# **Face Recognition using k-Nearest Neighbor**

Face recognition is the process of identifying a person in a digital image or video.

Example:

Applications of Face Recognition include:

* Video Surveillance
* Biometric Passwords
* Social Networks
* Object Recognition
  + Robotics
  + Self-driving cars
  + Online Shopping apps

The general steps to perform face recognition are:

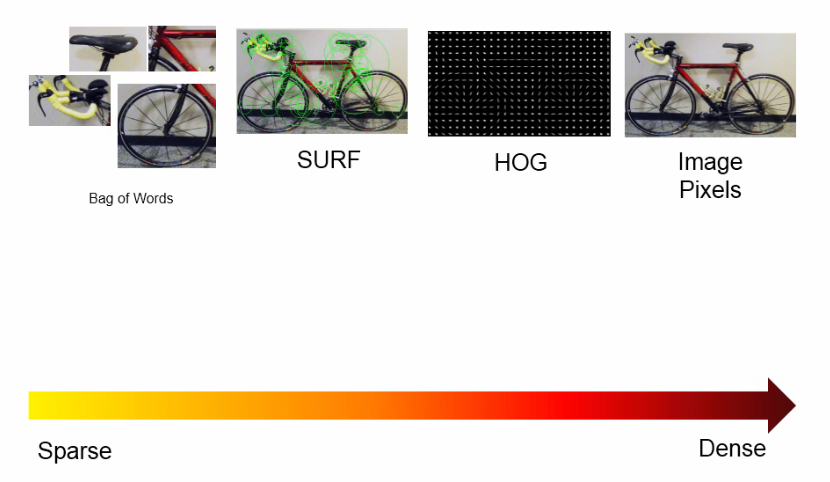
1. Face Database/Gallery:

The first step of the face recognition system is creating a database of facial images of people you are interested to recognize.

1. Feature Extraction:

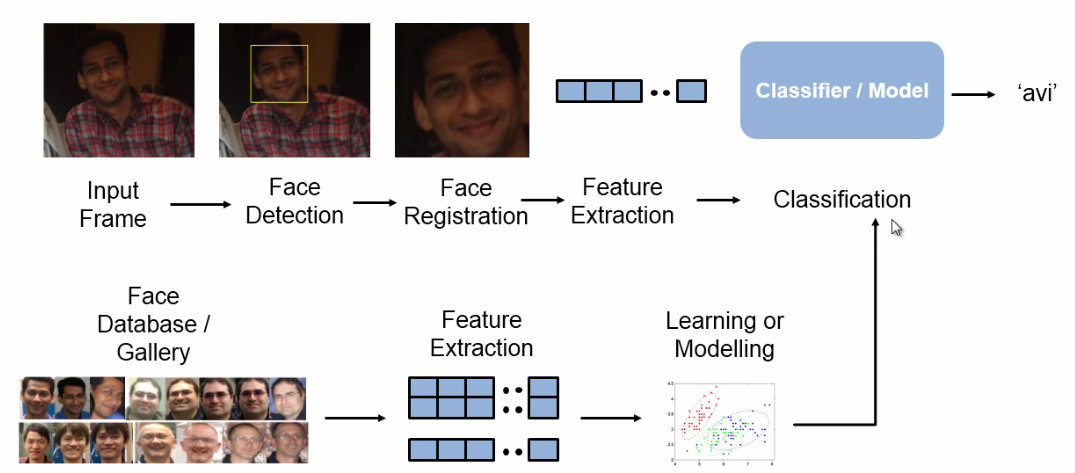
Extract critical features of every image and store them as feature vectors.

Examples:



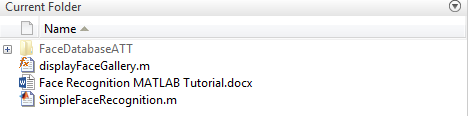
1. Learning or modeling of faces

Implementing machine learning algorithm to differentiate different face images



Face Recognition using k-Nearest Neighbor in MATLAB

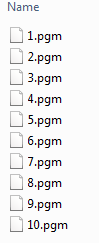
Please open the Matlab file “SimpleFaceRecognition.m” by double clicking. The Matlab should prompt out. Make sure your “MATLAB” current folder looks like this:



“displayFaceGallery” is just a function file. Please do not open it (but you need to keep it in the same directory with “SimpleFaceRecognition.m”).

1. **Import and read images**

We will use ATT Face Database. Each folder (S1, S2 … S40) in the database represents one person, and within each folder, you can see 10 different images for that person:



These images are saved in “pgm” format and they can be easily read in MATLAB.

* Please import the MATLAB files from the Matlab facial recognition folder and highlight the following section:



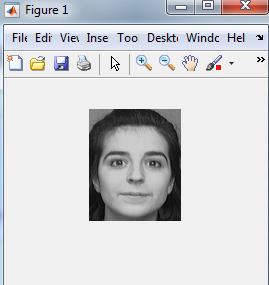
* Then click  by only run this small section. You will see  stored in your workspace. “imageSet” not only brings the images into MATLAB but also maintains the hierarchical relation between images.

Let us try to read one image from the database.

* Please type the following code in the “Command Window”:

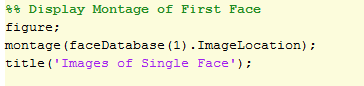


You are opening the second image in the second folder in the base:

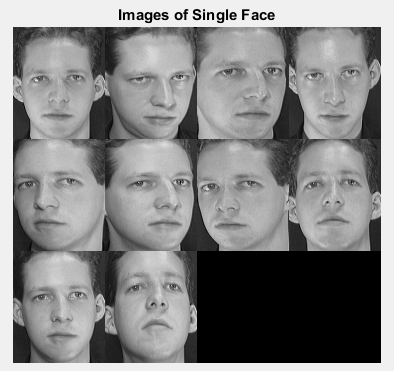


We can also open a group of images using the Matlab function montage.

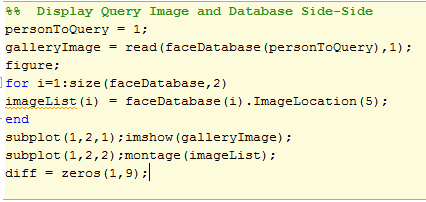
* Please highlight the following section:



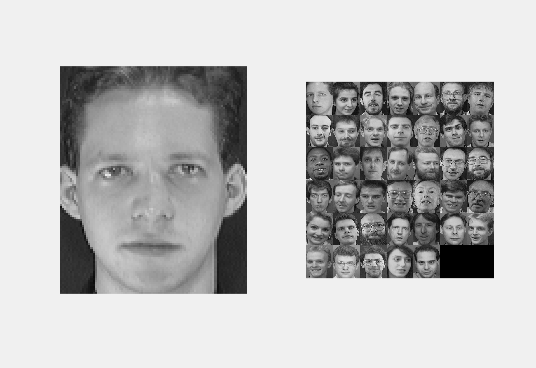
* Then click , you can see the images for the first person:



Our task is that, giving a query image, the retrieval system will find a similar image in the database. Please run the section of the code below:



You can see:



1. **Split the database into training and testing**

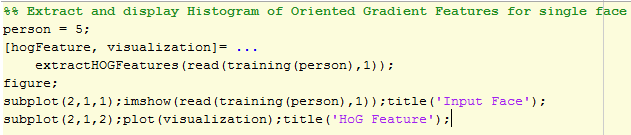
Please run the following section of the code:



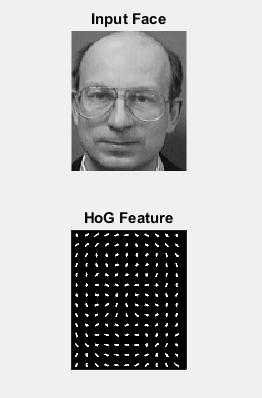
1. **Extract Features**

We are going to extract HOG features.

* Please run the following section of the code:



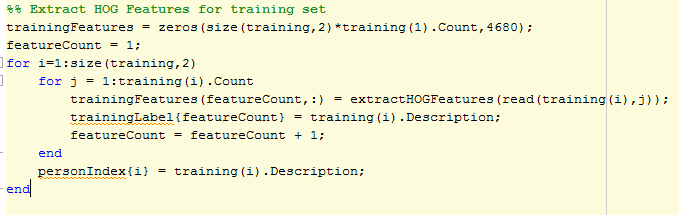
You can see:



HoG feature includes the edge information (direction).

Next, we are going to extract HoG features for all the training images.

* Please run the following section of the code



In the workspace, you can see: 

“320” represents the number of observations, we have 40 persons and 8 images per person were treated as training images. 4680 represents the number of features HoG algorithm extracted.

1. Apply a classification algorithm on the images represented in terms of feature vectors.

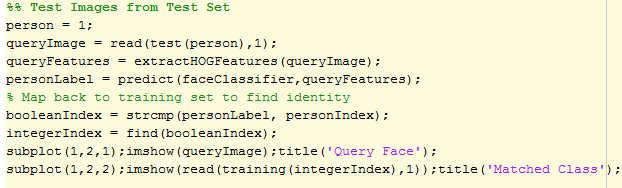
Here, we use the **KNN** as our classifier.

* Please run the following section of the code

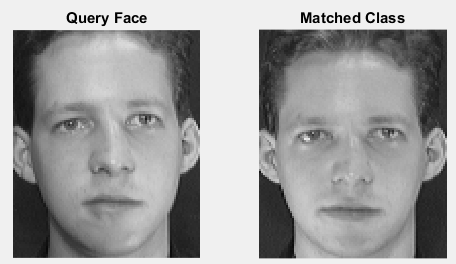


Next, let us check how our classifier works:

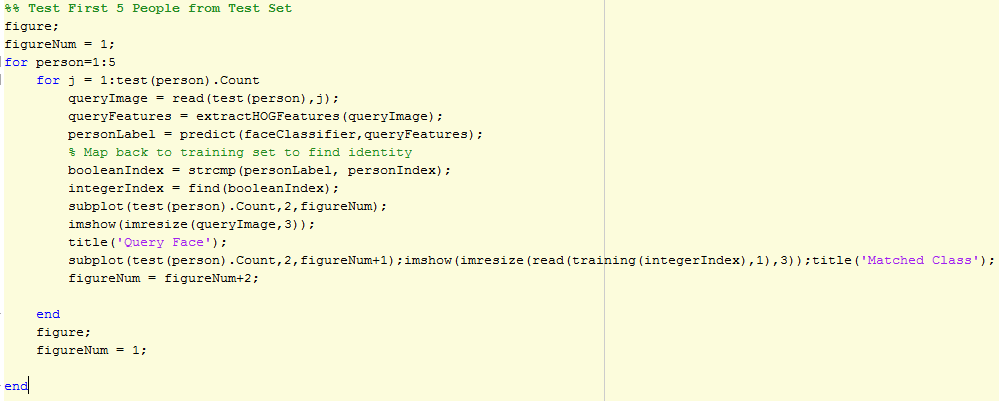
* Please run the following section of the code



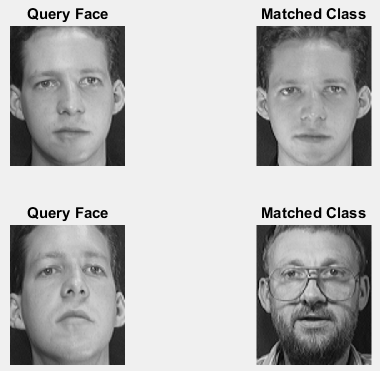
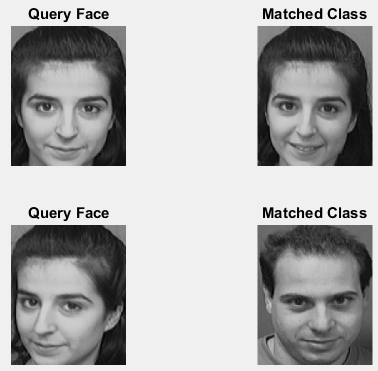
You can see:

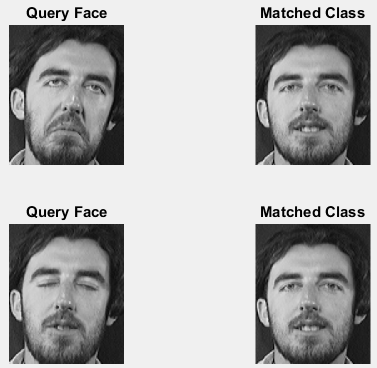
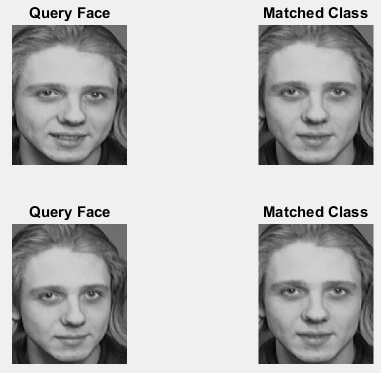


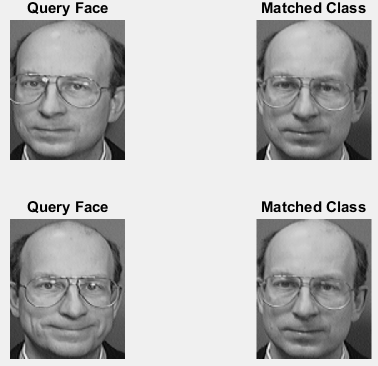
* You can also check the first 5 testing images by running the following section of the code



You will see 5 windows prompt out:



**Exercises:**

1. In the tutorial, you use the function montage to visualize all the images for the first person. Use the same function to visualize all the images for the second person. How are those images different from each other? Why do you think it is important to choose different face instances for the same person?
2. Change the number of K in the KNN classification approach from K=1 to K=3. Do you see any differences in the results? How do you explain these differences?
3. Change the sample ratio between the training and testing data. For example: 70% training and 30% training. Do you see any differences in the results? Can you explain? **(You need to clean your workspace before you make the change, and re-run the code from the beginning)**

How to clean the workspace?

In the command window, type “clear” and then type “enter”:

