EIGENFACES VS FISHERFACES

EE417 Computer Vision – Course Project

Abstract

Face is a multidimensional visual model which has an important impact on human relationships. Face has a very complex structure which makes it harder to recognize faces for computational models. However, it is very important to develop successful algorithms for computer face recognition since it has various essential applications. This project will focus on the implementation and comparison of two well-known algorithms, respectively Eigenfaces and Fisherfaces.

Keywords: Face Recognition, Eigenfaces, Fisherfaces

Gökberk Yar,Oğuz Çelik

Supervisor: Prof.Dr.Mehmet Keskinöz

Table of Contents

Problem Importance and Problem Definition

Face is one of the most important parts of the human body, it is the primary component to show emotions, so it has a significant impact on human relationships. Humans are very successful at learning and identifying faces. A human being can learn thousands of faces during a lifetime and still be able to remember most of them [1]. Unfortunately, it is not possible to say the same thing applies for computers. Computational face recognition can be stated as given a labeled set of human faces and an unlabeled set of same faces, identify each face in the unlabeled set using the labeled data. For example, assume there are 100 faces in labeled set(training) and let's say some of these faces belong to person X, the aim is to detect the face that belongs to X in the unlabeled(test) set. Face recognition is an important task which has a wide range of applications such as surveillance, criminal investigation, human-computer interaction. For example, if emotions and expressions can be detected through face recognition then it would be possible to build devices which interacts with a person based on his/her mood [2]. Unfortunately, face recognition is a very difficult problem and despite the fact that there have been many proposed algorithms, there is still need for a better one. Most of these algorithms perform well under small changes in lighting, face expression and pose however they fail under more extreme conditions. P. Belhumeur, J. Hespanha, and D. Kriegman developed an enhanced solution(FisherFaces algorithm) [1] to successfully recognize faces under advanced variations compared to EigenFaces algorithm which was proposed by M. Turk and A. Pentland [3]. In this report, EigenFaces and FisherFaces will be compared and discussed.

Problem Formulation

A square grayscale image of size NxN can be represented as N^2x1 column vector or any rectangular MxN image can be represented as $(NxM)x1$ column vector for the sake of generality images assumed to be square in this paper, although they don't have to be. Image recognition can be formulated as follows given a sample X of M square images $X = \{X_1, X_2, ..., X_M\}$, where each $X_i = [X_{i1}, X_{i1}, ..., X_{iN^2}]^T$ and each X_i has a label y_i where each $y_i \in C$, $C = \{C_1, C_2, ..., C_c\}$ represents classes and c is number of different classes, predicting the label of a sample X_t where y_t is not known. $\hat{y}_t$ will be predicted class for the label of X_t. $\hat{y}_t$ will be predicted as the class which X_t has the smallest distance to class's decision boundary. Mathematically, $\hat{y}_t = \min_{c \in C} \| B_c - X_t \|$, B_c is the class boundary for class c .Distance models and methods to create a decision boundary for each class will be discussed in the next chapter.

Solution Method

EigenFace

One way of predicting $\hat{y}_t$ is through correlation measurement. Correlation can be calculated using kNN however, training set will be normalized to 0 mean and unit variance in correlation calculation, this will make the model independent of illumination effect. When test images do not come from the same illumination intensities, correlation model will fail to classify correctly unless trainset contains continuum of intensity levels [1]. Also calculation of correlation is quite expensive since each image is N^2x1 vector and there exist M samples at least N^2xM operation has to take place for each test element. Although there exists algorithms and dedicated VLSI hardware for kNN computation, it is computational heavy for real time purposes [4]. Beside the complexity of calculation, not all features in the image space is equally strong to explain variance of the training set. One way to decrease complexity of algorithm and extracting relevant features which explains the variance strongly is to make a Principal Component Analysis (PCA). PCA projects N^2x1 vector to much smaller space $mx1$ through a linear transformation. Mathematically,

$$P_k = W^T X_k \ k = 1,2,...M$$

P_k is the projected version of the X_k, P_k has $mx1$ dimension. $W \in \mathbb{R}^{mx(N^2)}$ .In order to calculate W we have to calculate the covariance matrix S_t of M samples.

$$S_t = \sum_{i=1}^{M}(X_k - \mu)(X_k - \mu)^T,$$

$\mu \in \mathbb{R}^{N^2}$ is the mean of training samples. When X is projected with W matrix result will be $P = [P_1, P_2, ..., P_M]$. P is also equal to $W^T S_T W$. We need to maximize the determinant of $W^T S_T W$ to get best W that explains the variance of the training set. Formally,

$$\begin{aligned} W_{opt} &= \arg\max_W |W^T S_T W| \\ &= [w_1, w_2, ..., w_m] \end{aligned}$$

w_i is eigenvector corresponding ith largest eigenvalue of S_t. So W_{opt} consist m eigenvectors corresponding the m largest eigenvalues of S_t. For training, W will be calculated and X is project to P with $P = W^T X$ and for testing data X_t will be centered by subtracting μ of the training and project into PCA space to P_t with calculated W then using a kNN and distance model, distance between P_t and each member P will be calculated and $\hat{y}_t$ classified according to kNN.

For distance measurement Euclidean or Mahalanobis distance can be used.

$Euclidean = \sum_{i=1}^{m}(P_t - P^k{}_i)^2$, distance between kth training sample and test sample.

$Mahalanobis = \sum_{i=1}^{m}\frac{1}{\lambda_i}(P_t - P^k{}_i)^2$, λ_i is eigenvalue for the ith dimension it gives equal weights to all dimensions.

Figure 1: Yale Dataset PCA Components

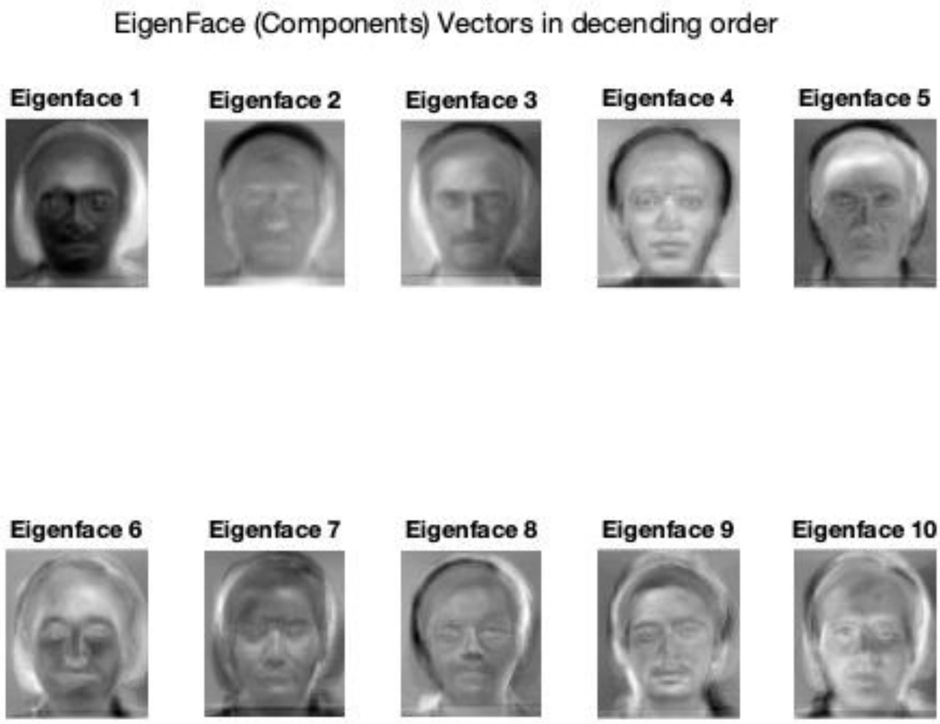


Figure 2: Yale Dataset Projected into PCA space

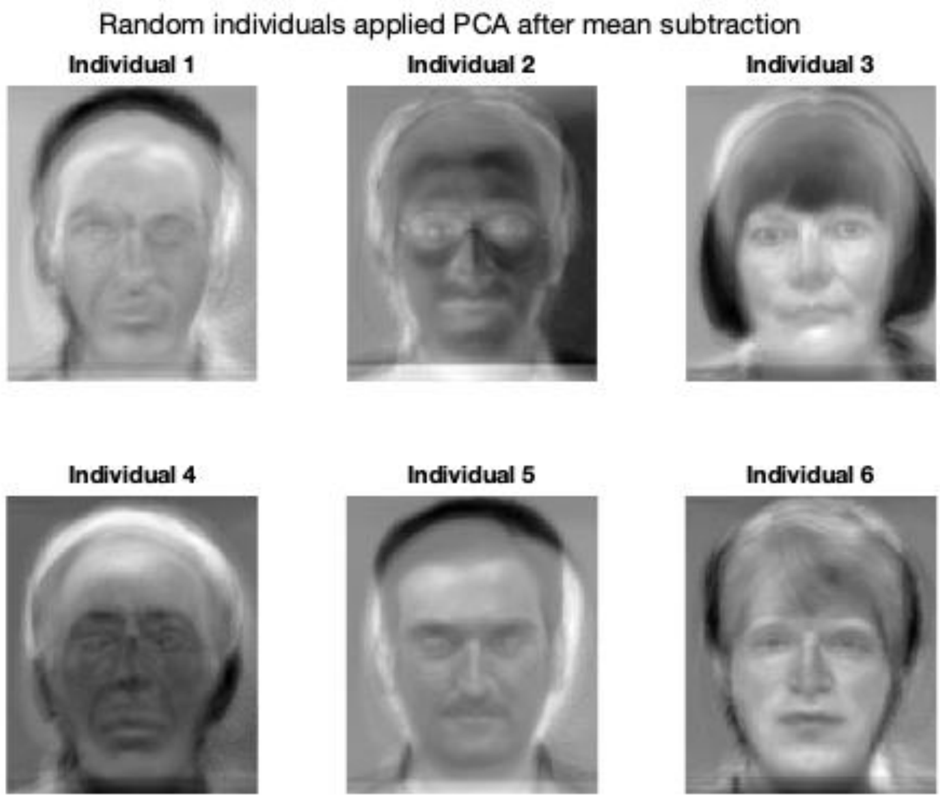


Figure 3: AT&T Dataset PCA Components

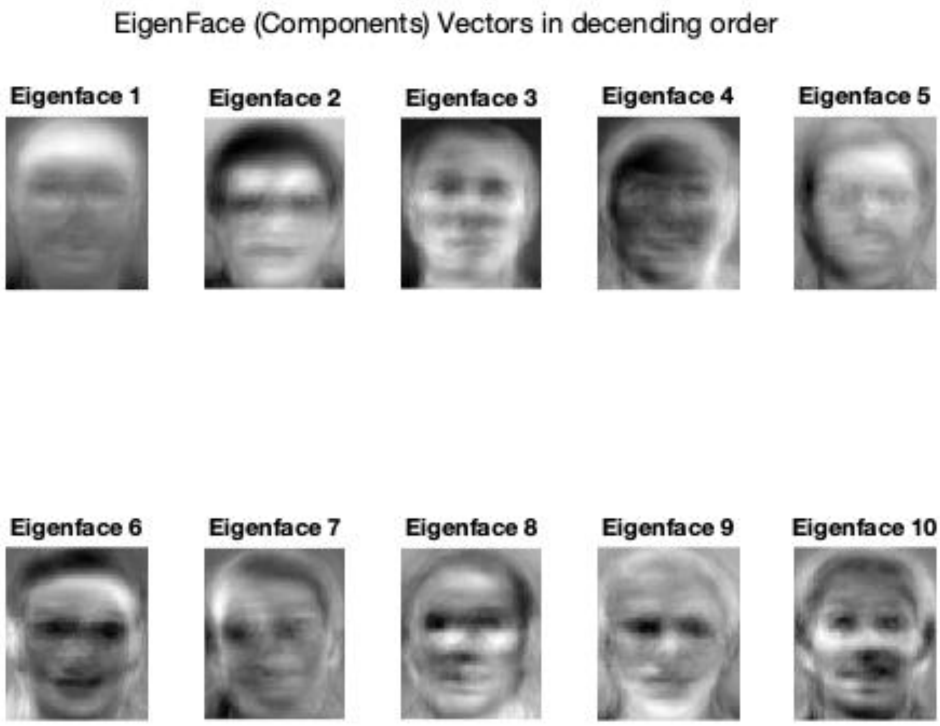
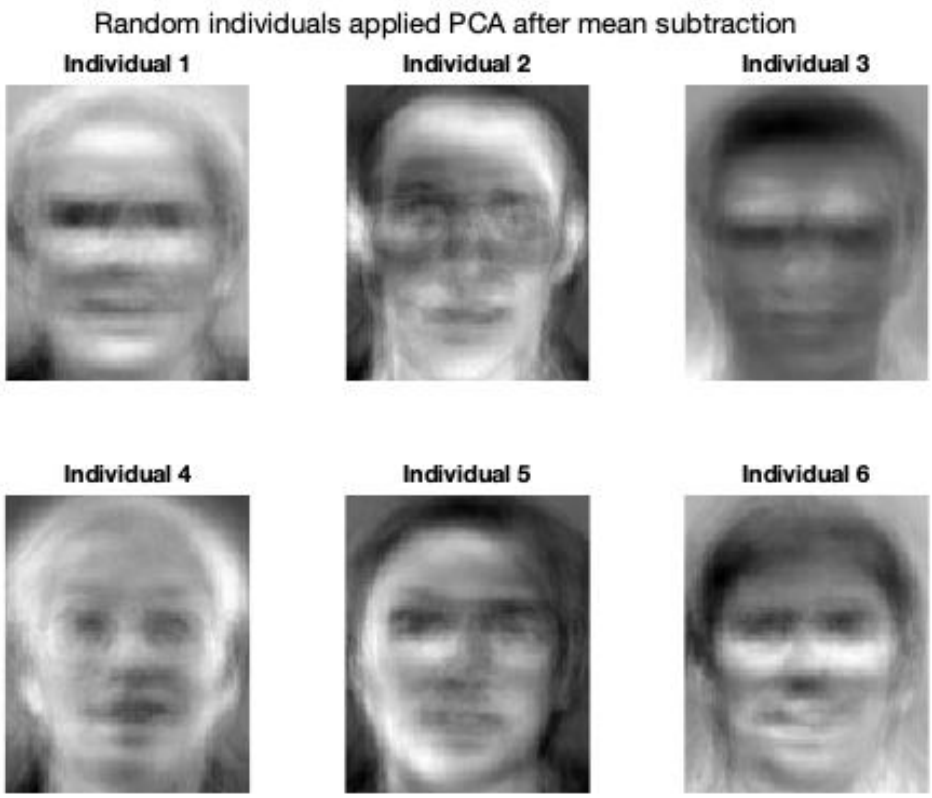


Figure 4: AT&T Dataset Projected into PCA Space



EigenFace Without 3

This method uses the same approach as the Eigenface with a slight modification. In this method during the calculation of W biggest 3 eigenvalues and eigenvectors corresponding to them are discarded. In order to have same number of principal components, principal component count m should be increased by 3 since first 3 of them will be discarded. Formally,

$W_{opt} = \arg\max_W |W^T S_T W|$

$= [w_1, w_2, \ldots, w_m, w_{m+1}, w_{m+2,} w_{m+3}]$

$W^{new}{}_{opt} = [w_4, w_5, \ldots., w_m, w_{m+1}, w_{m+2}, w_{m+3}]$

$W^{new}{}_{opt}$ will be used for linear transformations in this method. The intuition behind discarding the first 3 components is the practical evidence which suggests that first 3 component is highly related to illumination [5]. Although they contain valuable information and there is not clear distinction between illumination and other features, EigenFace without 3 is used in applications [1].

FisherFace

EigenFace method does no distinction on how features spread within class or between classes when features are projected into subspace. Projecting into much smaller subspace without considering the class could result as losing relevant features to distinguish classes. To solve this problem, Linear Discriminant Analysis (LDA) also known as Fischer Discriminant Analysis (FDA) [6] can be used. With LDA X will be projected into a subspace to P with $P = W^T X$ linear transformation where W is projection matrix. W will maximize projected ratio between class covariance S_B and within class covariance S_W .This can be formulized as follows,

$$S_B = \sum_{i=1}^{c} N_i (\mu_i - \mu)(\mu_i - \mu)^T$$

$$S_W = \sum_{i=1}^{c} \sum_{x_k \in X_i} (x_k - \mu_i)(x_k - \mu_i)^T$$

This equations define between and within class ratio, where μ is mean of all training samples and μ_i is the mean of the class i and N_i is the number of samples for the that class.

$$W_{opt} = \arg\max_W \frac{|W^T S_B W|}{|W^T S_W W|}$$

$$= [w_1, w_2 \ldots, w_m]$$

W_{opt} maximize discussed ratio, and solution for the W_{opt} is generalized eigenvalues for S_B and S_W .Formally,

$$S_B w_i = \lambda_i S_W w_i \quad i = 1, 2, \ldots, m$$

Due to limitation of generalized eigenvalue composition only $c-1$ generalized eigenvalues can be found [7].

However, S_W is a $N^2 x N^2$ matrix where N is the dimension of the image. A very modest estimation for image size goes over 100 pixels which results a $10e4x10e4$ sized matrix, which will make

S_W almost always singular and requires extreme computational power. Due to this limitation FisherFace proposes to reduce the dimension to M-c with PCA then applying above discussed LDA analysis. W_{opt} for FisherFace defines as follows,

$$W_{opt}^T = W_{fld}^T W_{PCA}^T$$

$$W_{pca} = \arg\max_W |W^T S_t W|$$

$$W_{fld} = \arg\max_W \frac{|W^T W_{PCA}^T S_B W_{PCA} W^T|}{|W^T W_{PCA}^T S_W W_{PCA} W^T|}$$

Applying both PCA and LDA as discussed above W_{fld} will be calculated in training using X then projection $P = W_{fld}{}^T X$ will be calculated. In test, X_t will also be projected into P_t using W_{fld} and then using a distance measure and kNN, $\hat{y}_t$ will be predicted. Part after projection is exactly the same as EigenFace, way that they differ is to pick W_{opt}.

Implementation of PCA

Calculation of eigenvalues, eigenvectors λ_i, u_i of S_t are computationally very expensive. Since $S_t = \sum_{i=1}^{M} (X_k - \mu)(X_k - \mu)^T$ results a matrix of size $N^2 x N^2$, instead calculate eigenvalues , eigenvectors λ_i, v_i of $Q_t = \sum_{i=1}^{M} (X_k - \mu)^T (X_k - \mu)$. They share the same eigenvalues and also their eigenvectors are linearly connected with $u_i = [X_1 - \mu, X_2 - \mu, ..., X_M - \mu] \cdot v_i$ [3].

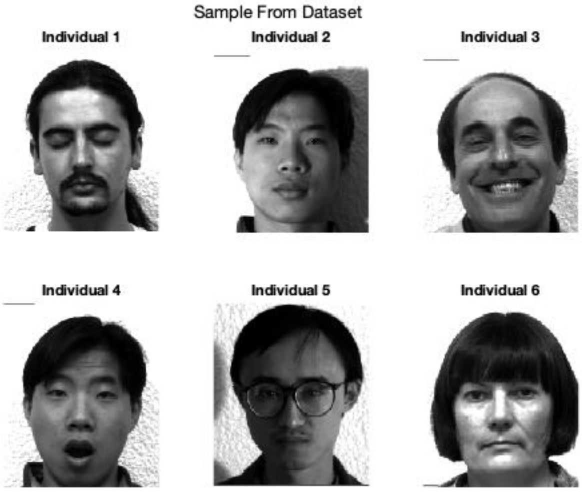
Results

Datasets

Yale

The dataset is originally created by Yale Vision Group but for this project it is downloaded from Kaggle. Dataset contains 165 GIF images of 15 subjects. Each subject has 11 images and each one of these 11 images focuses on a different variation. Each subject has images with respect to following variations: center-light, w/glasses, happy, left-light, w/no glasses, nor- mal, right-light, sad, sleepy, surprised, and wink. This dataset has a version with images are centered with using eye location information and a standard one, centered one used in this paper. The main characteristic of dataset is the fact that it contains lots of illumination differences.

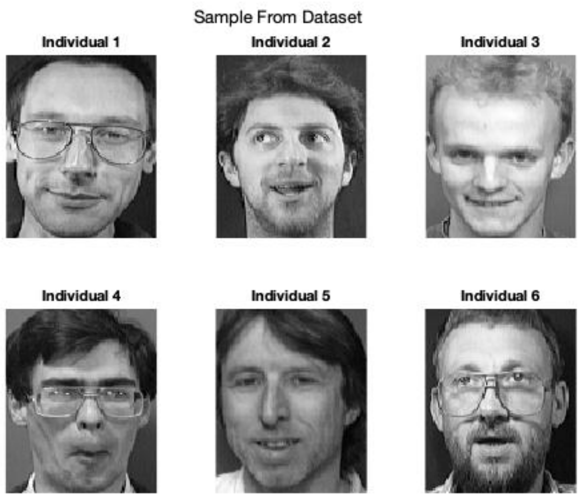
Figure 5: Sample from Yale Faces Dataset



AT & T

AT&T Dataset is created by AT&T Laboratories Cambridge and it contains images that are taken between April 1992 – April 1994. Images are in PGM format and each image is size of 92x112 pixels and there are 256 grey levels per pixel. Dataset has 400 images of 40 different subjects. Each subject has 10 images and some images are taken at different times to create variations such as lighting, facial expressions (open / closed eyes, smiling / not smiling) and facial details (glasses / no glasses). Also, in every image the subject is standing in a frontal position in front of a dark homogenous background. Illumination changes in this dataset is less compared to Yale Dataset. Images are not well centered as the Yale Dataset.

Figure 6: Sample from AT&T Faces Dataset



Tables

Yale

Figure 7: Table for Results on Yale Faces Dataset

Yale Faces Dataset				
EigenFace	Mahalanobis	numComp=10	K =1	19,39
			K=5	29,09
		numComp=30	K =1	**13,33**
			K=5	29,09
	Euclidean	numComp=10	K =1	19,39
			K=5	30,30
		numComp=30	K =1	18,18
			K=5	26,67
EigenFaceWO3	Mahalanobis	numComp=10	K =1	13,33
			K=5	27,27
		numComp=30	K =1	**11,52**
			K=5	29,09
	Euclidean	numComp=10	K =1	14,55
			K=5	23,64
		numComp=30	K =1	12,12
			K=5	23,64
FisherFace	Mahalanobis	numComp = class -1	K =1	9,09
			K =5	10,90
	Euclidean		K =1	**0,00**
			K =5	**0,00**

AT & T

Figure 8: Table for Results on AT&T Faces Dataset

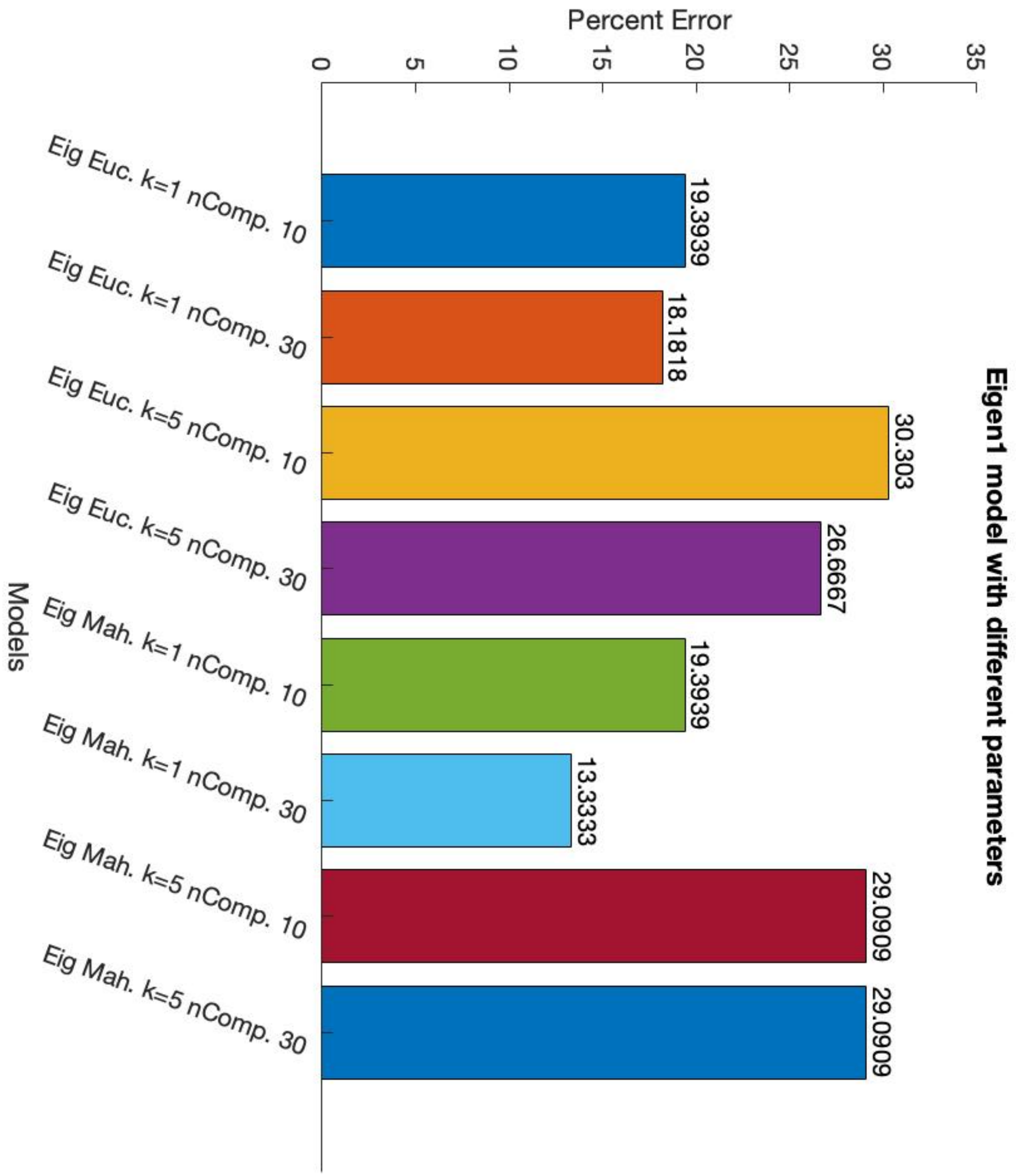
At&t Faces Dataset				
EigenFace	Mahalanobis	numComp=10	K =1	5,25
			K=5	34,00
		numComp=30	K =1	4,00
			K=5	26,25
	Euclidean	numComp=10	K =1	3,75
			K=5	34,00
		numComp=30	K =1	**1,75**
			K=5	26,25
EigenFaceWO3	Mahalanobis	numComp=10	K =1	11,25
			K=5	41,50
		numComp=30	K =1	4,75
			K=5	35,50
	Euclidean	numComp=10	K =1	12,25
			K=5	43,00
		numComp=30	K =1	**4,25**
			K=5	36,00
FisherFace	Mahalanobis	numComp = class -1	K =1	20,75
			K =5	24,00
	Euclidean		K =1	5,50
			K =5	**5,25**

Graphs

Yale

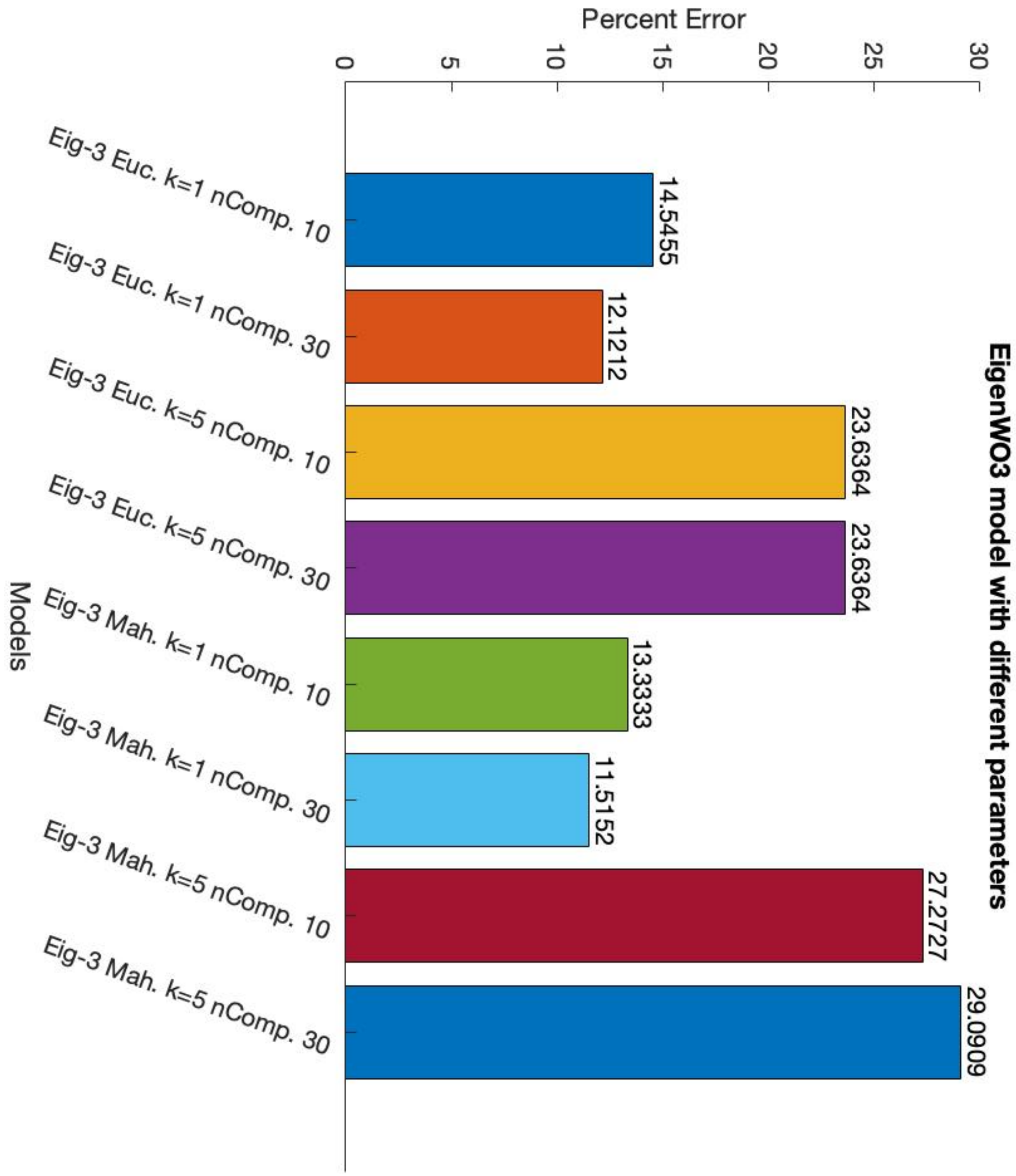
EigenFace

Figure 9: Results of EigenFaces Algorithm on Yale Dataset



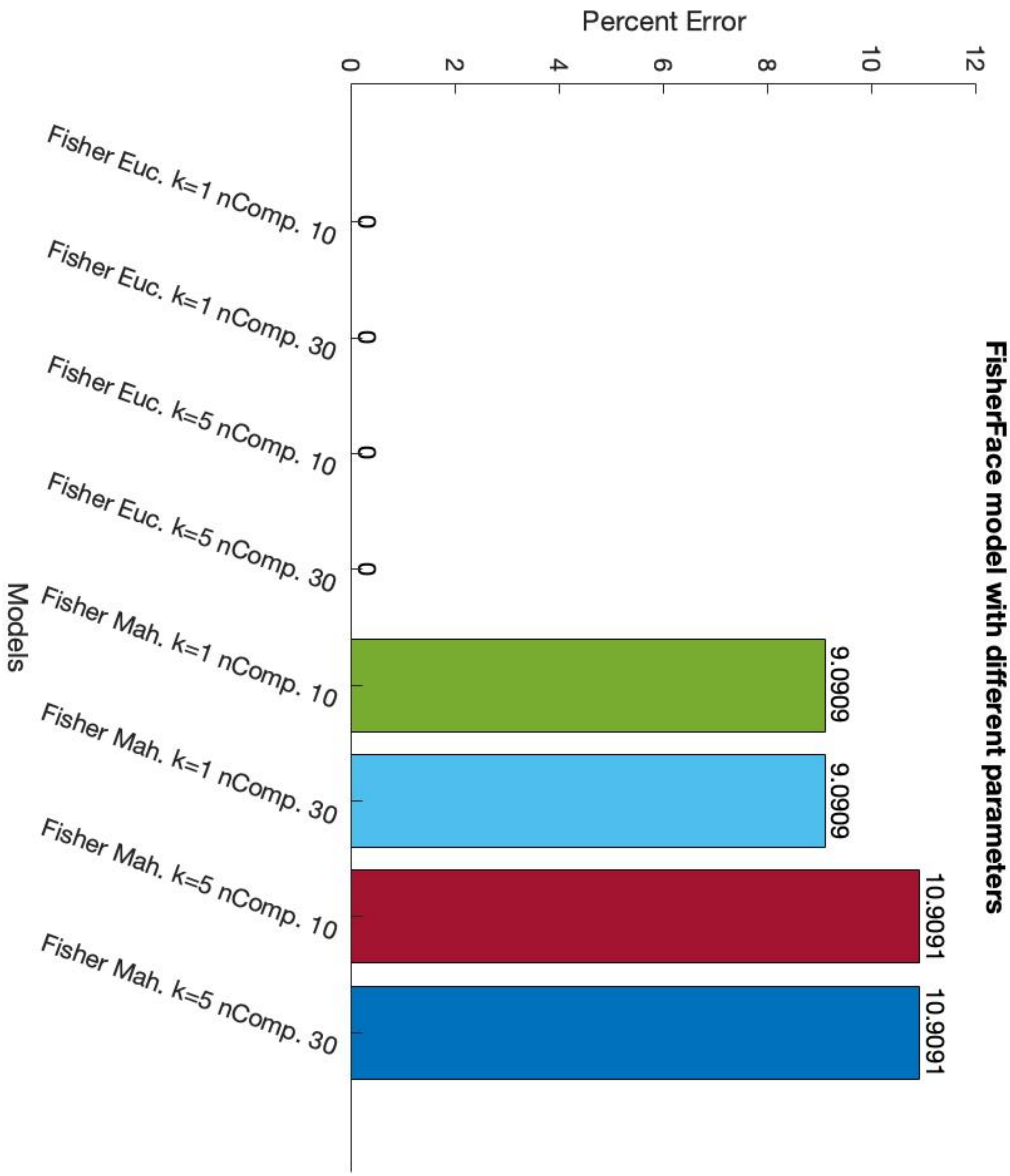
[EigenFace Without 3]

Figure 10: Results of EigenFacesWO3 Algorithm on Yale Dataset



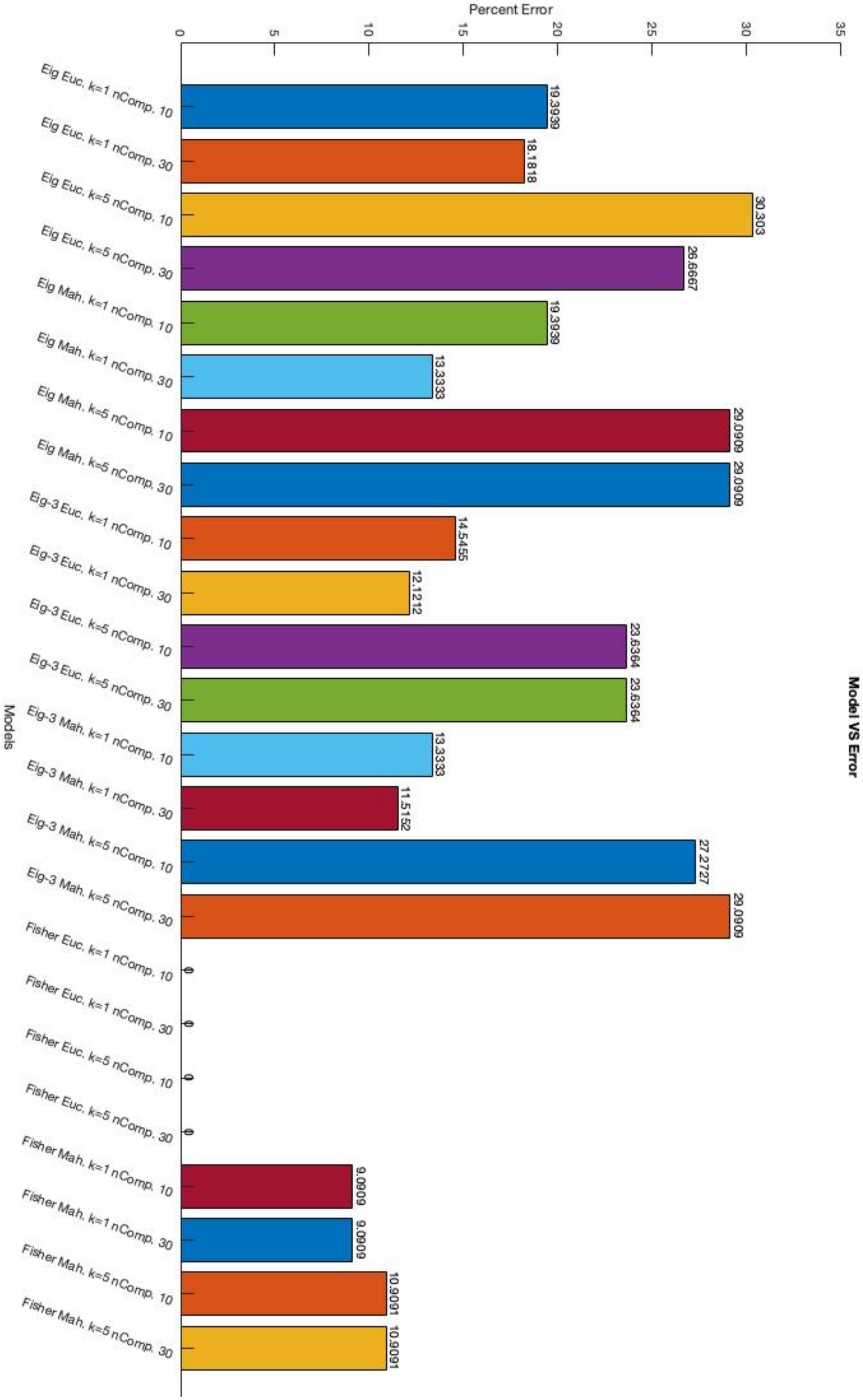
FisherFace

Figure 11: Results of FisherFace Algorithm on Yale Dataset



All

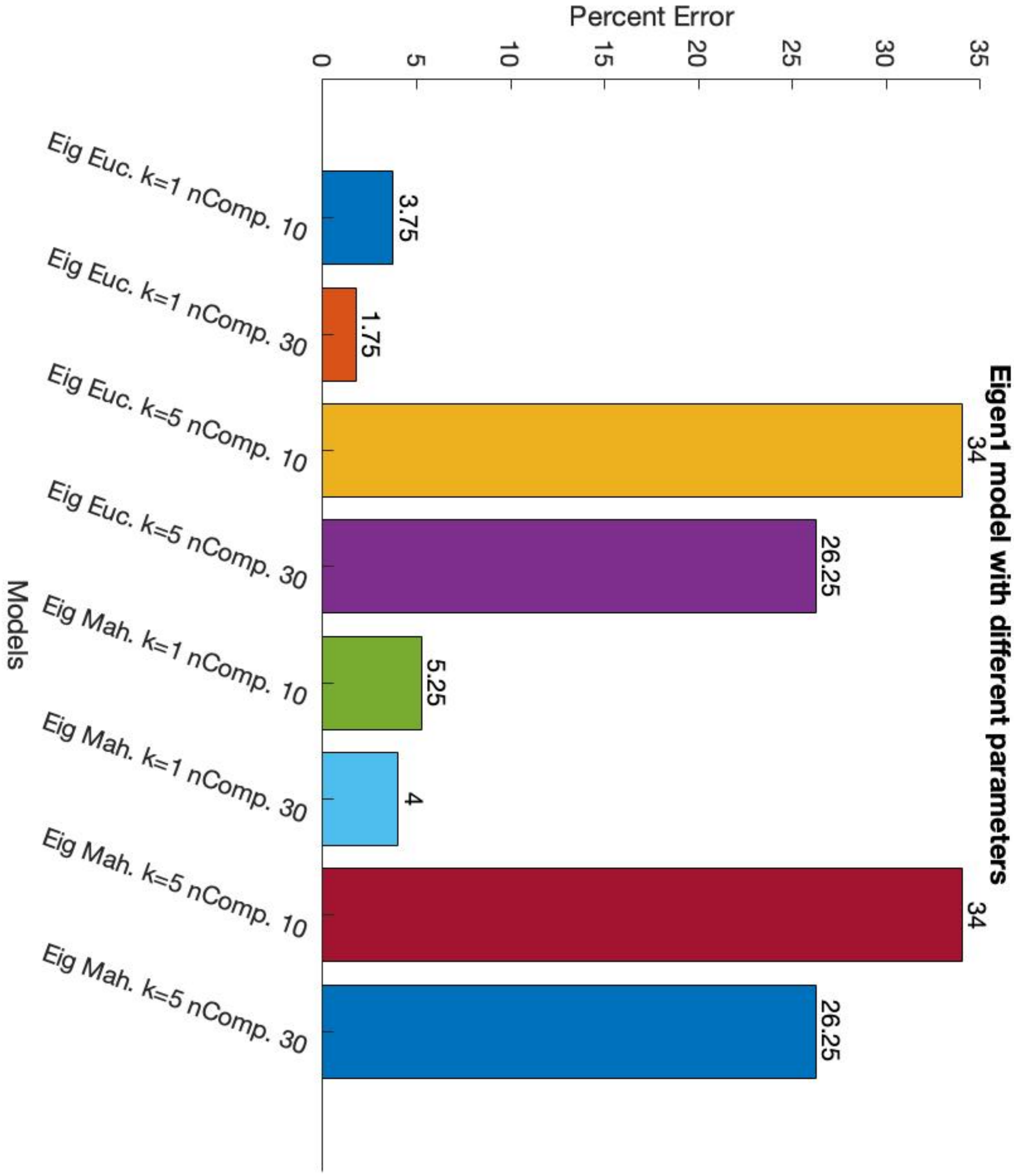
Figure 12: Results of All Algorithms on Yale Dataset



AT & T

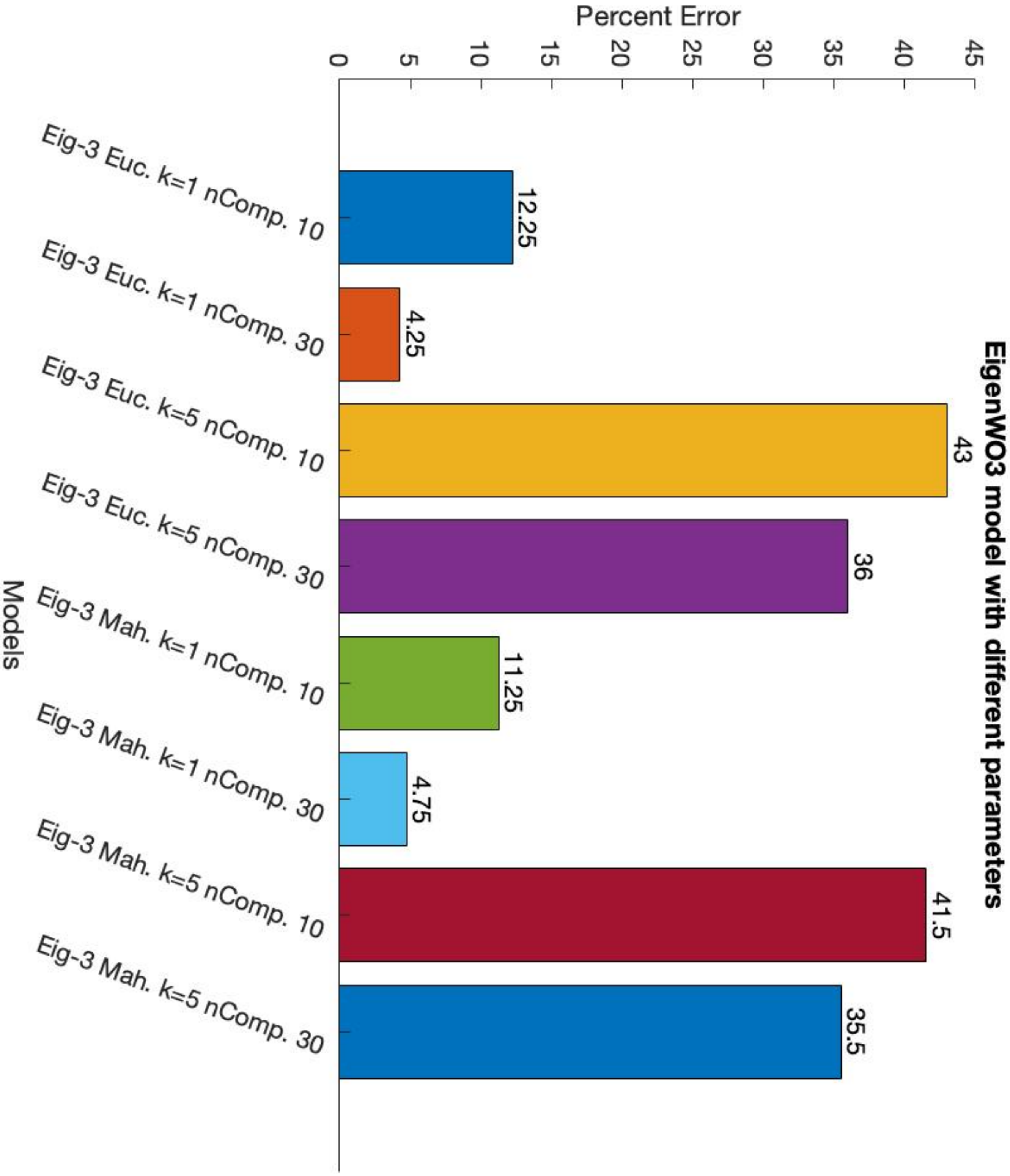
EigenFace

Figure 13:Results of EigenFaces Algorithm on AT&T Dataset



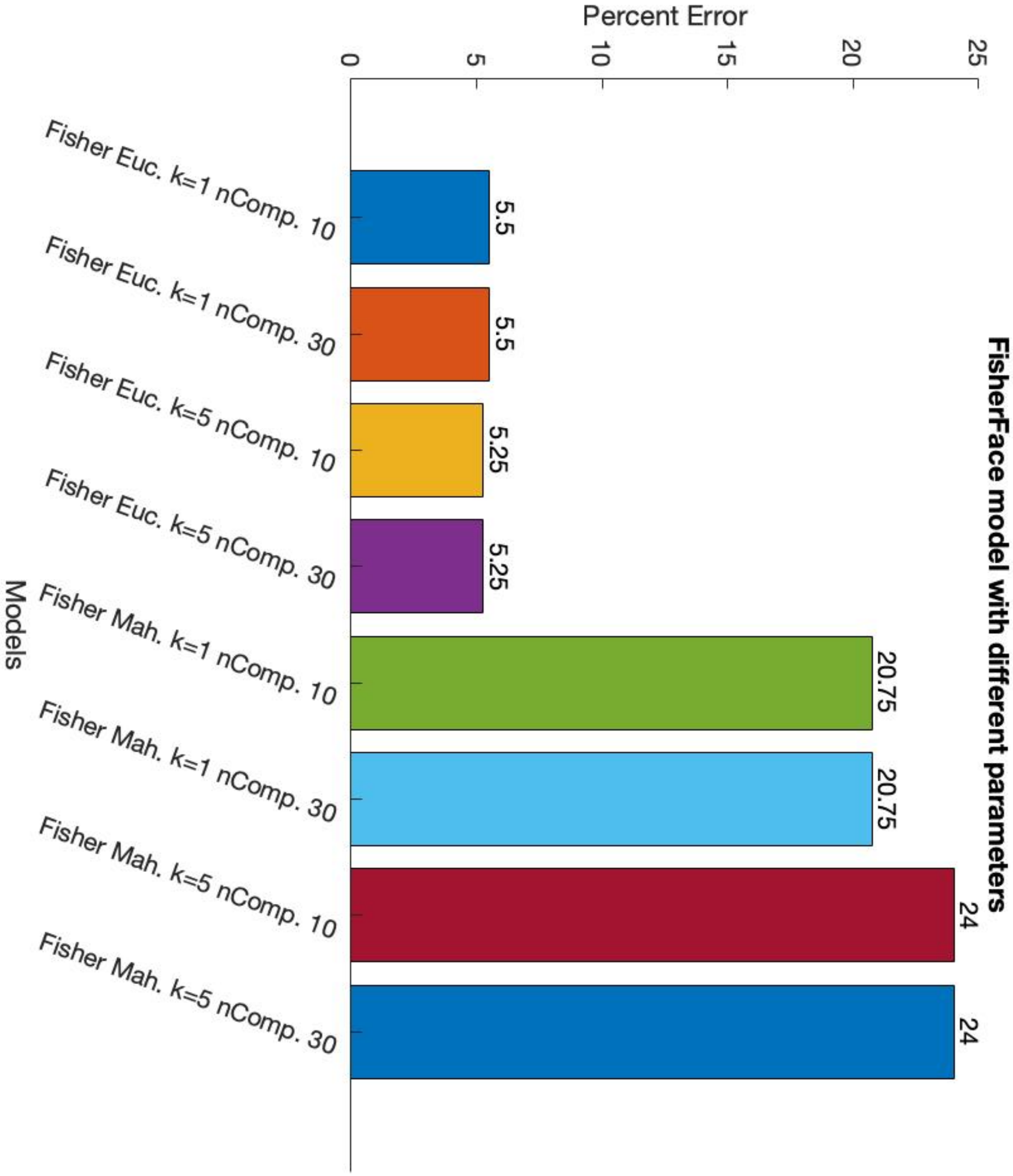
EigenFace Without 3

Figure 14: Result of EigenFacesWO3 Algorithm on AT&T Dataset



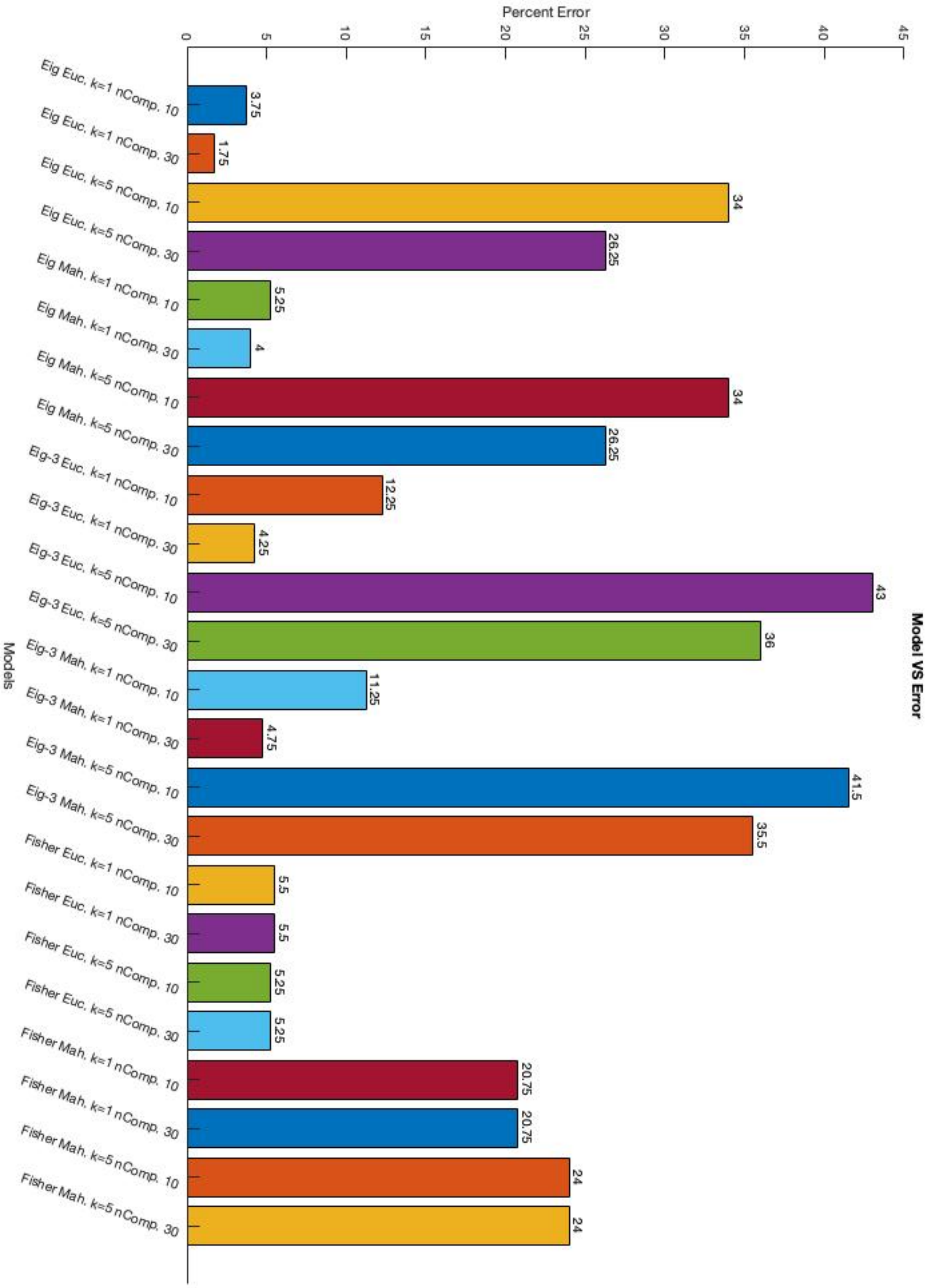
FisherFace

Figure 15: Result of FisherFaces Algorithm on AT&T Dataset



[All](#)

Figure 16: Result of All Algorithms on AT&T Dataset



Discussion

Yale

EigenFace

The smallest error is 13,33 and it is achieved by EigenFace model with Mahalanobis distance model, 30 PCA components and k=1 for KNN configuration. However, EigenFace is the worst model when it is compared to EigenFaceWO3 and FisherFace. From the results of EigenFace we may say Mahalanobis distance out performs Euclidean distance measure as suggested by M. Turk [3] since there is only 1 case, numComp = 30 and k =5, which implies otherwise and that one case performs very poorly . Increasing number of principal components increases accuracy or accuracy stays same for all the cases, this suggests 30 principal components explains variance of the dataset much better than the 10 principal components. Increasing the k seems to be not helping but it decreases the performance in all of the cases. It may suggest that class boundaries are very close to each other and using nearby points decreases the accuracy, so a better distinction between classes is needed.

EigenFace Without 3

Neglecting first 3 eigenvectors outperformed keeping them as in EigenFace, since there exist a good variation in illumination difference in the dataset. Smallest error is achieved by Mahalanobis Distance, numComp = 30 and k =1 which is same as the EigenFace model. It seems both models have similar behaviors to parameter changes, this could suggest first 3 components is not distinctive. Increasing the number of PCA components as 30 outperforms 3 out of 4 cases, it emphasizes 10 PCA components does not well explain the dataset. Problem with increasing the k still a problem for EigenFaceWO3 as well. It again emphasizes that decision boundaries are close to each other.

FisherFace

FisherFace outperforms other models and achieves zero error in 2 out of 4 setups which is remarkable. In FisherFace, it seems Euclidean Distance outperforms the Mahalanobis distance, this could happen because sample is already projected under consideration of their class within variance. Their number of components is fixed and always equal to c-1, in the graphs this identical process is repeated twice. Problem of k is not resolved, further investigation is needed to compare k values.

AT&T

EigenFace

In general AT&T database shows opposite characteristic as opposed to the Yale Dataset. EigenFace is the best model among 2 others but all models seem to perform considerably well in this model. Since illumination does not change a lot in this dataset its effects are limited. That's why EigenFace outperforms EigenfaceWO3 in 7 out 8 different setups. Both models have the lowest errors in the same setup again this emphasizes that the same situation happened in the Yale dataset which is first 3 PCA component are not distinctive for the model. Euclidean distance measure performs better in this model, Mahalanobis was better in the previous dataset so further investigation is needed. Increasing number of PCA components increased the model accuracy so again 30 principal components is a better choice compared to 10.

EigenFace Without 3

It performs best under Euclidean and numComp = 30 and k =1. The configurations for the best models are very similar for both of the datasets. In each configuration, principal component count and k value is same however, distance measure differs. Therefore, it is fair to say distance measure needs to be tuned for different datasets. It performs worse than the EigenFace model, this could be explained by small variation in the illumination levels.

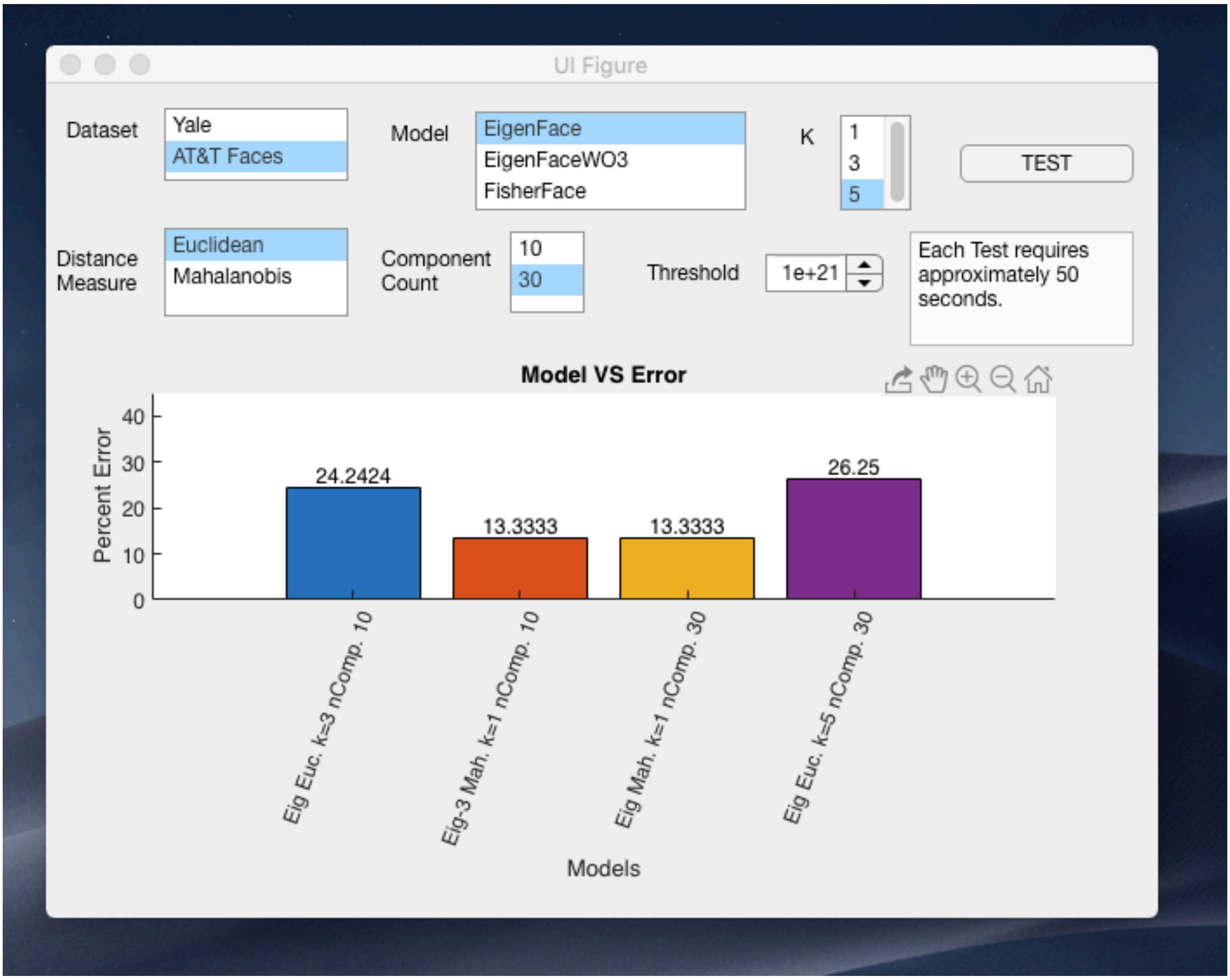
FisherFace

Even FisherFace performed remarkable with only 5 percent error. It is the worst one compared to EigenFace and EigenFaceWO3, yet for both datasets Euclidean as distance measure outperforms in each case. For the effect of k, further investigation is still needed,since there exists a tie between k=1 and k=5. FisherFace performed best in the same configuration and for each dataset success orders of the models are same, which implies FisherFace's characteristics could be dataset independent. Further research is needed to understand. Image centralization process is not as good as in the Yale dataset, its effects are unknown. Datasets with better alignment should be investigated.

GUI

We developed a MATLAB GUI for investigating performance under different setups below you can see the product and in the report file you can find setup.

Figure 17: GUI



Appendix

Bibliography

[1] P. N. Belhumeur, J. P. Hespanha and D. J. Kriegman, "Eigenfaces vs. Fisherfaces: Recognition Using Class Specific Linear Projection," *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE,* vol. 19, no. 7, p. 711, JULY 1997.

[2] J. Cabansag, . Y. Chen, J. Griffin, D. Lyons and G. Preudhomme, Lecture 13: Face Recognition and LDA, Stanford University.

[3] M. Turk and A. Pentland, "Eigenfaces for Recognition," *Journal of Cognitive Neuroscience,* vol. 3, no. 1, 1991.

[4] R. Brunelli ve T. Poggio, «Face Recognition: Features vs. Templates,» *IEEE Trans. Pattern Analysis and Machine Intelligence,* cilt 15, no. 10, pp. 1,042-1,053, Oct. 1993.

[5] Y. Moses, Y. Adini ve S. Ullman, «Face Recognition: The Problem of Compensating for Changes in Illumination Direction,» %1 içinde *European Conf. Computer Vision*, 1994.

[6] R. Fisher, «The Use of Multiple Measures in Taxonomic Problems,» *Ann. Eugenics,* cilt 7, pp. 179-188, 1936.

[7] R. Duda ve P. Hart, Pattern Classification and Scene Analysis, New York: Wiley , 1973.

Used Resources

“Christopher (view profile),” number of elements of each unique values in a matrix - MATLAB Answers - MATLAB Central. [Online]. Available: https://www.mathworks.com/matlabcentral/answers/158677-number-of-elements-of-each-unique-values-in-a-matrix. [Accessed: 05-Jan-2020].

F. Urrahman and F. U. F. Urrahman, “Replace -inf by min and inf by max value in matlab array,” Stack Overflow, 01-Feb-1967. [Online]. Available: https://stackoverflow.com/questions/41892107/replace-inf-by-min-and-inf-by-max-value-in-matlab-array/41892318. [Accessed: 05-Jan-2020].

Implementation

Code 1: CrossValidation.m

```
function [CorrectlyLabeled,missLabeled,ERRORPERCENT]=CrossValidation(X,y,modelName,distance-
Model,k,threshold,componentCount)
if modelName == "Eigen1"
    numInstance = size(X,2);
    missLabeled=0;
    CorrectlyLabeled=0;
    for i=1:numInstance
        [X_train,y_train,X_test,y_test] = Train_Test_Split(X,y,i);
        [X_train_projected,mu,U,D]=EigenFace_Train(X_train,y_train,componentCount);
        [predictClass,Correctly_Classified] = EigenFace_Test(X_test,y_test,X_train_pro-
jected,y_train,mu,U,D,distanceModel,k,threshold);
        if (~(predictClass ==-1) && Correctly_Classified)
            CorrectlyLabeled=CorrectlyLabeled+1;
        else
            missLabeled=missLabeled+1;
        end
    end

elseif modelName == "FisherFace"
    numInstance = size(X,2);
    missLabeled=0;
    CorrectlyLabeled=0;
    for i=1:numInstance

        [X_train,y_train,X_test,y_test] = Train_Test_Split(X,y,i);
         [X_train_projected,mu,PCA_LDA,D] = Fisherface_train(X_train,y_train);
         mu = zeros(size(mu));
        [predictClass,Correctly_Classified] = Fisherface_test(X_test,y_test,X_train_pro-
jected,y_train,mu,PCA_LDA,D,distanceModel,k,threshold);
        if (~(predictClass ==-1) && Correctly_Classified)
            CorrectlyLabeled=CorrectlyLabeled+1;
        else
            missLabeled=missLabeled+1;
        end
    end
elseif modelName == "EigenWO3"
    numInstance = size(X,2);
    missLabeled=0;
    CorrectlyLabeled=0;
    for i=1:numInstance
        [X_train,y_train,X_test,y_test] = Train_Test_Split(X,y,i);
        [X_train_projected,mu,U,D]=EigenFaceWithout_Train(X_train,y_train,componentCount);
        [predictClass,Correctly_Classified] = EigenFaceWithout_Test(X_test,y_test,X_train_pro-
jected,y_train,mu,U,D,distanceModel,k,threshold );
        if (~(predictClass ==-1) && Correctly_Classified)
            CorrectlyLabeled=CorrectlyLabeled+1;
        else
            missLabeled=missLabeled+1;
        end
    end
end
ERRORPERCENT = 100*missLabeled/(missLabeled+CorrectlyLabeled);
end
```

Code 2: EigenFace_Test.m

```
function [predictClass,Correctly_Classified] = Eigen-
Face_Test(X_test,y_test,X_train_projected,y_train,mu,U,D,distanceModel,k,threshold)
    X_test=WriteInPCABasis(X_test,mu,U);
   [predictClass,Error] = KNN(X_train_projected,X_test,k,distanceModel,D,y_train);
    if Error > threshold
        predictClass=-1;
        Correctly_Classified = false;
        display("Error is more than threshold")
    else
        if predictClass == y_test
            Correctly_Classified = true;
        else
            Correctly_Classified = false;
        end
    end
end
```

Code 3: EigenFace_Train.m

```
function [projection,mu,U,D] = EigenFace_Train(X,y,componentCount)
    pca_results =   my_PCA(X,componentCount);
    D = pca_results.d;
    U = pca_results.U;
    mu = pca_results.mu;
    projection = WriteInPCABasis(X,mu,U);
end
```

Code 4: EigenFaceWithout_Test.m

```
function [predictClass,Correctly_Classified] = EigenFaceWith-
out_Test(X_test,y_test,X_train_projected,y_train,mu,U,D,distanceModel,k,threshold)
    X_test=WriteInPCABasis(X_test,mu,U);
   [predictClass,Error] = KNN(X_train_projected,X_test,k,distanceModel,D,y_train);
    if Error > threshold
        predictClass=-1;
        Correctly_Classified = false;
        display("Error is more than threshold")
    else
        if predictClass == y_test
            Correctly_Classified = true;
        else
            Correctly_Classified = false;
        end
    end
end
```

Code 5: EigenFaceWithout_Train.m

```
function [projection,mu,U,D] = EigenFaceWithout_Train(X,y,componentCount)
```

```
    pca_results =   my_PCA(X,componentCount+3);
    D = pca_results.d;
    U = pca_results.U;
    mu = pca_results.mu;
    D = D(4:end);
    U = U(:,4:end);
    projection = WriteInPCABasis(X,mu,U);
end
```

Code 6: Fisherface_test.m

```
function [predictClass,Correctly_Classified] = Fisher-
face_test(X_test,y_test,X_train_projected,y_train,mu,PCA_LDA,D,distance-
Model,k,threshold)

    X_test=WriteInPCABasis(X_test,mu,PCA_LDA);

    [predictClass,Error] = KNN(X_train_projected,X_test,k,distanceModel,D,y_train);
    if Error > threshold
        predictClass=-1;
        Correctly_Classified = false;
        display("Error is more than threshold")
    else
        if predictClass == y_test
            Correctly_Classified = true;
        else
            Correctly_Classified = false;
        end
    end
end
```

Code 7: Fisherface_train.m

```
function [projection,mu,PCA_LDA,d,U]= Fisherface_train(X,y)

classes= unique(y);
num_classes  = size(classes,2);
num_instance = size(X,2);
pca_base_size = num_instance-num_classes;


pca_results =   my_PCA(X,pca_base_size);
mu= pca_results.mu;
D = pca_results.d ;
U = pca_results.U ;


X_PCA = WriteInPCABasis(X,mu,U);


Sb= zeros(pca_base_size,pca_base_size);
Sw= zeros(pca_base_size,pca_base_size);

general_mean = mean(X_PCA,2);

for i=1:size(classes,2)
    idx= find(y==classes(i));
    Number_class = size(idx,2);
    class_mean = mean(X_PCA(:,idx),2);
    class_mean_minus_general_mean = class_mean-general_mean;
    Sb_component = Number_class * class_mean_minus_general_mean * class_mean_mi-
nus_general_mean';
    centered_class = X_PCA(:,idx) - class_mean;
    Sw_component = centered_class*centered_class';
    Sb = Sb+ Sb_component;
    Sw = Sw+ Sw_component ;
end

[V,D] = eig(Sb,Sw);
[d,ind] = sort(diag(D),1,'descend');
D = D(ind,ind);
V = V(:,ind);


d = d(1:num_classes-1);
V = V(:,1:num_classes-1);
PCA_LDA = U * V;

projection = PCA_LDA'*X;
end
```

Code 8: KNN.m

```
function [predictClass,Error] = KNN(data,guess,k,distanceModel,D,y)

%data at each [p1 p2  .. pN] for each training sample PCA components
%guess single new observation in the same format
```

```
%distanceModel is euclidean or mahalonobis distance
%k is number of closest neigbours
%tie breaking strategy is by default nearest (1-NN)
%task is to find minimum distance points in the training set


%ind index of smallest k distances

%idx index of near neighbours which has the maximum count in unique count
%format

%indexes index of near neighbours which belong to predicted class


if distanceModel == "Mahalonobis"
    distances  = (data-guess).^2;
    %TO avoid numeric errors
    D(D==inf)  = max(D(isfinite(D)));

    distances = distances./D;
    distances=sum(distances,1);

elseif distanceModel == "Euclidean"
    distances  = (data-guess).^2;
    distances=sum(distances,1);
end

[values,ind] = sort(distances);
nearNeigbours = y(ind);

nearNeigbours=nearNeigbours(1:k);
B = unique(nearNeigbours);
out = [B,histc(nearNeigbours,B)];

maxval = max(out(:,2));
idx = find(out(:,2) == maxval);
guesses = out(idx,1);
currentGuess = guesses(1);

if size(guesses,1) > 2
    currentmin   = find(nearNeigbours==guesses(1),1);
    currentGuess =guesses(1);
    for i=2:size(guesses,1)
        if find(nearNeigbours==guesses(i),1) < currentmin
            currentmin =find(nearNeigbours==guesses(i),1);
            currentGuess=guesses(i);
        end
    end
else
    currentGuess = guesses(1);
end

predictClass= currentGuess;
indexes = find(nearNeigbours==predictClass);
Error = mean(values(indexes));

End
```

Code 9: LoadDataset.m

```
function [X,y,datasetName] = LoadDataset(datasetName)
if nargin <1
    datasetName = "CenteredYaleFaces-A";
end
if datasetName == "YaleFaces-A"
    numClasses = 15;
    numInstance = 11;
    InstanceTypes  = ["centerlight","glasses","happy", ...
        "leftlight","noglasses","normal","rightlight", ...
        "sad","sleepy","surprised","wink"];
    X = [];
    y = [];
    for i= 1:numClasses
        for j=1:numInstance
            %creating image name
            currentImageName = sprintf("yalefaces/subject%02.f.%s",i,In-
stanceTypes(j));
            %reading the image
            currentImage = imread(currentImageName);
            %flatenning the image and cover to double.
            currentImage= double(currentImage(:));
            %stackingImages as [ x1 , x2 , x3 ... xN]
            X =[X , currentImage];
            %creating labels for images [c1,c2 ... ] for each N image c can
            %vary as numberClasses
            y =[y;i];
        end
    end
elseif datasetName == "CenteredYaleFaces-A";
    numClasses = 15;
    numInstance = 11;
    InstanceTypes  = ["centerlight","glasses","happy", ...
        "leftlight","noglasses","normal","rightlight", ...
        "sad","sleepy","surprised","wink"];
    X = [];
    y = [];
    for i= 1:numClasses
        for j=1:numInstance
            %creating image name
            currentImageName = sprintf("centered/subject%02.f.%s.pgm",i,In-
stanceTypes(j));
            %reading the image
            currentImage = imread(currentImageName);
            %flatenning the image and cover to double.
            currentImage= double(currentImage(:));
            %stackingImages as [ x1 , x2 , x3 ... xN]
            X =[X , currentImage];
            %creating labels for images [c1,c2 ... ] for each N image c can
            %vary as numberClasses
            y =[y,i];
        end
    end
elseif datasetName == "attfaces";
    numClasses = 40;
    numInstance = 10;
    InstanceTypes  = 1:10;
```

```
    X = [];
    y = [];
    for i= 1:numClasses
        for j=1:numInstance
            %creating image name
            currentImageName = sprintf("attfaces/s%d_%d.pgm",i,InstanceTypes(j));
            %reading the image
            currentImage = imread(currentImageName);
            %flatenning the image and cover to double.
            currentImage= double(currentImage(:));
            %stackingImages as [ x1 , x2 , x3 ... xN]
            X =[X , currentImage];
            %creating labels for images [c1,c2 ... ] for each N image c can
            %vary as numberClasses
            y =[y,i];
        end
    end

end
end
```

Code 10: my_PCA.m

```
function pca_results =   my_PCA(X,componentCount)

mu= mean(X,2);

pca_results.mu = mu;
X = X - mu;

[V,D] = eig(X'*X);

V = X*V;


 for i=1:size(V,2)
     V(:,i) = V(:,i)./norm(V(:,i));
 end



[d,ind] = sort(diag(D),'descend');
D = D(ind,ind);
V = V(:,ind);

d = d(1:componentCount);
V = V(:,1:componentCount);

pca_results.d = d;
pca_results.U = V;
```

Code 11: ShowEigenFaces.m

```
function ShowEigenFaces(U,datasetName)
```

```
figure
for i=1:size(U,2)
    subplot(2,ceil(size(U,2)/2),i)
    if datasetName =="YaleFaces-A"
        current = reshape(U(:,i),243,320);
    elseif datasetName =="CenteredYaleFaces-A"
        current = reshape(U(:,i),231,195);
    end
    imshow(current,[])
    title(sprintf("Eigenface %d",i))

end
sgtitle("EigenFace (Components) Vectors in decending order")
end
```

Code 12: ShowSample.m

```
function ShowSample(Projection,U,datasetName)


y = datasample(Projection,5,2)

images = U*y;
figure
for i=1:size(images,2)
    subplot(2,ceil(size(images,2)/2),i)
    if datasetName =="YaleFaces-A"
        current = reshape(images(:,i),243,320);
    elseif datasetName =="CenteredYaleFaces-A"
        current = reshape(images(:,i),231,195);
    end
    imshow(current,[])
    title(sprintf("Individual %d",i))

end
sgtitle("Random individuals applied PCA after mean subtraction")
end
```

Code 13: Train_Test_Split.m

```
function [X_train,y_train,X_test,y_test] = Train_Test_Split(X,y,testIndex)
X_test=X(:,testIndex);
y_test=y(testIndex);
X(:,testIndex) = [];
y(testIndex) = [];

X_train = X;
y_train = y;


end
```

Code 14: WriteInPCABasis.m

```
function projection = WriteInPCABasis(X,mu,U)
X= X- mu;
projection = U' * X ;
end
```

Code 15: test.m

```
clear all
clc
[X,y,datasetName] = LoadDataset("attfaces");

models=["Eigen1","EigenWO3","FisherFace"];
distanceModels = ["Euclidean","Mahalonobis"];
kvalues =[1,5 ];
componentCounts = [10 , 30];
figure
counter =1;
counters=[];
modelnames=[];
for i_loop = 1:size(models,2)
    model=models(i_loop);
    for j_loop =1:size(distanceModels,2)
        distmodel=distanceModels(j_loop);
        for k_loop = 1:size(kvalues,2)
            k=kvalues(k_loop);
            for m_loop = 1:size(componentCounts,2)
                nComp=componentCounts(m_loop);
                if distmodel == "Euclidean"
                    distmodelshort="Euc."
                else
                    distmodelshort="Mah."
                end
                if model=="Eigen1"
                    modelshort="Eig"
                elseif model==  "EigenWO3"
```

```
                modelshort="Eig-3"
            else
                modelshort="Fisher"
            end
            modelname = sprintf("%s %s k=%d nComp. %d",modelshort,distmod-
elshort,k,nComp)
            [CorrectlyLabeled,missLabeled,ERRORPERCENT]=CrossValida-
tion(X,y,models(i_loop),distanceMod-
els(j_loop),kvalues(k_loop),1000000000000000000000,componentCounts(m_loop))

            bar(counter,ERRORPERCENT)
            hold on
            counters = [counters counter];
            modelnames=[modelnames modelname];
            xticks(counters);
            xticklabels(modelnames);
            xtickangle(70)

            text(counter,ERRORPERCENT,num2str(ERRORPERCENT),'vert','bot-
tom','horiz','center');
            box off
            drawnow;
            counter =counter +1;
            title("Model VS Error")
            xlabel("Models")
            ylabel("Percent Error")
        end
    end
  end
end
```

Code 16: testbymodel.m

```
clear all
clc
[X,y,datasetName] = LoadDataset("attfaces");

models=["Eigen1", "EigenWO3", "FisherFace"];
distanceModels = ["Euclidean","Mahalonobis"];
kvalues =[1,5 ];
componentCounts = [10 , 30];


for i_loop = 1:size(models,2)
    counter =1;
    counters=[];
    modelnames=[];
    figure
    model=models(i_loop);
    for j_loop =1:size(distanceModels,2)
        distmodel=distanceModels(j_loop);
        for k_loop = 1:size(kvalues,2)
            k=kvalues(k_loop);
            for m_loop = 1:size(componentCounts,2)
                nComp=componentCounts(m_loop);
                if distmodel == "Euclidean"
                    distmodelshort="Euc.";
                else
```

```
                    distmodelshort="Mah.";
                end
                if model=="Eigen1"
                    modelshort="Eig";
                elseif model==  "EigenWO3"
                    modelshort="Eig-3";
                else
                    modelshort="Fisher";
                end
                modelname = sprintf("%s %s k=%d nComp. %d",modelshort,distmod-
elshort,k,nComp);
                [CorrectlyLabeled,missLabeled,ERRORPERCENT]=CrossValida-
tion(X,y,models(i_loop),distanceMod-
els(j_loop),kvalues(k_loop),1000000000000000000000,componentCounts(m_loop));
                bar(counter,ERRORPERCENT)
                hold on
                counters = [counters counter];
                modelnames=[modelnames modelname];
                xticks(counters);
                xticklabels(modelnames);
                xtickangle(70);

                text(counter,ERRORPERCENT,num2str(ERRORPERCENT),'vert','bot-
tom','horiz','center');
                box off

                counter =counter +1;
                title( sprintf("%s model with different parameters",model))
                xlabel("Models")
                ylabel("Percent Error")
                drawnow;
            end
        end
    end
end
```

Code 17: tutorialApp.mlapp

```
classdef tutorialApp < matlab.apps.AppBase

    % Properties that correspond to app components
    properties (Access = public)
        UIFigure                   matlab.ui.Figure
        ModelListBoxLabel          matlab.ui.control.Label
        ModelListBox               matlab.ui.control.ListBox
        TESTButton                 matlab.ui.control.Button
        KListBoxLabel              matlab.ui.control.Label
        KListBox                   matlab.ui.control.ListBox
        DatasetListBoxLabel        matlab.ui.control.Label
        DatasetListBox             matlab.ui.control.ListBox
        DistanceMeasureLabel       matlab.ui.control.Label
        DistanceMeasureListBox     matlab.ui.control.ListBox
```

```
        ComponentCountListBoxLabel  matlab.ui.control.Label
        ComponentCountListBox       matlab.ui.control.ListBox
        ThresholdSpinnerLabel       matlab.ui.control.Label
        ThresholdSpinner            matlab.ui.control.Spinner
        UIAxes                      matlab.ui.control.UIAxes
        TextArea                    matlab.ui.control.TextArea
    end
    properties (Access = public)
        counter = 1;
        counters=[];
        modelnames=[];


    end
    % Callbacks that handle component events
    methods (Access = private)


        % Button pushed function: TESTButton
        function TESTButtonPushed(app, event)
            dataset = app.DatasetListBox.Value;
            if dataset == "Yale"
                dataset = "CenteredYaleFaces-A";
            elseif dataset == "AT&T Faces"
                dataset = "attfaces";
            end

            modelName = app.ModelListBox.Value;

            if modelName == "EigenFace"
                modelName = "Eigen1";
            elseif modelName == "EigenFaceW03"
                modelName = "EigenW03";
            elseif modelName == "FisherFace"
                modelName = "FisherFace";
            end

            k = str2double(app.KListBox.Value);

            distanceModel = app.DistanceMeasureListBox.Value;
            if distanceModel == "Mahalanobis"
                distanceModel = "Mahalonobis";
            end
            componentCount = str2double(app.Compo-
nentCountListBox.Value);
            threshold = app.ThresholdSpinner.Value;


             [X,y,datasetName] = LoadDataset(dataset);
```

```
            [CorrectlyLabeled,missLabeled,ERRORPERCENT]=CrossValida-
tion(X,y,modelName,distanceModel,k,threshold,componentCount)
%
%
             if distanceModel == "Euclidean"
                    distmodelshort="Euc.";
                else
                    distmodelshort="Mah.";
                end
                if modelName=="Eigen1"
                    modelshort="Eig";
                elseif modelName==  "EigenW03"
                    modelshort="Eig-3";
                else
                    modelshort="Fisher";
                end
                modelname = sprintf("%s %s k=%d nComp. %d",mod-
elshort,distmodelshort,k,componentCount);
                title(app.UIAxes,"Model VS Error")
                xlabel(app.UIAxes,"Models")
                ylabel(app.UIAxes,"Percent Error")
                ylim(app.UIAxes,[0 45])
                %ERRORPERCENT=randi(30);
                bar(app.UIAxes,app.counter,ERRORPERCENT);
                hold(app.UIAxes,"on");
                app.counters = [app.counters app.counter];
                app.modelnames=[app.modelnames modelname];
                xticks(app.UIAxes,app.counters);
                xticklabels(app.UIAxes,app.modelnames);
                xtickangle(app.UIAxes,70);

                text(app.UIAxes,app.counter,ERRORPERCENT,num2str(ER-
RORPERCENT),'vert','bottom','horiz','center');
                box(app.UIAxes,'off');


                app.counter =app.counter +1
        end
    end


    % Component initialization
    methods (Access = private)


        % Create UIFigure and components
        function createComponents(app)


            % Create UIFigure and hide until all components are created
```

```
            app.UIFigure = uifigure('Visible', 'off');
            app.UIFigure.Position = [100 100 640 480];
            app.UIFigure.Name = 'UI Figure';

            % Create ModelListBoxLabel
            app.ModelListBoxLabel = uilabel(app.UIFigure);
            app.ModelListBoxLabel.HorizontalAlignment = 'right';
            app.ModelListBoxLabel.Position = [194 441 39 22];
            app.ModelListBoxLabel.Text = 'Model';

            % Create ModelListBox
            app.ModelListBox = uilistbox(app.UIFigure);
            app.ModelListBox.Items = {'EigenFace', 'EigenFaceW03',
'FisherFace'};
            app.ModelListBox.Position = [248 408 156 57];
            app.ModelListBox.Value = 'EigenFace';

            % Create TESTButton
            app.TESTButton = uibutton(app.UIFigure, 'push');
            app.TESTButton.ButtonPushedFcn = createCallbackFcn(app,
@TESTButtonPushed, true);
            app.TESTButton.Position = [527 424 100 22];
            app.TESTButton.Text = 'TEST';

            % Create KListBoxLabel
            app.KListBoxLabel = uilabel(app.UIFigure);
            app.KListBoxLabel.HorizontalAlignment = 'right';
            app.KListBoxLabel.Position = [418 439 25 22];
            app.KListBoxLabel.Text = 'K';

            % Create KListBox
            app.KListBox = uilistbox(app.UIFigure);
            app.KListBox.Items = {'1', '3', '5'};
            app.KListBox.Position = [458 408 41 55];
            app.KListBox.Value = '1';

            % Create DatasetListBoxLabel
            app.DatasetListBoxLabel = uilabel(app.UIFigure);
            app.DatasetListBoxLabel.HorizontalAlignment = 'right';
            app.DatasetListBoxLabel.Position = [7 443 47 22];
            app.DatasetListBoxLabel.Text = 'Dataset';

            % Create DatasetListBox
```

```
            app.DatasetListBox = uilistbox(app.UIFigure);
            app.DatasetListBox.Items = {'Yale', 'AT&T Faces'};
            app.DatasetListBox.Position = [69 425 106 42];
            app.DatasetListBox.Value = 'Yale';

            % Create DistanceMeasureLabel
            app.DistanceMeasureLabel = uilabel(app.UIFigure);
            app.DistanceMeasureLabel.Position = [7 349 90 47];
            app.DistanceMeasureLabel.Text = {'Distance'; 'Measure'};

            % Create DistanceMeasureListBox
            app.DistanceMeasureListBox = uilistbox(app.UIFigure);
            app.DistanceMeasureListBox.Items = {'Euclidean', 'Mahalano-
bis'};
            app.DistanceMeasureListBox.Position = [69 347 106 51];
            app.DistanceMeasureListBox.Value = 'Euclidean';

            % Create ComponentCountListBoxLabel
            app.ComponentCountListBoxLabel = uilabel(app.UIFigure);
            app.ComponentCountListBoxLabel.Position = [194 349 90 47];
            app.ComponentCountListBoxLabel.Text = {'Component';
'Count'};

            % Create ComponentCountListBox
            app.ComponentCountListBox = uilistbox(app.UIFigure);
            app.ComponentCountListBox.Items = {'10', '30'};
            app.ComponentCountListBox.Position = [268 349 43 47];
            app.ComponentCountListBox.Value = '10';

            % Create ThresholdSpinnerLabel
            app.ThresholdSpinnerLabel = uilabel(app.UIFigure);
            app.ThresholdSpinnerLabel.HorizontalAlignment = 'right';
            app.ThresholdSpinnerLabel.Position = [341 361 59 22];
            app.ThresholdSpinnerLabel.Text = 'Threshold';

            % Create ThresholdSpinner
            app.ThresholdSpinner = uispinner(app.UIFigure);
            app.ThresholdSpinner.Position = [415 361 70 22];
            app.ThresholdSpinner.Value = 1e+21;

            % Create UIAxes
            app.UIAxes = uiaxes(app.UIFigure);
            title(app.UIAxes, 'Title')
```

```
            xlabel(app.UIAxes, 'X')
            ylabel(app.UIAxes, 'Y')
            app.UIAxes.Position = [25 22 562 300];

            % Create TextArea
            app.TextArea = uitextarea(app.UIFigure);
            app.TextArea.Editable = 'off';
            app.TextArea.Position = [498 330 129 66];
            app.TextArea.Value = {'Each Test requires approximately 50
seconds.'};
            % Show the figure after all components are created
            app.UIFigure.Visible = 'on';
        end
    end
    % App creation and deletion
    methods (Access = public)

        % Construct app
        function app = tutorialApp

            % Create UIFigure and components
            createComponents(app)

            % Register the app with App Designer
            registerApp(app, app.UIFigure)

            if nargout == 0
                clear app
            end
        end
        % Code that executes before app deletion
        function delete(app)

            % Delete UIFigure when app is deleted
            delete(app.UIFigure)
        end
    end
end
```