ELEC6910R Robotic Perceptions and Learning
Chung Hee KIM 20080862

Homework 4: Testing Algorithms

This homework presents the demonstration of four algorithms along with some background knowledge:

I. K-Means
II. Meanshift
III. Iterative Closest Point
IV. PCA and Eigenface.

The results of the demonstration is documented here. To see the code that produced the result, refer to my github repository:

I. K-MEANS CLUSTERING

A. Background

k-means is a method of cluster analysis that aims to partition *n* observations into *k* clusters in which each observation belongs to the cluster with the nearest mean.

B. Application

- Separate groups of points into separate clusters
- The number of cluster must be known
- Requires a distance over the points

C. When not to use K-mean?

- K is not known
- There is no distance on the data point
- Data points are not linearly separable
- If clusters correspond to severely non-gaussian distributions

D. Sudo Algorithm

Input:

- k (number of clusters)
- Training data set {$x_{(1)}$, $x_{(2)}$, ... , $x_{(m)}$}

Output:

- group data points in k clusters centered at cluster means

```
Randomly initialize k cluster centroids $\mu_1, \mu_2, \ldots, \mu_k$
Repeat {
   for i = 1 to m
      $c_{(i)}$ = index (from 1 to k) of cluster centroid closest to $x_{(i)}$
   for k = 1 to k
      $\mu_k$ = average of points assigned to cluster k
   }
```

*Note: the first loop is the cluster assignment step and the second loop is the move centroid step.

Notation

$c_{(i)}$ = index of cluster (1, 2, ... k) to which example $x_{(i)}$ is currently assigned
μ_k = cluster centroid k
$\mu_{c(i)}$ = cluster centroid of cluster to which exmaple $x_{(i)}$ has been assigned

E. K-means Optimization Objective

$$J(c^{(1)},\ldots,c^{(m)},\mu_1,\ldots,\mu_k)=\frac{1}{m}\sum_{i=1}^{m}\left\|x^{(i)}-\mu_{c(i)}\right\|^2$$

$$\min_{c^{(1)},\ldots,c^{(m)},\mu_1,\ldots,\mu_k} J(c^{(1)},\ldots,c^{(m)},\mu_1,\ldots,\mu_k)$$

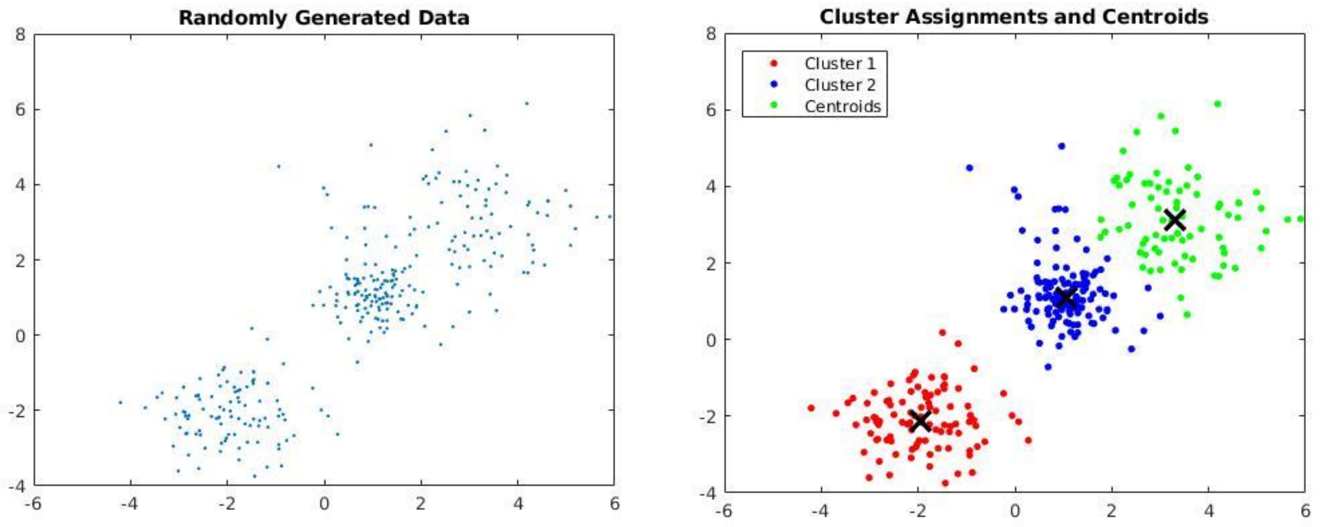
F. Algorithm Demonstration

Dataset: Random generation of 2D points
Tutorial: https://www.mathworks.com/help/stats/kmeans.html
Language: MATLAB

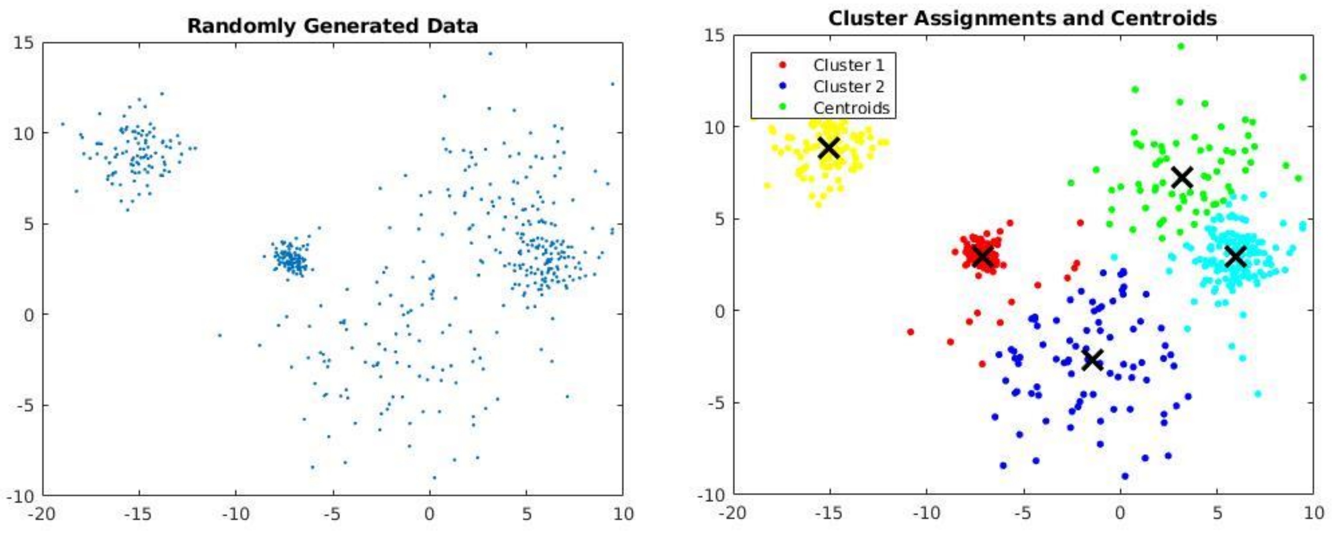
Source Code: Refer to github

Results:

K-means clustering with 3 clusters



K-means clustering with 5 clusters



II. MEANSHIFT

A. Background

Meanshift is a non-parametric feature-space analysis technique for locating the maxima of a density function, a so-called mode-seeking algorithm.
Objective: Find the mode of a general distribution without computing the density
Core Idea: Follow the density gradient

B. Application

- Finding the mode of a distribution from samples, with a single node
- For multi-modal distribution, find local maxima
- Particle Filter
- Clustering

C. Sudo Algorithm

Assuming *n* input points: $\{x_i\}$

Assuming a window W centered on X, and G the Gaussian function: $G(x) = \exp(-\frac{x^2}{2})$

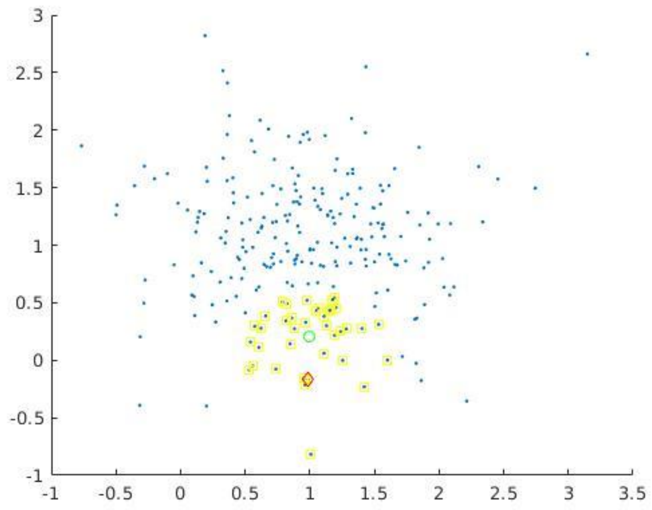
```
Do:
    Compute the mean-shift vector
    Move the window center according to the mean-shift vector
    Until convergence: $||X_{k+1} - X_k|| \leq \varepsilon$
```

D. Algorithm Demonstration

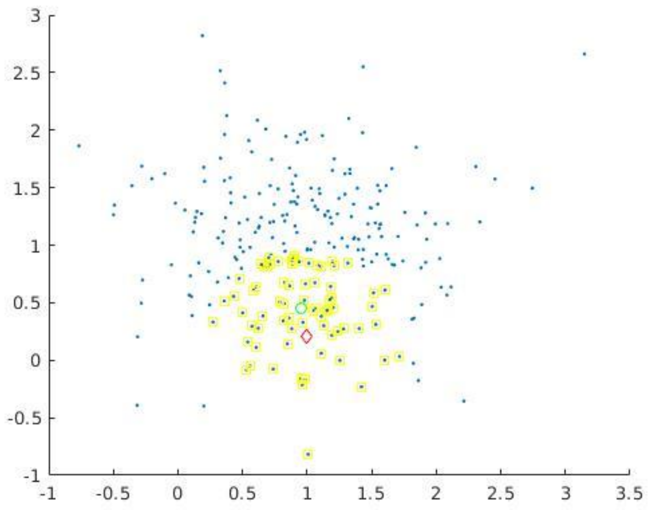
Dataset: Random generation of 2D points (250 samples)
Tutorial: https://www.mathworks.com/matlabcentral/fileexchange/10161-mean-shift-clustering
Language: MATLAB
Source Code: Refer to github
Results:

- **Red Marker:** initial center
- **Green Marker:** New window center after mean-shfting
- **Yellow Marker:** Data within window

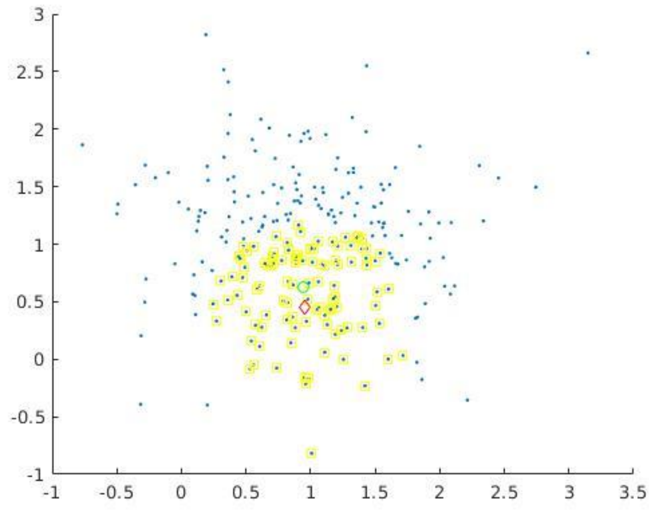
Iteration 1



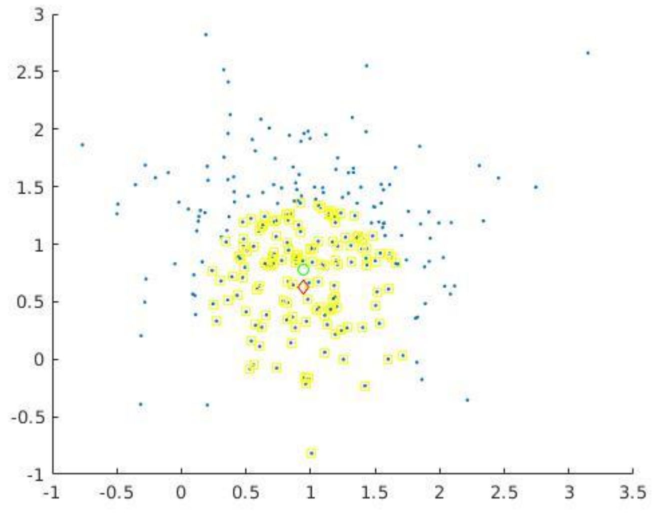
Iteration 2



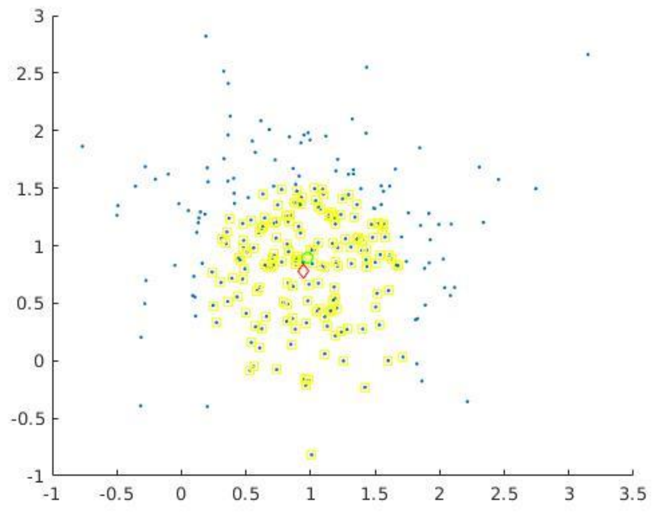
Iteration 3



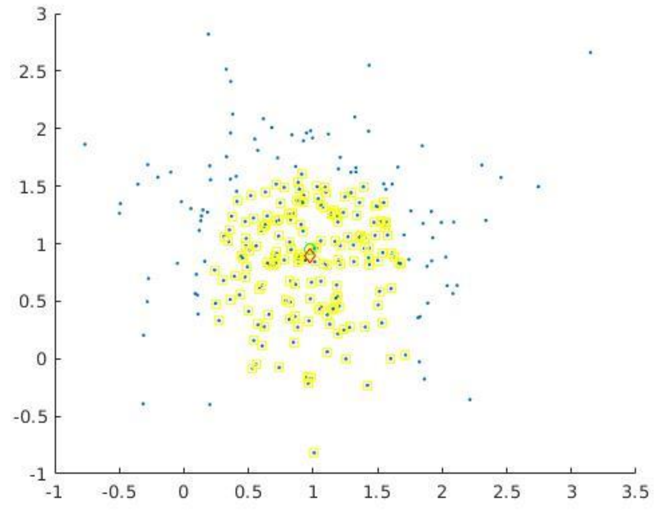
Iteration 4



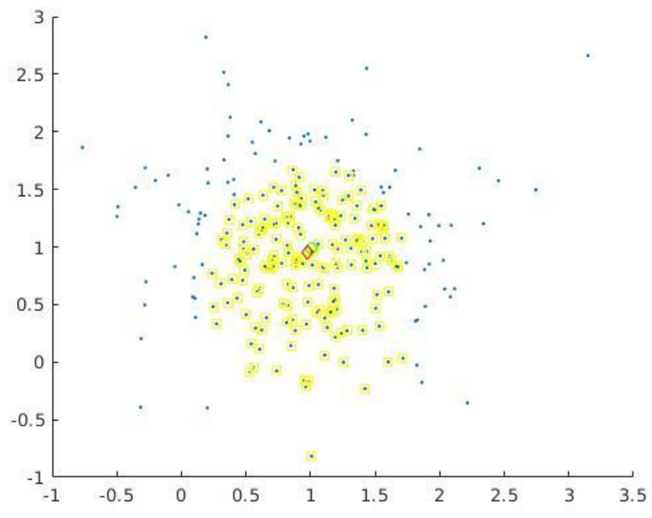
Iteration 5



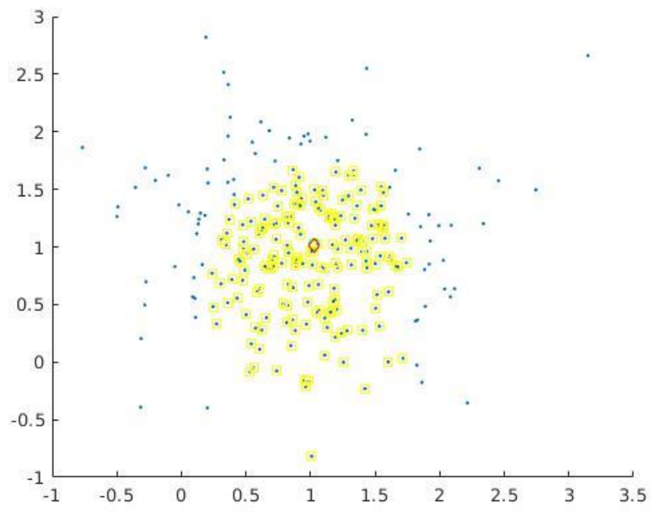
Iteration 6



Iteration 7



Iteration 8



III. Iterative Closest Point (ICP)

A. Background

Iterative closest point (ICP) is an algorithm employed to minimize the difference between two clouds of points. ICP is often used to reconstruct 2D or 3D surfaces from different scans such as laser scan point clouds.

B. Application

- find object into a scene
- estimate motion of sensor
- compare two scenes
- merge observations into a map

C. Sudo Algorithm

Definition: Find transform between 2 set of points
Input:

- Reference point cloud
- Data point cloud

Output:

- Transformation between reference and data (3DOF in 2D, 6DOF in 3D)

1. Preprocessing: clean data
2. Matching: Associate poinits from reference to data via neighbor search
3. Weighting: change importance of some matching pairs
4. Rejection: discard some pairs
5. Compute error for each pair
6. Minimization: Find best transform
7. Loop to step to unless converges

D. Methods

Matching: 2 methods
1. Linear nearest neighbor search
- Compute distance for every point (exhaustive)
- Keep current maximum
- Good for high dimesions and low number of points

2. k-d tree nearest neighbor search
- use segmentation of points in a tree structure
- good for low dimension and high number of points

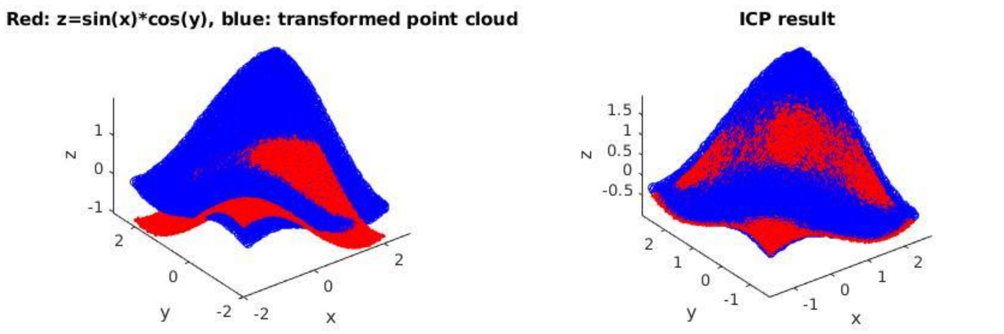
Error Minimization: 2 methods
1. Point to point: standard with closed-form solution
2. Point to plane: better but with approximate techniques

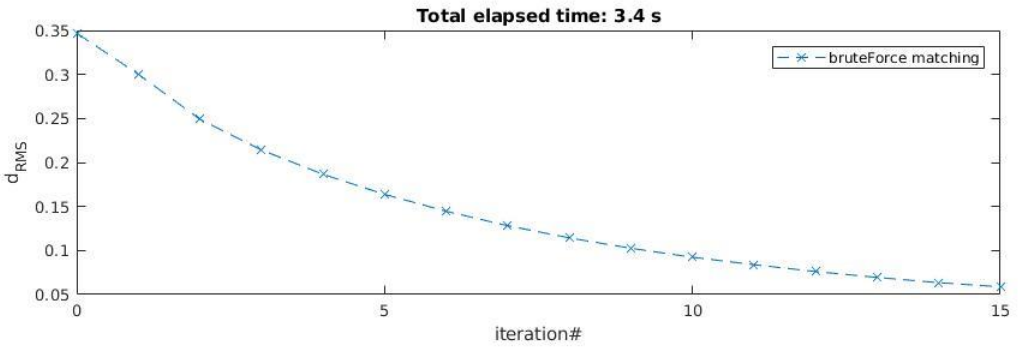
E. Algorithm Demonstration

Dataset: Generation of 3D points with a 3D equation in a meshgrid
Tutorial: https://www.mathworks.com/matlabcentral/fileexchange/27804-iterative-closest-point
Language: MATLAB
Source Code: Refer to github

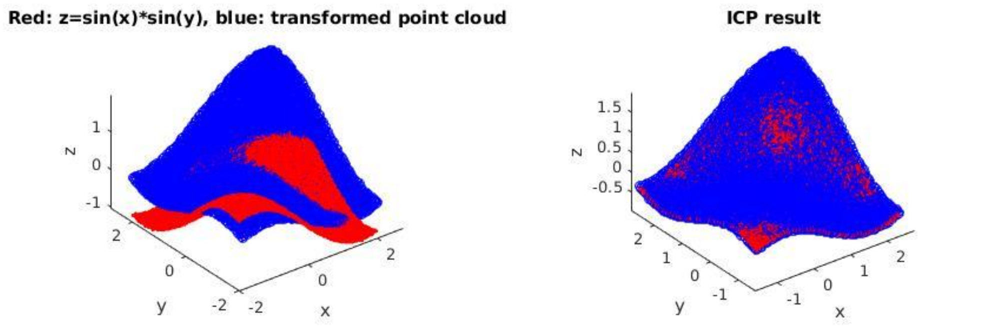
Results:

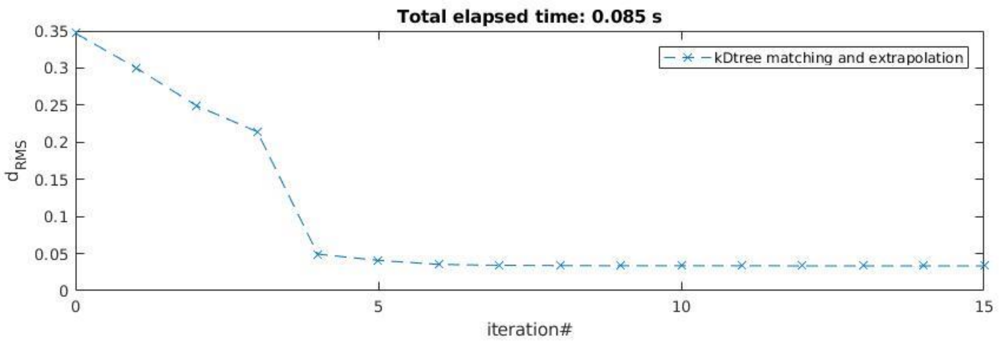
ICP using bruteforce linear nearest neighbour search





ICP using kDtree nearest neighbour search (much faster)





IV. Principal Component Analysis (PCA) & Eigenfaces

A. Background

Principal Component Analysis is a tool to identify main characteristics of a data set. This is done by identifying the direction in a feature space along which points have the greatest variance. The direction that captures the maximum covariance of the data is the eigenvector corresponding to the largest eigenvalue of the data covariance matrix. Furthermore, the top k orthogonal directions that capture the most variance of the data are the k eigenvalues corresponding to the largest eigenvalue.

First Principle Component: direction of maximum variance

Subsequent Principle Components: orthogonal to previous PC's and describe maximum residual variance.

Eigenfaces is the name given to a set of eigenvectors when they are used in the computer vision problem of human face recognition. The eigenfaces form a basis set of all images used to construct the covariance matrix. This produces dimension reduction by allowing the smaller set of basis images to represent the original training images.

B. Application

- Dimension reduction
- Face recognition

C. Sudo Algorithm

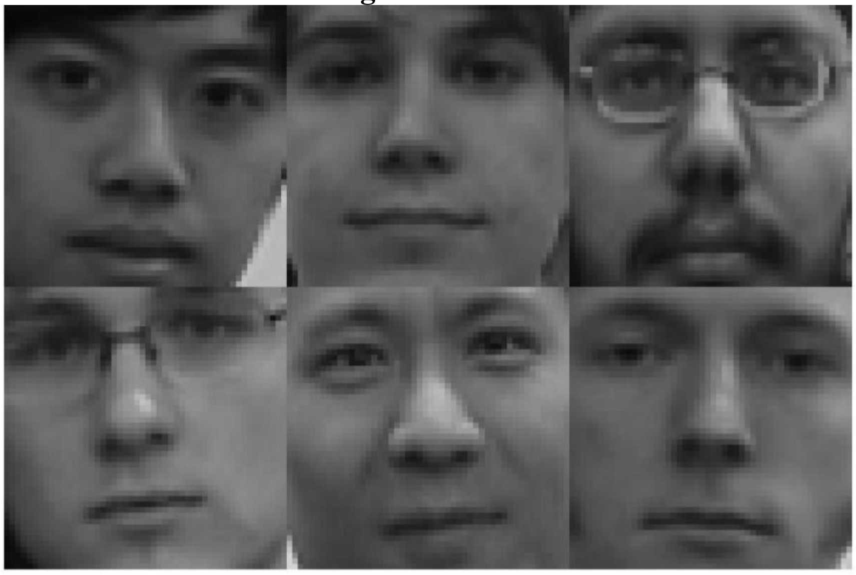
1. Convert n x m matrix of pixels into nm vector by stacking columns
2. M = compute average vector
3. Subtract M from each vector such that zero centered distribution is obtained
4. C = compute covariance matrix
5. Change all points into eigenvector frame
6. Select "Just enough" dimensions according to the strength of eigenvalues
7. Discard remaining dimensions

D. Algorithm Demonstration

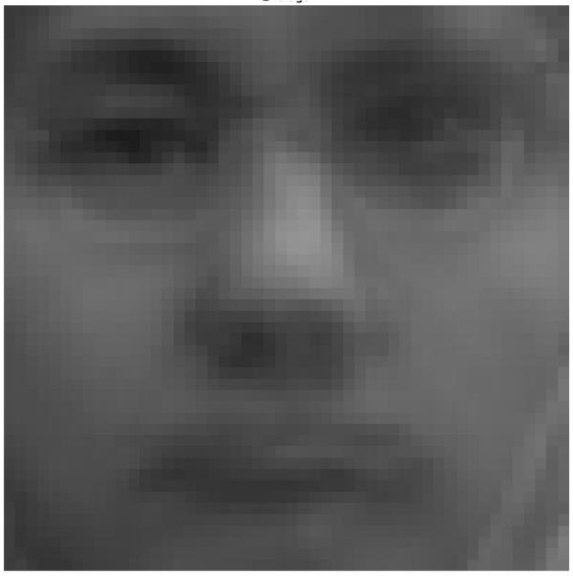
Dataset: 6 images for training, 3 images for recognition testing
Tutorial: https://www.mathworks.com/matlabcentral/fileexchange/45915-eigenfaces-algorithm
Language: MATLAB
Source Code: Refer to github

Results

Training set of 6 faces



Average face



Eigenfaces