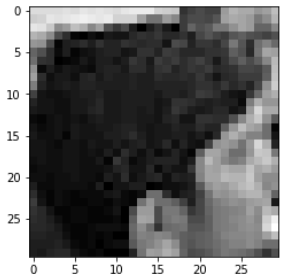
In [163]:
```
side_img_3_gray = side_img_3.convert('L')
plt.imshow(side_img_3_gray, cmap = 'gray')
side_img_gray.append(np.asarray(side_img_3_gray).flatten())
```



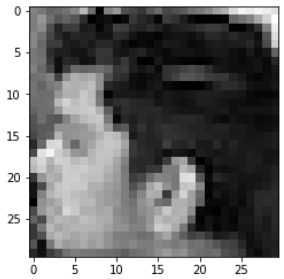
In [164]:
```
side_img_4_gray = side_img_4.convert('L')
plt.imshow(side_img_4_gray, cmap = 'gray')
side_img_gray.append(np.asarray(side_img_4_gray).flatten())
```



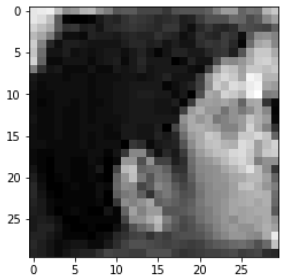
In [165]:
```
side_img_5_gray = side_img_5.convert('L')
plt.imshow(side_img_5_gray, cmap = 'gray')
side_img_gray.append(np.asarray(side_img_5_gray).flatten())
```



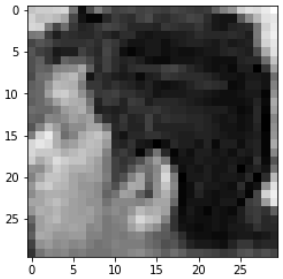
In [166]:
```
side_img_6_gray = side_img_6.convert('L')
plt.imshow(side_img_6_gray, cmap = 'gray')
side_img_gray.append(np.asarray(side_img_6_gray).flatten())
```



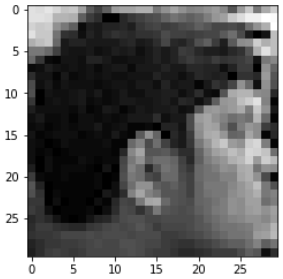
In [167]:
```
side_img_7_gray = side_img_7.convert('L')
plt.imshow(side_img_7_gray, cmap = 'gray')
side_img_gray.append(np.asarray(side_img_7_gray).flatten())
```



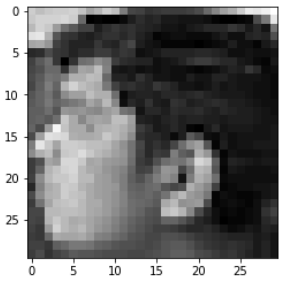
In [168]:
```
side_img_8_gray = side_img_8.convert('L')
plt.imshow(side_img_8_gray, cmap = 'gray')
side_img_gray.append(np.asarray(side_img_8_gray).flatten())
```



In [169]:
```
side_img_9_gray = side_img_9.convert('L')
plt.imshow(side_img_9_gray, cmap = 'gray')
side_img_gray.append(np.asarray(side_img_9_gray).flatten())
```



In [170]:
```
side_img_10_gray = side_img_10.convert('L')
plt.imshow(side_img_10_gray, cmap = 'gray')
side_img_gray.append(np.asarray(side_img_10_gray).flatten())
```



In [172]:
```
mean_list_q4 = []
i = 0
while i < len(side_img_gray[0]):
    mean = []
    for j in side_img_gray:
        mean.append(int(j[i]))
    m = (sum(mean)/30)
    mean_list_q4.append(m)
    i+=1

mean_list_q4
```

Out[172]:
```
[60.0,
 60.5,
 60.56666666666667,
 58.1,
 56.9,
 52.5,
 49.7,
 48.53333333333333,
 43.0,
 44.166666666666664,
 47.2,
 44.766666666666666,
 42.93333333333333,
 42.1,
 41.4,
 37.7,
 32.53333333333333,
 32.56666666666667,
 26.666666666666668,
 28.733333333333334,
 31.966666666666665,
```

In [173]:
```
pca_data_q4 = []
i = 0
while i < len(side_img_gray):
    difference = np.array(side_img_gray[i]) - np.array(mean_list_q4)
    pca_data_q4.append(difference)
    i+=1
pca_data_matrix_q4 = np.matrix(pca_data_q4)
pca_data_matrix_q4.shape
```

Out[173]: (10, 900)

In [174]:
```
covariance_q4 = np.dot(pca_data_matrix_q4.T, pca_data_matrix_q4)
covariance_q4.shape
```

Out[174]: (900, 900)

In [175]:

```
eig_val_q4, eig_vec_q4 = eigen(covariance_q4,25)

eigen_coefficients_q4 = []
i = 0
while i < len(side_img_gray):
    e_c = np.dot(eig_vec_q4.T, side_img_gray[i]-mean_list)
    e_c = np.array(e_c)
    eigen_coefficients_q4.append(e_c)
    i+=1
print(eigen_coefficients_q4)
```

```
[array([[1161.64669284,  967.89280501,  -99.55454012,  753.88389404,
        -354.30152531,  -73.37412875,  513.96036741,  209.34663384,
        -279.48492395,  -40.18947441,  -33.92815358,  -45.34747203,
         -45.34747203,   12.4568171 ,   12.4568171 ,   77.52378142,
          77.52378142,   -1.56846456,   -1.56846456,   -5.61804295,
         -18.99325624,  -18.99325624,   25.66238177,   25.66238177,
          58.73964939,   58.73964939,   36.84463211,   36.84463211,
          37.51281242,   37.51281242,    5.50985845,    5.50985845,
          18.3107492 ,  -31.23114162,  -31.23114162,   62.27484438,
          62.27484438,  -12.77530046,   14.44069482,   14.44069482,
          55.8102783 ,   55.8102783 ,  -26.09683834,  -26.09683834,
         -21.65339354,  -21.65339354,    9.18113812,    9.18113812,
         -12.4691749 ,  -12.4691749 ]]), array([[  798.75716001, -1412.51388623,   610
.77566718,   578.76130758,
        -1182.86462285,    55.21281718,  -105.79192609,  -281.52962062,
          -90.47775833,   -28.15921336,   -33.92815358,   -45.34747203,
          -45.34747203,    12.4568171 ,    12.4568171 ,    77.52378142,
           77.52378142,    -1.56846456,    -1.56846456,    -5.61804295,
          -18.99325624,   -18.99325624,    25.66238177,    25.66238177,
           58.73964939,    58.73964939,    36.84463211,    36.84463211,
           37.51281242,    37.51281242,     5.50985845,     5.50985845,
```

In [176]:

```
def genuine_score_calculator_q4(x):

    genuine_scores_q4 = []

    i = 0
    while i < 10:
        j = i+1
        while j<10:
            difference = eigen_coefficients_q4[i] - eigen_coefficients_q4[j]
            result = difference**2
            genuine_scores_q4.append(np.sum(result))
            j+=1
        i+=1
    return genuine_scores_q4
```

In [177]:

```
scores_length_q4 = genuine_score_calculator_q4(25)
print("genuine scores = ",scores_length_q4)
print('\n'"Total number of genuine scores = ",len(scores_length_q4))
```

```
genuine scores =  [7697233.999999996, 2671427.999999999, 7844609.0, 2367476.9999999986
, 6915396.999999999, 2085000.0000000007, 6026125.000000001, 1556644.0, 6760947.9999999
99, 7695782.000000002, 2814143.000000002, 7438070.999999999, 2979384.999999998, 824767
1.999999996, 3505934.999999998, 7052259.999999997, 2422812.0, 7404416.999999996, 34912
14.9999999977, 6831499.000000001, 2755326.000000003, 6432404.999999998, 2907271.999999
9995, 6399032.000000002, 7048913.999999996, 3165362.0000000014, 7872817.000000001, 322
0446.0000000014, 7256088.999999999, 2467549.0, 7126172.000000001, 2470881.0000000005,
6342684.0, 1914054.999999999, 6775922.999999996, 7224481.0, 1535266.0000000005, 640147
1.0, 1208338.999999999, 6492231.000000001, 1608978.0000000002, 6958441.999999997, 5691
537.000000001, 1898120.9999999977, 6231283.999999996]

Total number of genuine scores =  45
```

In [179]:

```
plt.hist(x, bins=25, label='genuine score_2c', alpha=0.4, color = 'red')
plt.hist(y, bins=25, label='imposter score_2c', alpha=0.3, color = 'black')
plt.hist(scores_length_q4,bins=25, label="Genuine Scores", alpha=0.9, color = 'yell
plt.xlabel('Genuine Score or Impostor Score')
plt.ylabel('total')
plt.legend()
plt.title("Histogram of 45 genuine scores")
plt.show()
```