1.1)

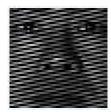
Test set	error rate
1	5.83%
2	48.33%
3	80.71%
4	84.74%

1.2) This performance makes sense since the accuracy lowers as the lighting conditions of the test images get worse. As the lighting conditions get worse, the faces become harder to distinguish since the unique features of each of the faces get overwritten by similar shadows. The error rate is very high for sets 3 and 4 because more than half of most faces are shown as dark/black pixels which are similar to the other images.

1.3)

Person 4_03
Img 24 (training set)

Person 2_16
img 21 (test set 1)



Since the image for person 2_16 has a shadow with the pattern of a mesh wire overlaying his face the closest match was person 4_03 from the training set because his picture was taken under a uniformly dar lighting condition. The lighting in the training image showed particular dark areas under his eyebrows, around the nose, and on the mustache. The mesh wire over Person 2_16 accented these features and may have been interpreted as a match to Person 4_03.

```
function Hw5
%load data
[trainset, trainlabels] = loadSubset(0);
[testset, testlabels] = loadSubset(1);
testNorms = calcNorms(testset,trainset);
predictions = KNN(testset,trainset,trainlabels,testNorms,1);
errorrate = errRate(predictions,testlabels)

%imshow(drawFaces(trainset,10));

%calculate the norm of each pair
function norms = calcNorms(tests,trains)
    [trainRow,trainCol] = size(trains);
    [testRow,testCol] = size(tests);
    norms = zeros(testRow,trainRow);

    for testR = 1:testRow
        for trainR = 1:trainRow
            dist = norm(tests(testR,:) - trains(trainR,:));
            norms(testR,trainR) = dist;
        end
    end
end

%calculate the label based on k
function knn = KNN(tests,trains,tLabels,norms,k)
    [trainRow,trainCol] = size(trains);
    [testRow,testCol] = size(tests);
    knn = zeros(testRow,1);
    for currRow = 1:testRow
        [normVals,normIndex] = sort(norms(currRow,:),'ascend');
        normInd = normIndex(1:k);
        kLabels = [];
        for label = normInd
            %[testRow,trainRow] = ind2sub(size(tLabels),normInd);
            kLabels = [kLabels;tLabels(normInd)];
        end
        knn(currRow) = mode(kLabels);
        if currRow == 21
            normInd
        end
    end

end

%calculate error rate
function err = errRate(predL,testL)
    s = size(predL);
    err = 0;
    for L = 1:s(1)
        if predL(L) ~= testL(L)
            err = err + 1;
        end
    end
    err = err/s(1);
end

end
```

2.1) l2 norm

K = 1

Test set	error rate
1	5.83%
2	48.33%
3	80.71%
4	84.74%

K = 3

Test set	error rate
1	5.83%
2	52.50%
3	82.14%
4	87.89%

K = 5

Test set	error rate
1	5.0%
2	53.33%
3	82.86%
4	88.95%

2.2) l1 norm

K = 1

Test set	error rate
1	5.83%
2	48.33%
3	78.57%
4	86.84%

K = 3

Test set	error rate
1	9.17%
2	51.67%
3	77.14%
4	86.32%

K = 5

Test set	error rate
1	6.67%
2	53.33%
3	82.14%
4	85.26%

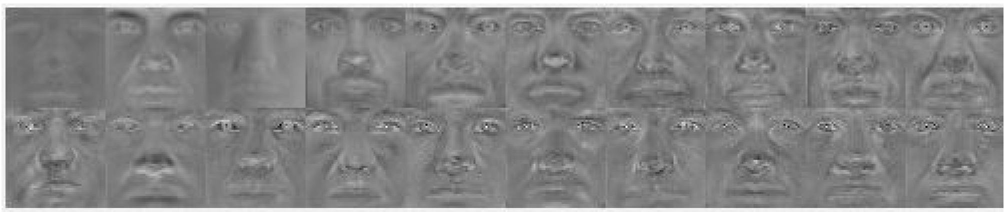
Same code as before except for this line

```
dist = norm(tests(testR,:) - trains(trainR,:),1);
```

2.3) As K was increased while using the l2 norm the error rate for the test sets with darker lighting increased while the error rate for test set 1 stayed the same/decreased. This may be because the top 3/5 images from the darker test sets are those with similar lighting to the training image rather than similar facial features. Test set 1 was improved when K = 5 because the images in this set were well lit, so the majority label depended more on facial features rather than the similar effects that certain lighting conditions had. The l1 norm had a slight positive effect on predicting the poorly lit images from test sets 2-4, and a negative effect on images from set 1. When using the l1 norm large differences between two images were weighted equally to small differences, in contrast the l2 norm weighted larger differences higher than smaller ones. I think that using the l2 norm may have filtered out correct matches because differing lighting conditions between two images of the made more drastic changes to the returned norm.

3.1) In order to get the top k eigenvectors of the image, the total mean of the training set is subtracted from the training images to obtain the matrix D. Using the matlab svd function the top k eigenvalues of D are obtained from V.

3.2)



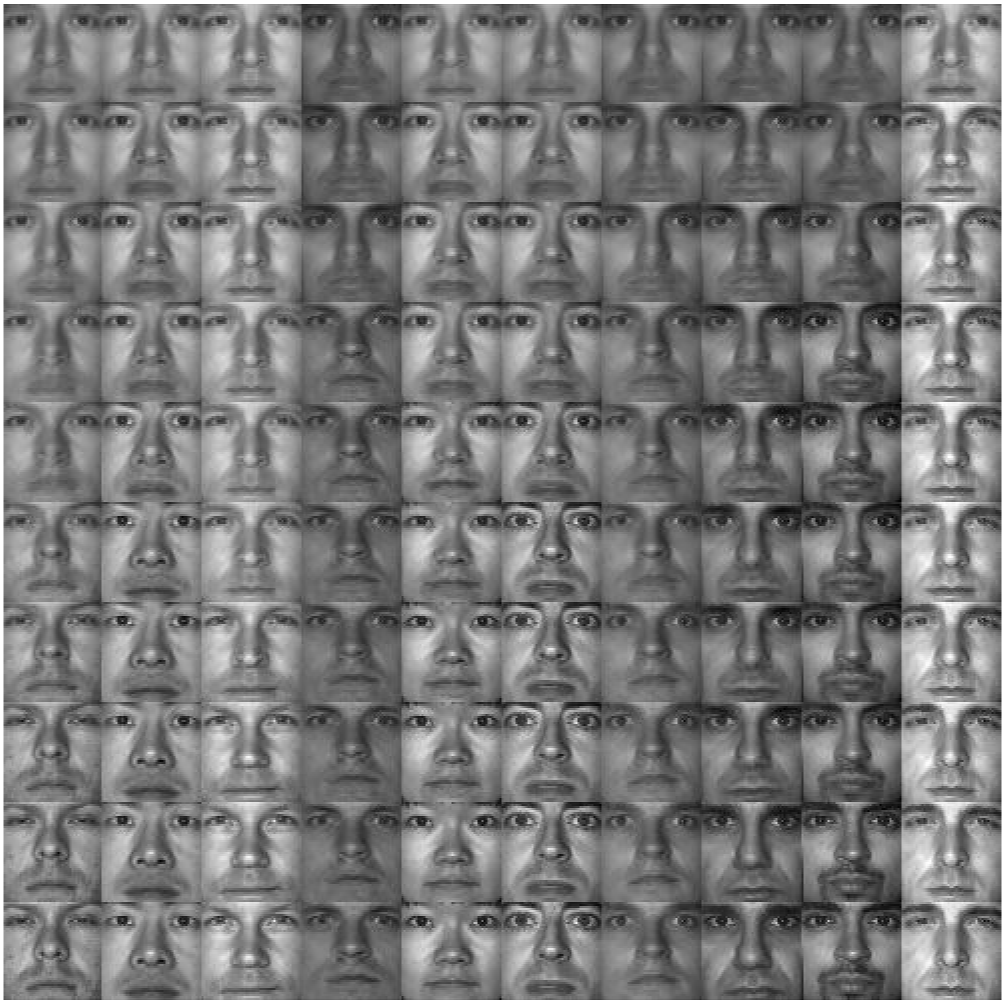
Top 20 eigenvectors

3.3) PCA identifies features where there is a lot of variance. The eigenvectors can be used to approximate an eigenface, which resembles a generic face.

```
[W,mu] = eigenTrain(trainset,20);
for k = 1:20
    W(k,:) = W(k,:) ;
end
imshow(drawFaces(W,10),[]);
```

```
%train eigenfaces
function [W,mu] = eigenTrain(trains,k)
    [trainRow,trainCol] = size(trains);
    D = zeros(size(trains));
    mu = mean(trains);
    for r = 1:trainRow
        D(r,:) = trains(r,:) - mu;
    end
    [u s v] = svd(D,'econ');
    W = v(:,1:k);
    W = transpose(W);
end
```

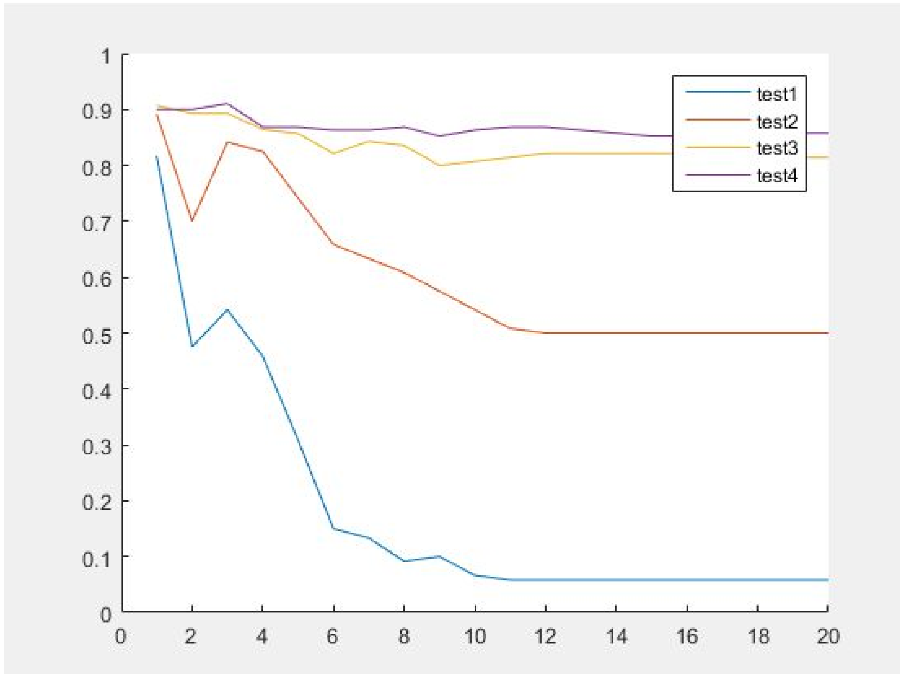
3.4) When k = 1 the faces look similar and they differ as k increases



People = zeros(100,2500)

```
%draw with different k's
function peop = drawPeople(set)
people = zeros(200,2500);
    for kp = 1:20
        [Wp,mup] = eigenTrain(set,kp);
        for p = 1:10
            people(p+(kp-1)*10,:) = (set(p*7,:) - mup)*Wp'*Wp;
        end
    end
    imshow(drawFaces(people+mup,10),[]);
end
```

3.5) Top values



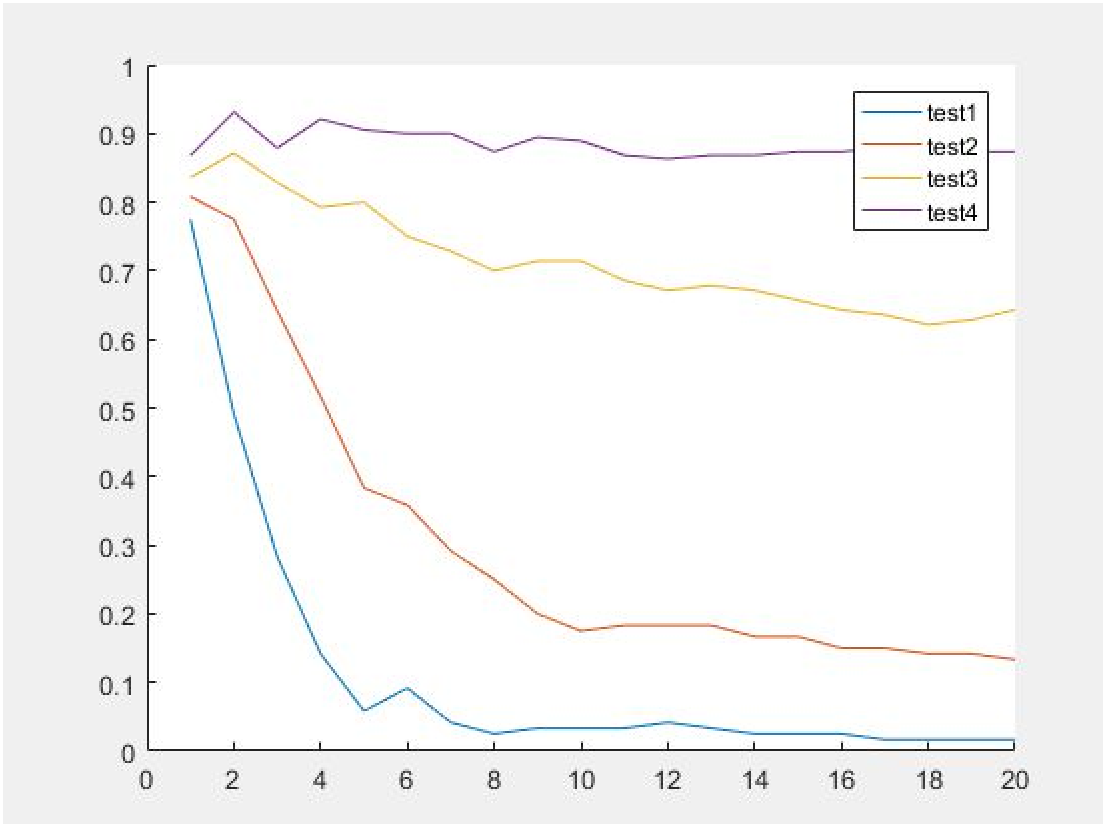

```
testerr = zeros(4,20);
kplot = [1:20];
for set = 1:4
[testset, testlabels] = loadSubset(set);
    for k = 1:20
        [W,mu] = eigenTrain(trainset,k);
        testerr(set,k) = eigenTest(trainset,trainlabels,testset,testlabels,W,mu,k);
    end
    hold on;
    plot(kplot,testerr(set,:))
end
legend('test1','test2','test3','test4');

```

```
function err = eigenTest(trains,trainlabels,tests,testlabels,W,mu,k)
    [trainRow,trainCol] = size(trains);
    [testRow,testCol] = size(tests);
    pTrain = zeros(size(trains));
    pTest = zeros(size(tests));
    for R = 1:trainRow
        pTrain(R,:) = (trains(R,:) - mu)*W'*W;
    end
    for R = 1:testRow
        pTest(R,:) = (tests(R,:) - mu)*W'*W;
    end
    testNorm = calcNorms(pTest,pTrain);
    pred = KNN(pTest,pTrain,trainlabels,testNorm,1);
    err = errRate(pred,testlabels);

end
```

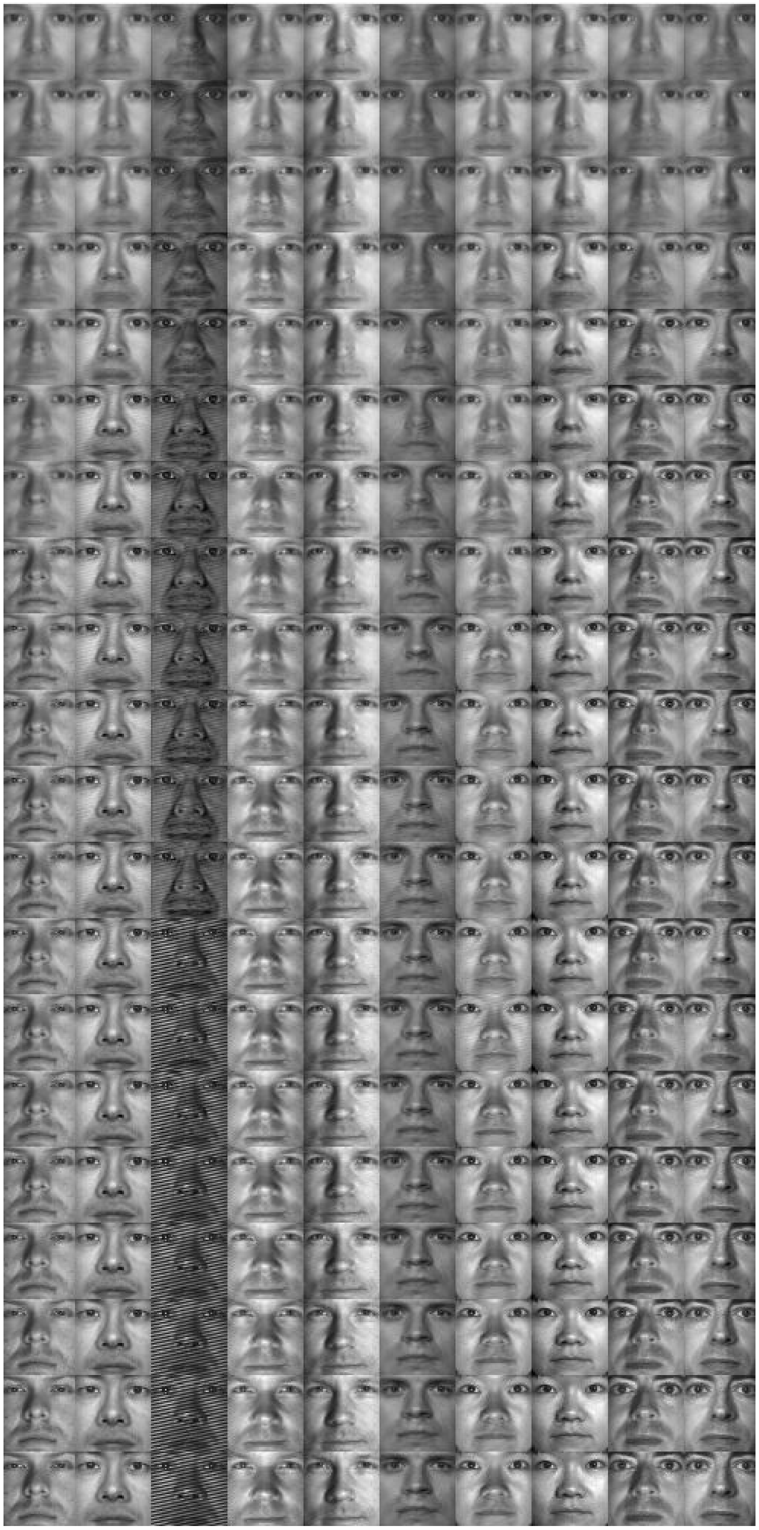
3.6) Top 4 thrown out



3.7) The eigenfaces with the top 4 eigenvectors thrown out performed better that those with the top 4 still there. This may be because the reconstructed faces (below) had more uniform lighting when eigenvectors were thrown out. The higher error rates in the later training sets can be explained by dark patches in the reconstructed images and the differences between the faces becoming less pronounced.

All eigenvalues

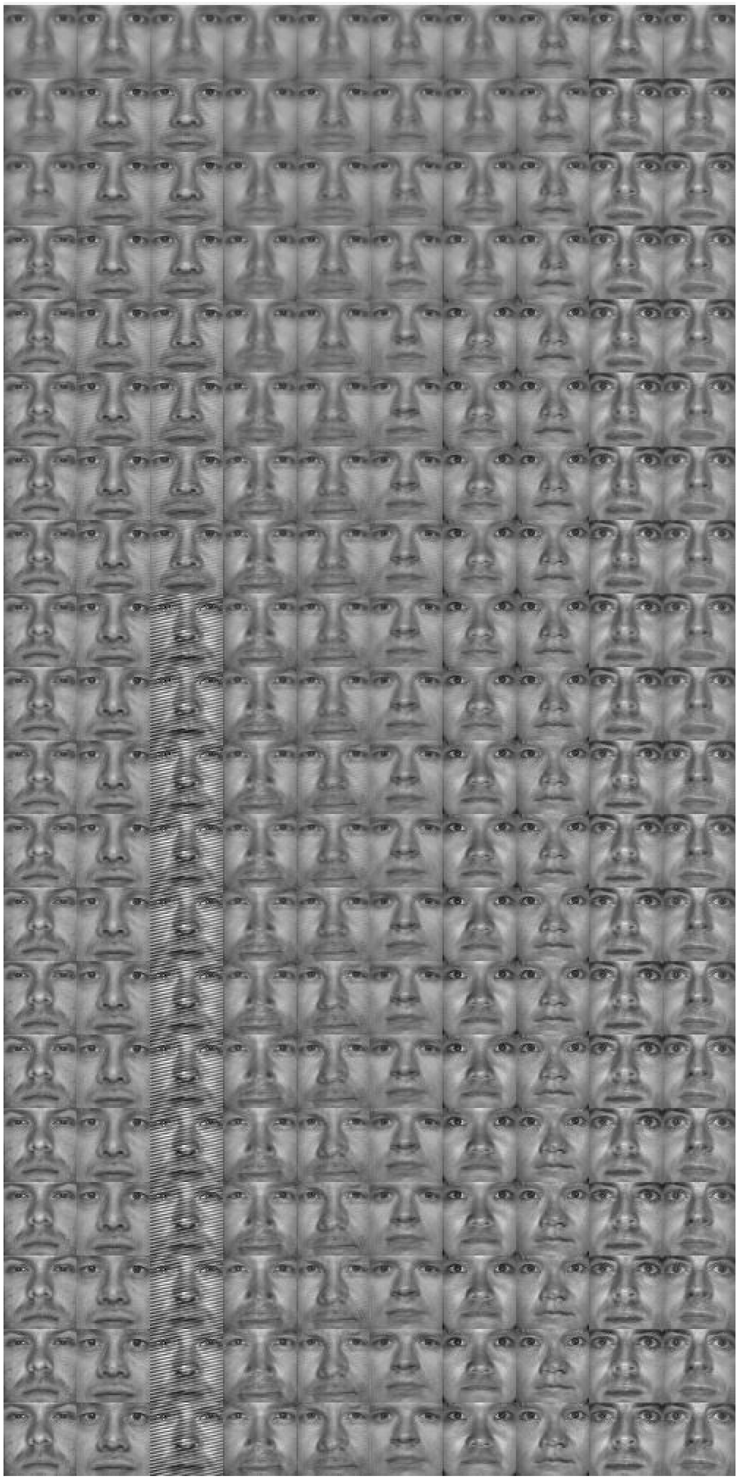
Test set 1



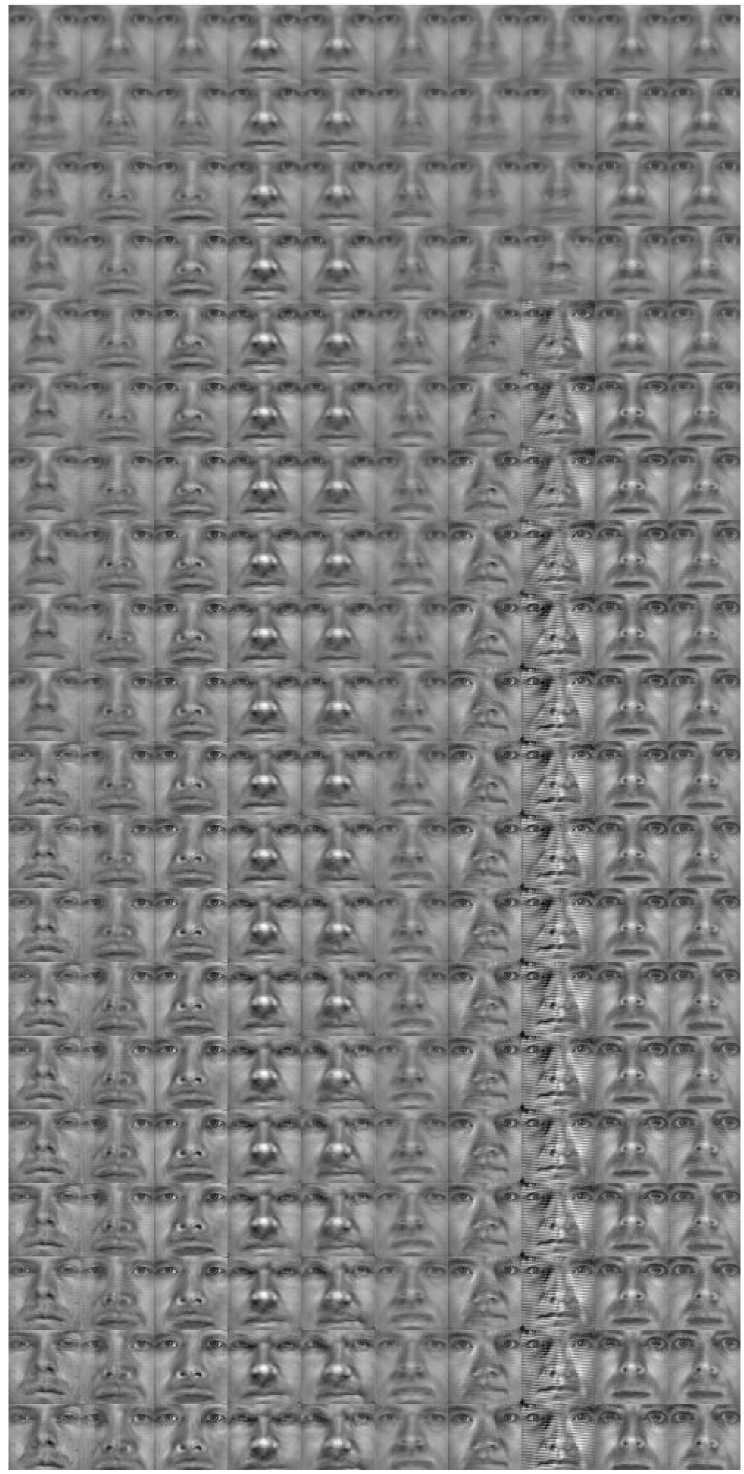
Test set 2

Throw out top 4 eigenvalues

Test set 1



Test set 2



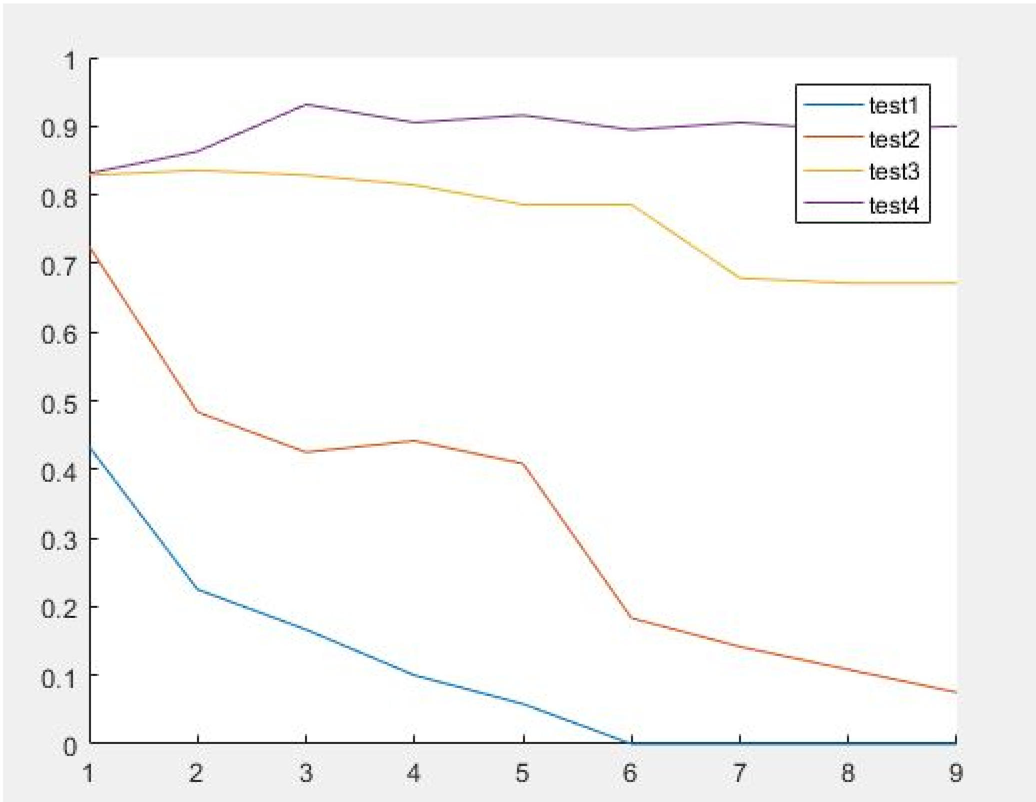
4.1) In order to calculate W for the fisherfaces, first Wpca of the image was computed with k = 60. Using this Wpca, the down projected training set was computed. With the projected training set Wp, Sb and Sw were calculated and used to calculate Wfld, the top 9 eigenvectors of the two. Finally W for the fisherface was computed by multiplying Wfld and Wpca.

```
%fisherfaces
function [W,mup]=fisherTrain(trains,trainlabels,c,k)
    [trainRow,trainCol] = size(trains);
    Nc = trainRow-c;
    [Wp,mup] = eigenTrain(trains,Nc);
    Wpca = Wp;
    pTrain = zeros(size(trains));
    for R = 1:trainRow
        pTrain(R,:) = (trains(R,:) - mup);
    end
    Wp = (pTrain)*transpose(Wp);
    muw = mean(Wp);
    %calculate Sb
    Sb = zeros(Nc,Nc);
    for class = 1:c
            m = mean(Wp(class*7-6:class*7,:));
            Sb = Sb + 7*(m-muw)'*(m-muw);
    end
    Sb
    %calculate Sw
    Sw = zeros(Nc,Nc);
    for sw = 1:trainRow
        m = mean(Wp(floor((sw-1)/7)*7+1:floor((sw-1)/7)*7+7,:));
        Sw = Sw + (Wp(sw,:)-m)'*(Wp(sw,:)-m);
    end
     [V,D] = eig(Sw,Sb);
     Wfld = V(:,1:k);
     W = Wfld'*Wpca;


end
```

4.2) The images look well defined with uniform lighting. The features are more distinct than those of eigenfaces.

4.3) fisher predictions (mostly same code as graphing eigenfaces)



Compared to eigenface, Fisherface showed a large improvement for test sets 1 and 2 as k was increased and a slight improvement for test set 3. This is because the fisherfaces had very well defined facial features returned. Test set 1 had good lighting so the error rate went to 0 at k = 6 and test set 2 which had poorer lighting took longer to decrease in error rate. The lighting for test set 4 was horrible so that the fisherface method wasn't enough to save it.