# Eigenfaces

A face recognition project for data structures and algorithms (documentation written in finnish)

# The problem to be solved

This project intents to use Principal Component Analysis and general ideas relating to linear algebra, eigenvectors and such to create a face recognition system. A large set of facial images will be translated into vectors via PCA. This vector set is then converted into a set of eigenvectors (or more familially, eigenfaces) which can be used to create any given face within the dataset. Used datastructures are vectors and matrices, with the primary algorithm being a self-implemented PCA algorithm.

The reason for the selection of these specific algorithms is derived from the fact that linear algebra is COOL, and using it to solve a face recognition problem seems like an interesting idea.

# Programming languages

This project utilises python. I also know some C#, haskell and some lisp.

# Language

All related documents are in english.

# Specific data structures and algorithms

### algorithms

_The following is a paraphrase of the wikipedia-article about PCA: https://www.wikiwand.com/en/Principal_component_analysis, specifically the section "Computing PCA using covariance method". The idea is to present a rough sketch of the algorithm, and it is very likely that this will not reflect the state of the final project.._

Principal Component Analysis (PCA) (covariance method):

1. Sorting images into matrices; time complexity is O(_p_*_n_), as the size of each vector is _p_ and they will be processed only once:

a. Turn each image into a row vector _p_ (_p_ is based on the resolution of the image).

b. Use vectors _p_ to create matrix _m_ so that Dim(_m_) = _n_ x _p_, each vector being length _p_, with _n_ being the amount of images in the dataset

2. Calculating empirical mean; time complexity is O(_p_*_n_) as the size of each vector is _p_, and there's _n_ of them:

a) Find empirical mean from columns 1,...,_p_

b) Add mean to column vector _u_

3. Create matrix _B_ that contains all row vectors - _u_^T (transpose); Time complexity is O(_p_*_n_) as the length of each vector is _p_ and there's _n_ of them.

a) For each row of _m_ calculate row - _u^T_ (transpose)

b) Create new matrix _B_ intailing all of these row vectors.

4. ![Find covariance-matrix C with this formula.](https://github.com/MiikaMatias/Eigenface/tree/main/assets/images/CovarianceMatrix_formula.png); From this point onward my knowledge of the required algorithms is not enough to attempt a time-complexity analysis.

5. Find the eigenvalues and eigenvectors of _C_

6. Sort the eigenvectors and eigenvalues of _C_

7. Calculate cumulative energy content

8. Choose a basis from _C_.

9. Project the image into the basis of _C_.

### data structures
- Vectors, matrices and other such datastructures either found in python libraries or made myself.

# inputs and outputs
The program takes an input of a face and outputs it as a combination of eigenvectors from the model.

# Study program
I am a computer science (TkT) student from Helsinki University TkT study track

# Sources
All information par from PCA information is from this article:
https://www.wikiwand.com/en/Eigenface

The PCA paraphrase is form this article:
https://www.wikiwand.com/en/Principal_component_analysis