# Reconnaissance faciale avec MATLAB

R. Rodriguez Salas
*ESIEE Paris, Univ. Paris-Est*
Paris, France
r.rodriguez@esiee.fr

***Abstract*—On this work based on the Reconnaissance faciale avec MATLAB webinar code is tested and results are obtained on PubFig faces database, the workflow is explored and introduces the use of Histogram Oriented Features for image classification. This work is intended to get acknowledgement of the field, of the HoG Features and the workflow of neural networks.**

***Index Terms*—image, classification, HoGFeatures**

## I. Introduction

Face recognition could be explained as the process of analyze and comparing patterns on an image looking for identifying one or more people in images or videos. Giving a gallery or data set images of people you want to recognize, when an input image is presented a face recognition algorithm matches the input with a person in the gallery.

Face recognition is used in a wide area of applications, one of the most commonly used is Video Surveillance to match the identity of people to an existing database. Most recently used by social networking sites for tagging people on the uploaded media.

## II. Face recognition work flow

The first thing needed for creating a face recognition system is a Face Database Gallery also known as a Face Gallery, then perform a processing step know as feature extraction to discriminate the information on images, this also reduces the size of working elements by not working on pixels and working on features. Following this a machine learning algorithm is used to fit a model of the appearance of the faces on the gallery so you can discriminate between faces. The output of this stage is a Classifier, a model that will be used to recognize input images.

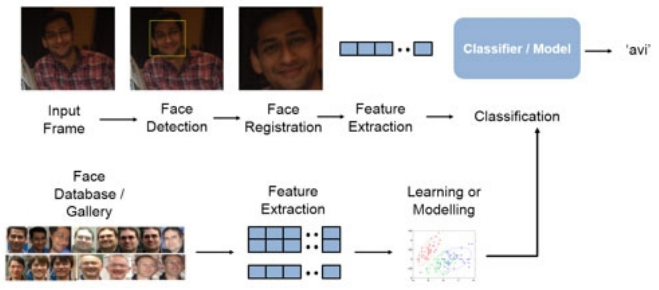


Figure 1. Recognition workflow

Second stage consists on a Face Detection preprocessing stage consisting on a face detection algorithm that finds the face of the person on the image. From the cropped face image the same feature extraction made on first step is made.

Finally, you run the output of the input image feature extractor to the classifier, the system output is a label or an indicator that holds the information of the person on the gallery the input comes from.

### A. Simple FaceRecognition

```
faceDatabase = imageSet('FaceDatabaseATT'
    ,'recursive');
```

Listing 1. Load FaceDatabaseATT

Following up the work flow the command imageSet loads the Database and stores on a variable at MATLAB.

```
%% Display Montage of First Face
figure;
montage(faceDatabase(1).ImageLocation);
title('Images of Single Face');

%% Display Query Image and Database Side
    -Side
personToQuery = 1;

galleryImage = read(faceDatabase(
    personToQuery),1);
figure;
for i=1:size(faceDatabase,2)
imageList(i) = faceDatabase(i).
    ImageLocation(5);
end
subplot(1,2,1);imshow(galleryImage);
subplot(1,2,2);montage(imageList);
diff = zeros(1,9);
```

Listing 2. Display Database and Query Image

Also, the command montage allows the display of one of the database objects. In this case, the subject one is displayed. Second part of the code displays the query image(input image) and the database side by side.

Next step is the extraction of HOG (Histogram Oriented Gradients) Features of the training set. The HOG Features are the commonly used method, instead of using pure raw pixels that don't get good results features have remarkably more positive results on illumination change and face deviation. The HOG process outputs an image that represents the structure of an object and that is invariant to the lighting.

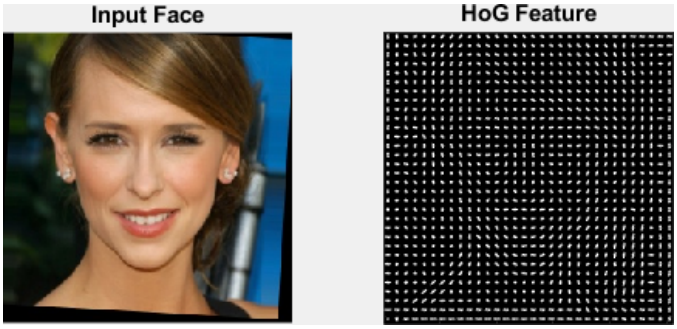


Figure 2. HoGFeatures extracted from image

The resultant HoG features give form semblance from the original image, the image contains data from the borders, placing and size of the objects making this more reliable when comparing images with different illumination or characteristics.

```
faceClassifier = fitcecoc(
    trainingFeatures,trainingLabel);
```

Listing 3. Create 40 class classifier using fitcecoc

Then we classify the database images as explained on the work flow. This is done by the fitcecoc command.

Once we have the classifier done, results can be obtained by testing images from the test set. HOG Features from input image are obtained and we input them to the deep learning neural network with the command predict. Then a string comparison is done and find the string label on our database leading with a match result.

## III. Results

The result on this webinar was achieved by learning the work flow of a simple face detector, although simple it contains the basic steps to deep learning. The number of images at database are not discussed on this webinar but the focus about the easy way of loading databases to MATLAB is mentioned.

About the feature extraction stage it would be interesting to try another type of extractors, HOG works well when working with structural analysis, MATLAB contains SURF, FREAK, BRISK, LBP and HOG descriptors on the Computer Vision System Toolbox each with one type of algorithm for extraction.

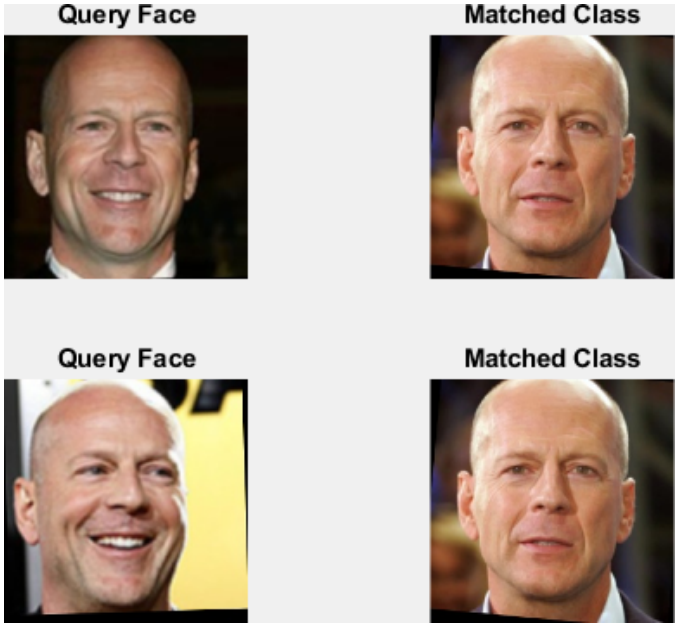


Figure 3. Query and Matched class

Training of modeling step is done by a machine learning algorithm named classifier. To create a deep learning on model, is needed to specify if it is a classification o regression model, the method and the inputs. The command *fitcecoc* is the fit command, followed up by the letter c meaning a classification type, and the letters coc, who are the method used.

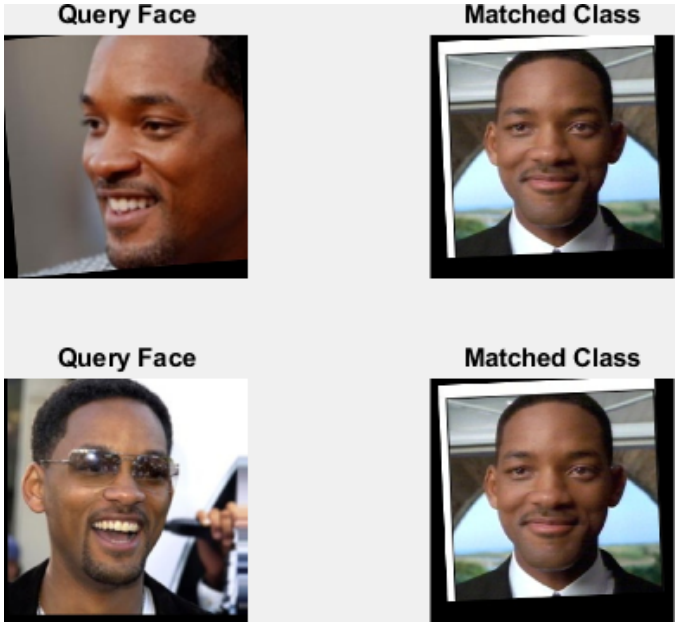


Figure 4. Predicted results

To use the model the *predict* method is used followed by the model declared before and the input features. The output of this method is a label in string format.

The extraction of features illustrates an advantage over pixel methods, Figure 4 shows a good match despite of the subject in a totally different angle and in the second one wearing sunglasses.


Figure 5. False positive match

If the condition of the featured image are changed the network experiments a mismatch, on Figure 5 we found a mismatch on the second run. The shadow on the chin, the cigarrette on the mouth could be some of things that changed from the original featured sample.

Results of the resulting network had an **90%** of accuracy, in the example the input consists of 10 images, and the networks fails to classify one of them. Testing the whole database gets 91% of accuracy. Training time on an Intel i7 4th generation processor took about 20 seconds, this time can be reduced by using the Parallel Computing Toolbox that allows us to used more cores or a local GPU, also it allows us to connect to the MATLAB Parallel cloud but not tested because a message about charges to the account.

## REFERENCES

[1] Valerie Leung, MathWorks, Reconnaissance faciale avec MATLAB: https://fr.mathworks.com/videos/face-recognition-with-matlab-99456.html Last visted: 24 Jan/2018

[2] Avi Nehemiah, MathWorks, Face Recognition with MATLAB: https://fr.mathworks.com/videos/face-recognition-with-matlab-98076.html