CSE-4084

Multimedia Systems

## Face Recognition Techniques with Python

**Instructor :** Prof. Cigdem Eroglu Erdem

**Researcher :** Cafer S. Yükseloğlu

**Abstract**

With the start of the big data age in the world and the commercial value of facial recognition technology, the prospects for facial recognition technology are very bright and there is a huge market demand. With this project, it aims to design a face recognition system based on photographic learning, photo recognition, real-time video definition detection. This project also aims to take into account the common problems in face recognition systems and identify them, these issues are briefly: the accuracy rate of the face recognition system during the actual check-in, the stability of the facial recognition system with real-time video processing, the performance impact of the face recognition system.

**Project Progress**

For this project, I allocated my time to 4 simple tasks: Finding the necessary information by scanning the literature and academic studies, creating an interface by combining the most appropriate tools, revealing the software result suitable for the project purpose, completing tests and academic studies. As a first step, I examined the data and face libraries required in this project. In this project, the library of Labeled Faces in the Wild, one of the human face photography libraries suggested by MIT, was used. Used MIT site is: <http://web.mit.edu/emeyers/www/face_databases.html>

Labeled Faces in the Wild:

<http://vis-www.cs.umass.edu/lfw/>

From this library I used the Images aligned with deep funneling. This was my main resource for teaching my code to known faces also I added my own face for test purposes. I started looking for libraries for python to teach the faces I identified. I found most useful one as opencv. OpenCV supports a wide variety of programming languages such as C++, Python, Java, etc., and is available on different platforms including Windows, Linux, OS X, Android, and iOS. Interfaces for high-speed GPU operations based on CUDA and OpenCL are also under active development. Some common usages for OpenCv are:

* Reading, Writing and Displaying Images
* Changing Color Spaces
* Resizing Images
* Image Rotation
* Image Translation
* Simple Image Thresholding
* Adaptive Thresholding
* Bitwise Operations
* Edge Detection
* Image Filtering
* Image Contours
* Feature Matching
* Face Detection

After using this library in Python, I had to use another library for easy coding which is face\_recognation by **Adam Geitgey.** In this library I could easily use two different face recognition technique which are HOG and CNN.

**[What is the difference between HOG, SIFT and CNN?](https://www.quora.com/What-is-the-difference-between-HOG-SIFT-and-CNN" \t "_blank)**

**HOG**

Stands for histogram of oriented gradients. Which is based on first order image gradients. The image gradients are pooled into overlapping orientation bins in a dense manner.

HOG is:

1. Based on first order image gradients pooled in orientation bins.
2. Dense (evaluated all over the image)
3. Hand engineered, no learning algorithms for HOG features.

**SIFT**

Which stands for scale invariant feature transform is similar to HOG only that SIFT is specifically a 128-dimensional vector that summarizes/describes a 16×16 window patch. The SIFT is obtained by dividing the 16×16 window into 4×4 bins. Each bin has 8 orientation bins or channels. So that makes the dimensionality of SIFT descriptor equal to

4×4×8 = 128

SIFT is:

1. Based on first order gradients
2. Course, that is, it is evaluated around scale invariant feature points obtained using the difference of gaussian (DoG) key point detector. There is a dense variant known as the dense-SIFT.
3. Hand engineered and thus does not learn the representation by itself, it is hard coded.

**CNN**

Stands for convolutional neural network, this is a hierarchical deep learning architecture. It is based on repeated convolutional operations which repeatedly filter the signal at each stage. The filters are trainable, that is, they learn to adapt to the task at hand during learning.

CNNs are:

1. Trainable feature detectors which make them highly adaptive. That is why they can achieve high accuracy levels in most applications such as image recognition. They can be trained end-to-end.
2. Mainly supervised deep learning models motivated by the primary visual cortex with alternating layers of convolutions and pooling layers.
3. They can learn low-level features similar to SIFT and HOG features from training examples alone, that is amazing. Thus, one can minimize feature engineering when it comes to using CNNs.

Thus, SIFT and HOG features are low-level features which don't make use of hierarchical layer-wise representation learning while the CNN is a hierarchical deep learning model which is able to model data at more and more abstract representations.

# **Haar Cascades**

# Object Detection using Haar feature-based cascade classifiers is an effective object detection method proposed by Paul Viola and Michael Jones in their paper, “Rapid Object Detection using a Boosted Cascade of Simple Features” in 2001. All possible sizes and locations of each kernel is used to calculate features , even a 24x24 window results over 160000 features.

1. Harr is a machine learning based approach to face recognition.
2. It is a machine learning based approach where a cascade function is trained from a lot of positive and negative images. It is then used to detect objects in another image.
3. Even 200 features provide detection with 95% accuracy

# **DNN**

The DNN model is included in opencv and is based on a single shot multibox detector. Also, this method is called SSD. Method for detecting objects in images using a single deep neural network. Discretizes the output space of bounding boxes into a set of default boxes over different aspect ratios and scales per feature map location. For 300×300 input, SSD achieves 72.1% mAP on VOC2007 test at 58 FPS on a Nvidia Titan X and for 500×500 input, SSD achieves 75.1% mAP, outperforming a comparable state of the art Faster R-CNN model.

**Accomplishment**

Python not compiled program, thus slower than compiled based programs. However, thanks to its compatibility it can be used in multiplatform and well working.

There are many things that affect the speed of face recognition systems. But we can say basically we could say Hardware and Software.

For hardware part I have an NVDIA GTX 1080 GPU and i7 7700K CPU with this system I tried to get measurements. My tests show that 12gb ram for face locating took 1 second for HOG but 30,5 second for CNN. With CUDA I took new results as 2,5 seconds for CNN. When it came to the face recognition depending to documentation, I could order them HARR HOG DNN and CNN for speed faster to slower.

* HARR 60 FPS
* HOG 56 FPS
* DNN 30 FPS
* CNN 10 FPS

But it is different when it came to the accuracy in such feature performance could be negligible. If I want to order to their accuracy for Large Faces:

* CNN
* DNN
* HARR
* HOG

However, this project showed me the results are little different than documentations. In my testes I got that without GPU the order is for speed:

HOG 60 FPS

HAAR 50 FPS

DNN 40 FPS

CNN ~10 FPS

With the GPU open the results are:

Only difference is CNN 100 FPS

Thus, under right hardware learning face and detecting face is very good and fast for accuracy I saw that DNN is very accurate in smaller faces.

## Conclusion

## Through this project, I have determined which of these tools and methods is the best and changes according to the place and event we will use. If we want a definitive result, what I learned from this project and would be recommending DNN or HOG. Since we cannot know exactly what size the faces will be in many areas of use, DNN will be the most stable in both an average speed and different angles. For speed in large scale images I would like to use HOG because it is fast in lower hardware.

## REFERENCES

* [Color FERET DATABASE V’ by NIST](https://www.nist.gov/itl/products-and-services/color-feret-database)
* [MIT FACE Databese List For Other Resources](http://web.mit.edu/emeyers/www/face_databases.html)
* SARKAR, & BALİ, & SHARMA (2018) Practical Machine Learning with Python. ISBN-13 (pbk): 978-1-4842-3206-4
* Adjabi & Ouahabi & Benzaoui & Taleb-Ahmed (2020) Past, Present, and Future of Face Recognition
* “Digital Video Processing” by M. Tekalp, Prentice Hall, 1995, ISBN 0-13-190075-7.
* "Digital Video: An Introduction to MPEG-2", B.G. Haskell, A. Puri, and A.N. Netravali, Chapman & Hall, 1997. A detailed description of the MPEG-2 video compression standard as well as a good overview of video compression.
* "Motion Analysis and Image Sequence Processing", edited by M.I. Sezan and R.L. Lagendijk, Kluwer Academic Publishers, 1993. Individual chapters cover a variety of topics in motion analysis and video processing.