**Assignment #2 Report – EigenFaces, Adaboost, Detection**

**Written By: mhtay**

**P1)**

Eigen-Face was computed from the corpus of images, and the mean and divided by subsequent zero-mean variance to obtain the normalized Eigen-face:



Figure : Eigen-Face

The dot product of the normalized eigen-face was taken against the image, cornered at every pixel of the image except the corners using a sliding window.

The maximum values of the dot product can be seen in an image of the scores obtained.

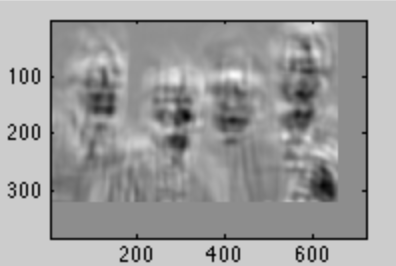


Figure :DotProductImage

However, due to the high intensity of features like the hand, false positives is possible. Thus, a normalized dot product of the faces with the eigenface might produce better results, by not only subtracting the mean of the patches, but also dividing by the sum of each zero mean patch. The resulting score is more informative as the brighter regions correspond to patches cornered at that area.

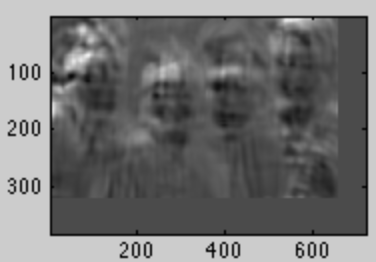


Figure :Zero-Mean, 1 variance patches dot product

Dividing each zero mean patch with the sum of the patch before the mean normalization, however, gives better distinction of the faces:

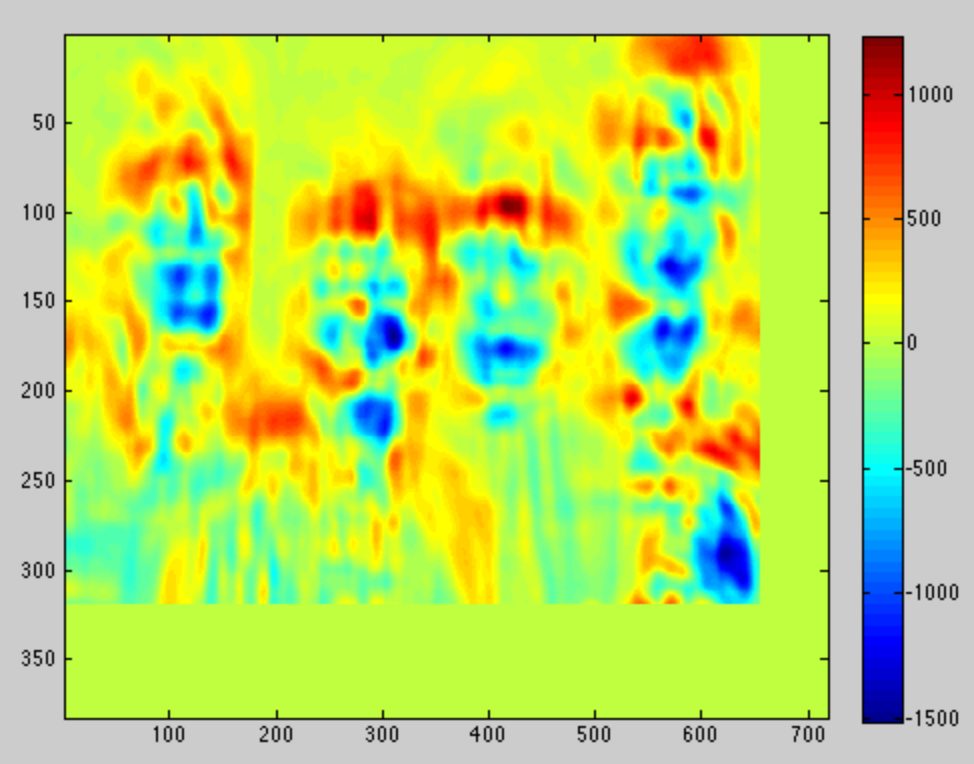
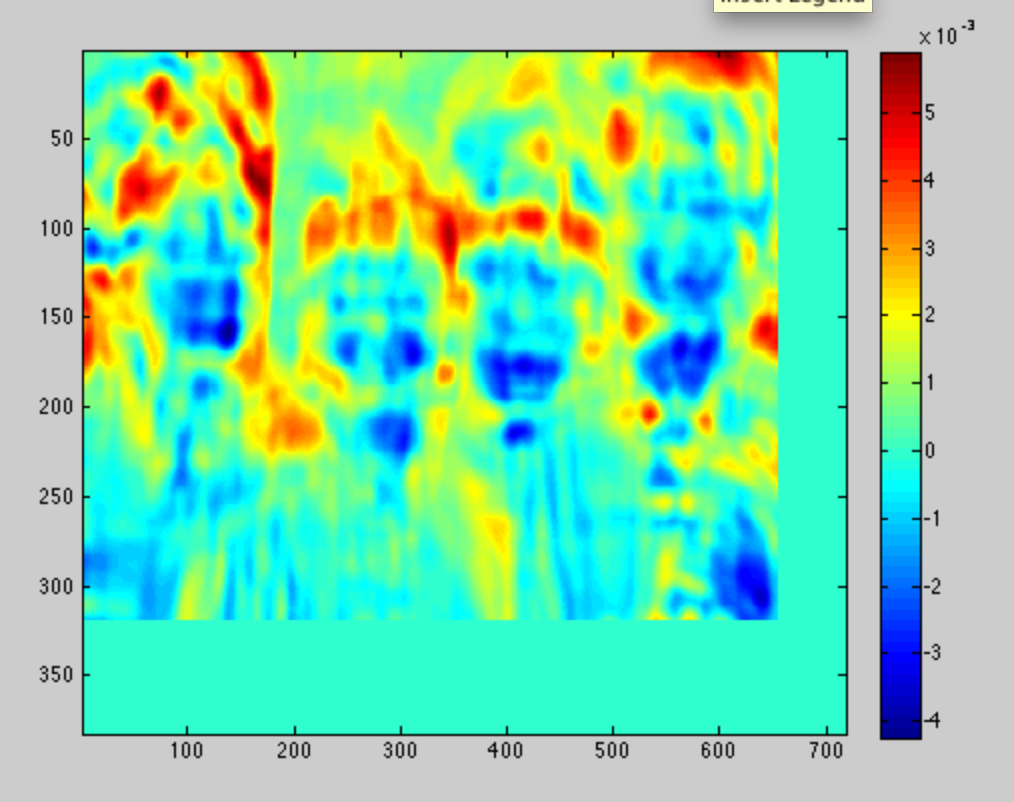


Figure 5: After dividing by sum of patch (before mean norm)

Figure :Without dividing by sum

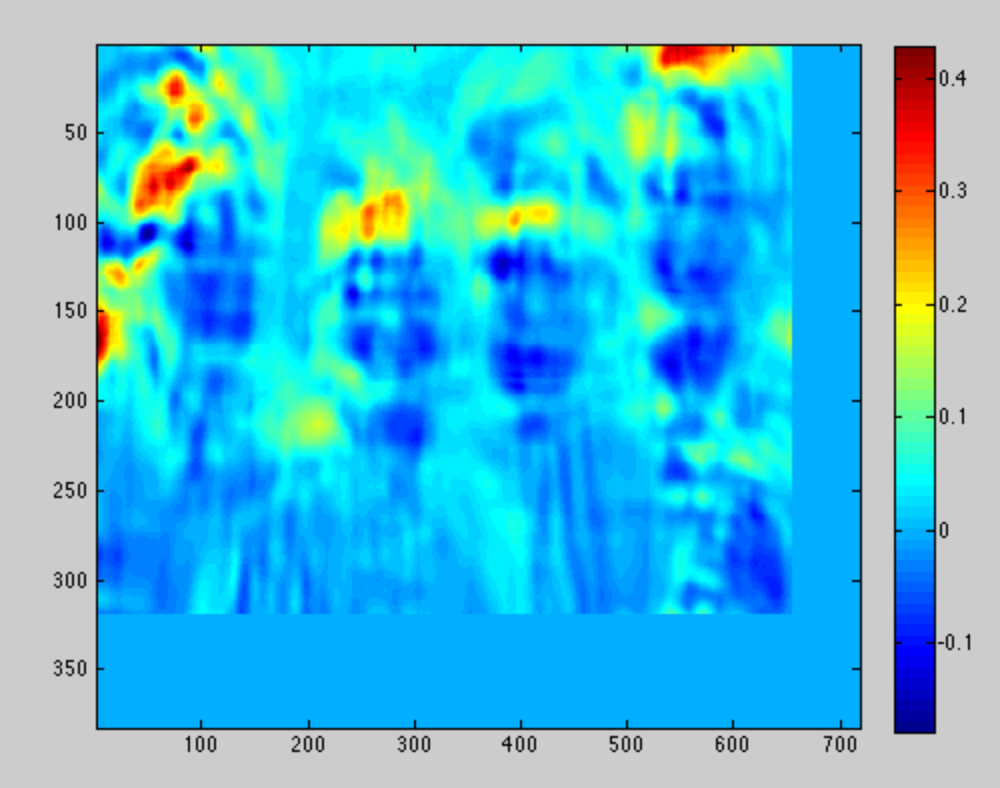


Figure 6: after dividing by sum of mean norm patch

However, the approach taken in figure 6 does not work well on the other image, so the approach was taken to carry out the normalized dot product on each patch after subtracting the patch mean and without dividing by norm, using the eigenface learned after mean and standard deviation normalization.

A query sliding face window, which indicated a 1 in the presence of a face, and 0 otherwise was carried out. The decision on whether a face was present was based on a threshold that was empirically found to be 70% of the maximum patch score.

This was repeated at each of the 5 scales, from a value of 0.5, 0.75, 1.0, 1.5 and 2.0. The resulting faces were detected at:

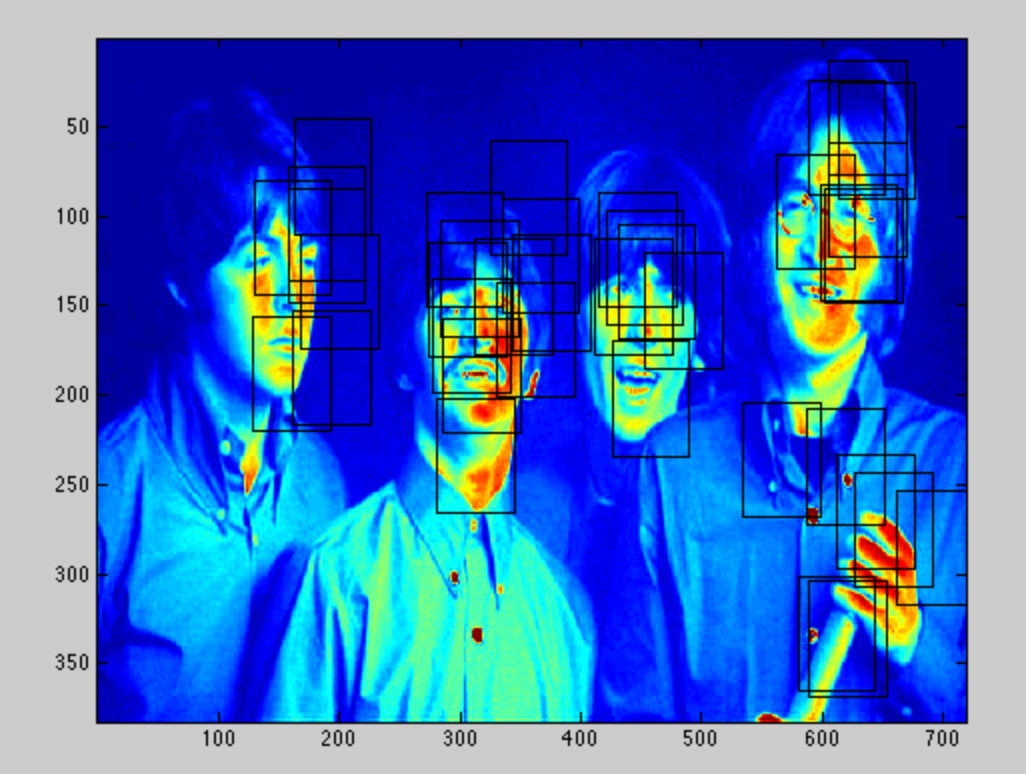


Figure 7: Multi-scale sliding window face query

Thereafter, bounding boxes in proximity were coalesced into a single bounding box. This was done by searching for the nearest bounding box to the existing box, then taking the average of the 2 boxes as the new box. However, the new box produced might be close to an existing box in the shortlisted list, thus the process of searching for nearest boxes needs to be repeated until no closest boxes are found. The limit for the closest box is set to 64, the approximate size of the Eigen-face rectangle.

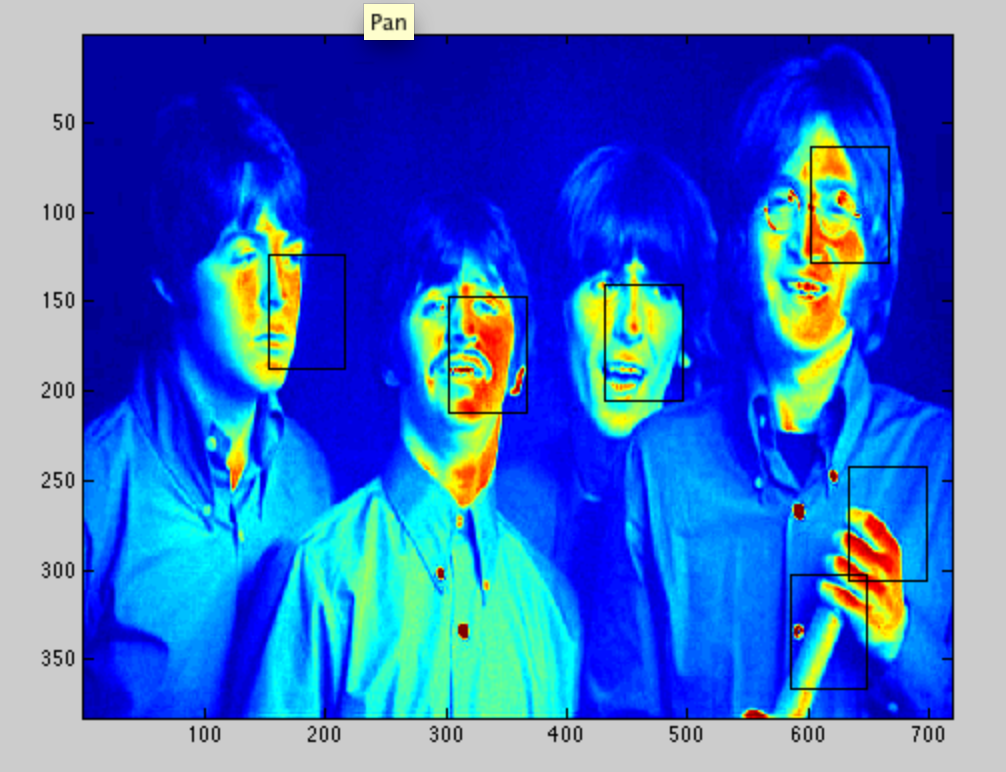


Figure 8: Short-listed faces

Repeating the Process on the other images:

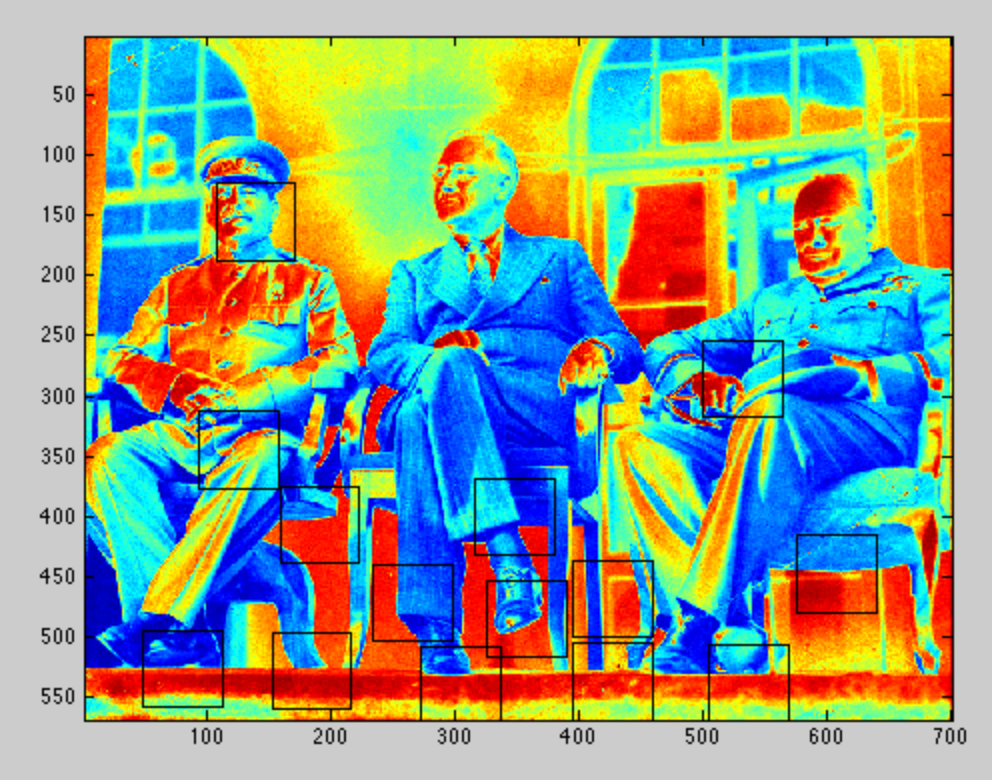


Figure 9:Russian gathering photo

The faces that are turned away from the camera are not well detected, as the training data are all faces that are looking directly at the camera. (front-profile)

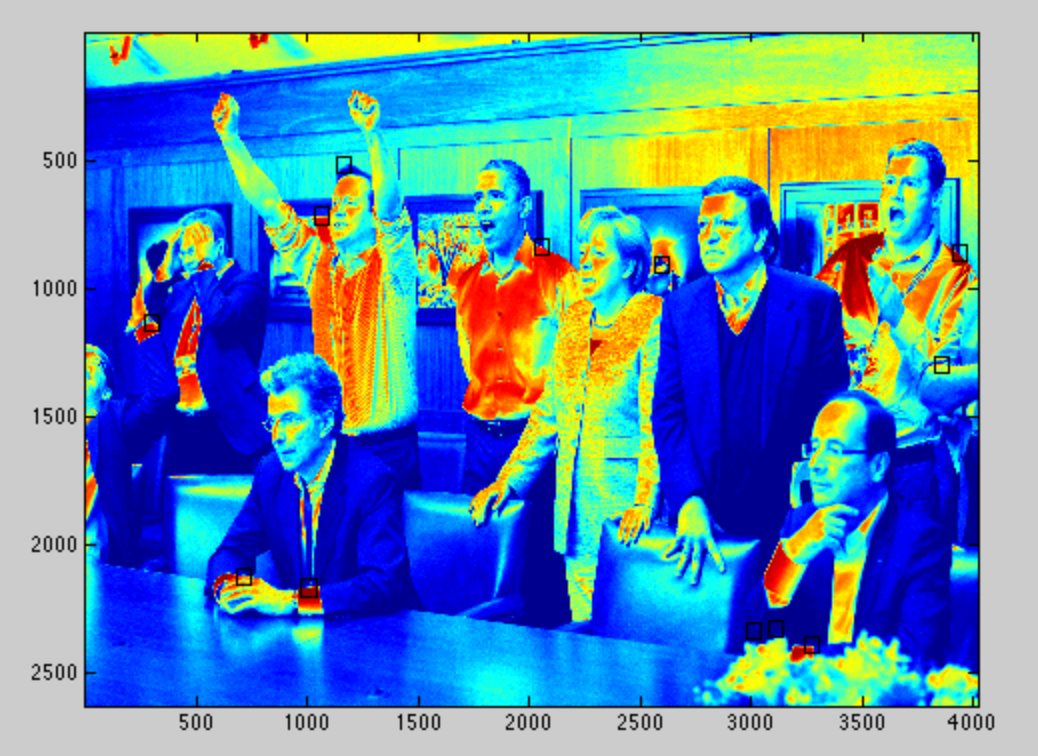


Figure : Group meeting after operation mission success

**P2)**

Initially, we created a naïve bayes classifier to be used in conjunction with adaboost for classification. The weighted values were incorporated into the classifier such that more highly weighted values were given higher counts and hence higher probability. However, the probability of observing a feature was the sum of all the observed features, and it was a problem since the feature values could be positive and negative and thus sum to 0, thus multiple different feature sets could be read as the same value in this version of the naïve bayes classifier. A train accuracy of 65% was obtained.

Thereafter, a switch was made to the stump classifier to be used as the weak classifier. A stump is a vector of values for a certain feature. In this case, each feature is the weights the dot product of an image with an Eigen-face vector. There are K feature stumps, each corresponding to an Eigen-face. The stump classifier could accommodate both negative and positive values of the feature sets, and worked by making a prediction based on the best threshold and direction that could make a prediction that minimized the weighted error of the training data. The best threshold and corresponding error was recorded for all the stumps, and the best stump was chosen as the weak classifier for the t-th iteration of adaboost.

The final classifier was constructed by adaboost, by a weighted sum of the different weak classifiers learnt.

The number of feature vectors K, was varied from 2 to 16, while fixing the number of weak classifiers at 20. The train and test accuracy were then computed. Here the train data is made up from the corpus of labeled face and non-face images, while the test data is made up from a separate corpus of labeled face and non-face images.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | K=2 | K=4 | K=6 | K=8 | K=10 | K=12 | K=14 | K=16 |
| Train Accuracy |  |  |  |  |  |  |  |  |
| Test  Accuracy |  |  |  |  |  |  |  |  |

Using K=, the number of weak classifiers was varied from range of 50 to 400. In general, the larger the number of weak classifiers, the better the train and test accuracy, but the longer the time it takes to train the classifier.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | T=50 | T=100 | T=200 | T=300 | T=400 |
| Train  Accuracy |  |  |  |  |  |
| Test  Accuracy |  |  |  |  |  |

Using the best set of parameters learnt, the classifier was then used