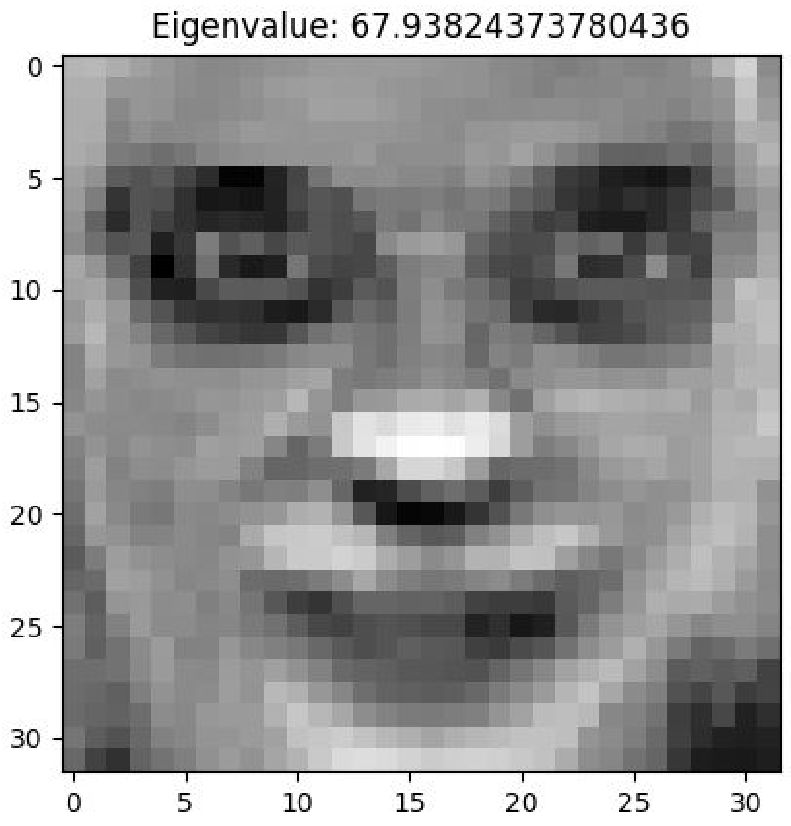
FACE RECOGNITION

A report on the application of eigenfaces in facial recognition.



Richard Polzin

29.05.2017

Maastricht University - Computer Vision - Stelios Asteriadis

INTRODUCTION

The following report will contain information about the experiments conducted on the 32x32 ORL Database of Faces [1]. To reduce the size of the report only a short introduction to eigenfaces is given - more information is available on the web [2]. This introduction takes place in the first chapter Eigenfaces. The second chapter Implementation quickly details the structure of the algorithm. In the third chapter Experiments the experiments conducted are discussed and the results of the algorithm are shown. Finally Conclusion & Future Work summarizes the results and offers ideas for future work.

EIGENFACES

The eigenface technique is used to reduce the dimensionality of facial images. It takes advantage of the concept that faces align roughly along a subspace for which an representation can be found via principal component analysis. The derived eigenvectors can be used to represent faces with reasonable loss of information. These vectors are called eigenfaces and often reduce the dimensionality of the data significantly, allowing eigenface-based algorithm to be fast and efficient.

IMPLEMENTATION

Roughly four steps can be separated in the implementation of the face recognition. In the first step the training images are used to generate the eigenfaces. This steps mostly takes place in the *pca* method. Those eigenfaces are then used to project the training data to the new space and the results can be compared when passing the --*comptrain* flag. See Image 2 for reference. In the third step the dest data is projected to the new space. To compare the original to the projection pass --*comptest*. Finally the Nearest Neighbour implementation of *scikit-learn*[3] is used to find nearest neighbours and generate an accuracy score. Besides the *pca* all steps are executed on module level for easier readability while still providing good separation of concerns.

[1] The full-size version is available here: http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html
[2] E.g. http://www.cs.ucsb.edu/~mturk/Papers/mturk-CVPR91.pdf
[3] A Python machine learning library available at http://scikit-learn.org/ .

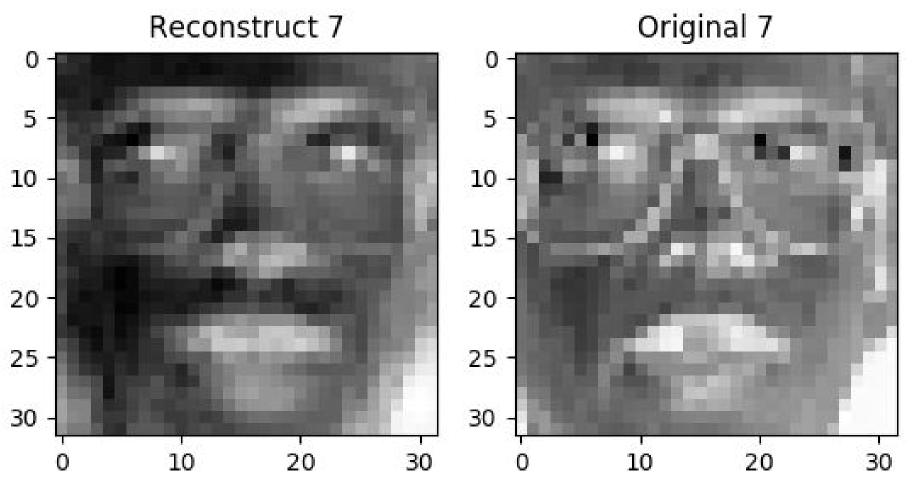


Image 2: Comparison of reconstructed and original test image.

EXPERIMENTS

Experiments were conducted on 400 32x32 images from 40 people. Different facial expressions, details and lighting were instituted to generate a diverse data set. For the experiments either three, five or seven images per person were used to generate the eigenfaces. These will be referred to as training data. For the calculation of the accuracy the other images were used. The accuracy is calculated as the fraction of correctly labeled faces. Furthermore the amount of principal components taken into consideration was varied from k=10 to k=100, step 5. The accuracies achieved can be seen in Image 3.

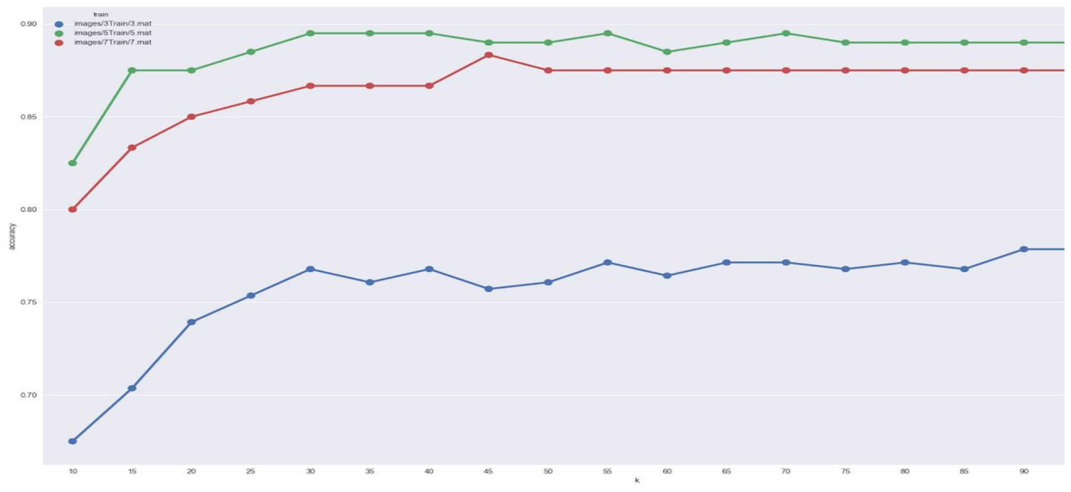


Image 3: Plot showing the accuracy over different k values and training sizes. For an easier to read visualization check the Appendix.

An interesting result is that the accuracy for seven training images is worse than the accuracy for five training images. This is probably an issue of overfitting, where the eigenfaces generated became too tailored to the problem, such that they don't offer a good generalization anymore. Furthermore the difference in accuracy between five and three training images should be noted. Even though overfitting is a possible issue having too little training data is more of an issue. The accuracy is improved greatly by adding more images to the training data. In general a relatively small number of principal components, roughly 30, seem to be sufficient for a accuracy that is almost converged. Considering the original data had 1,024 dimensions (32x32 pixels) this reduction is tremendous.

CONCLUSION & FUTURE WORK

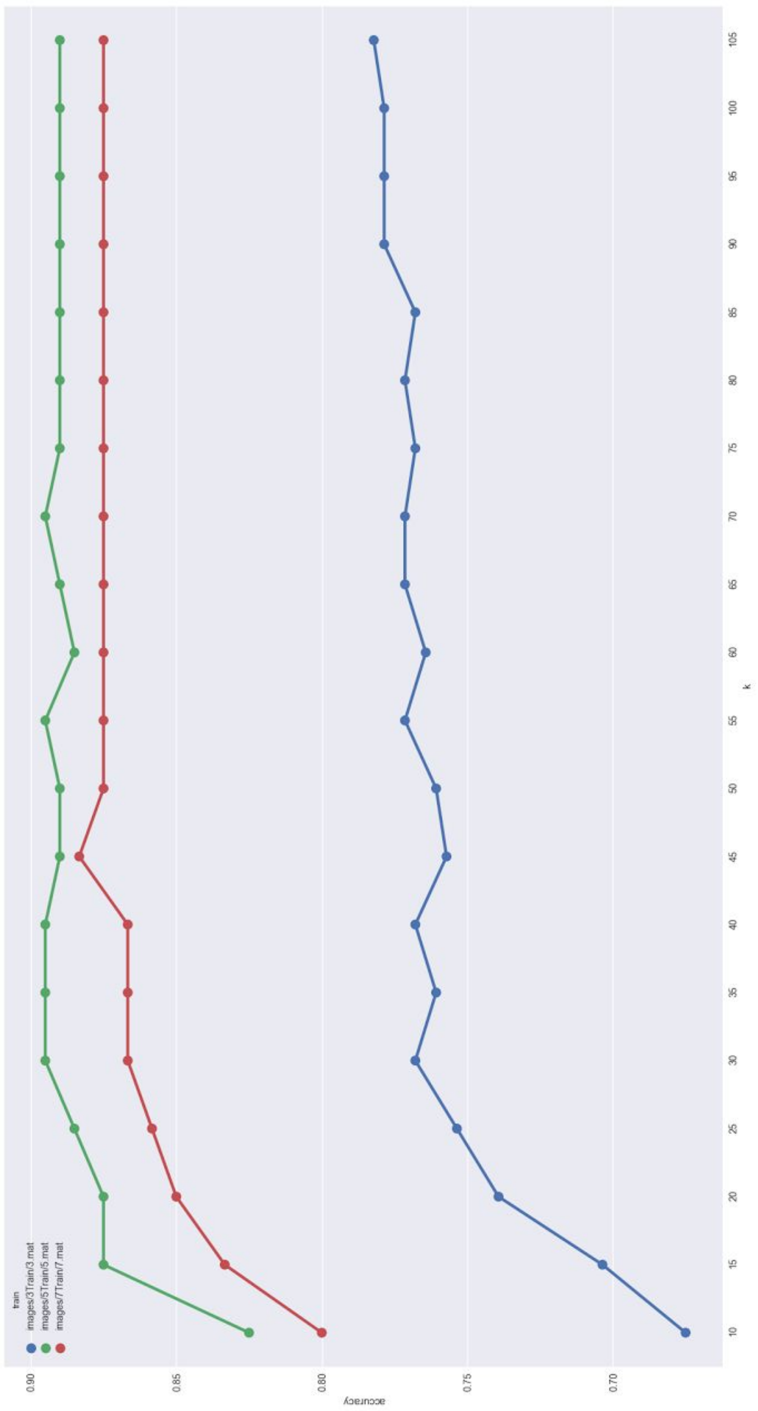
Experiments on the usage of eigenfaces showed different aspects of this technique that are important to consider when working with this technique. First of all both of the algorithms parameters, k and the number of training images, greatly influence the accuracy. For k choosing a good value is rather simple as any large [4] k offers good results, fine-tuning this parameter can reduce the dimensionality even further and help speeding up the algorithm while reducing its memory footprint. Furthermore this parameter might be dependent on the image resolution.

Choosing a good value for the amount of training data is a more complex task. Based on the small scope of this project it seems to be important to have sufficient training data, yet not too much as not to overfit the model.

Researching this in more detail is up to future work. Another topic for future work is the analysis of changes due to larger or smaller dimensionality. How many principal components are required for the accuracy to converge for training images of different resolutions ?

[4] In this case >100 could be considered large. In general large is considered a size where the accuracy certainly already converged.

APPENDIX


train
images/3Train/3.mat
images/5Train/5.mat
images/7Train/7.mat
accuracy
0.90
0.85
0.80
0.75
0.70
k
10
15
20
25
30
35
40
45
50
55
60
65
70
75
80
85
90
95
100
105