*International Journal for Modern Trends in Science and Technology*, 6(8): 285-290, 2020

ISSN: 2455-3778 online
DOI: https://doi.org/10.46501/IJMTST060846
Available online at: http://www.ijmtst.com/vol6issue08.html

# Face Detection System in Python using HoG

Shaik Mohammed Imran[1] | Dr. Ranjith P.N.[2] | Farooq Mohammmed[1]

[1]Computer Science and Engineering, Guru Nanak Institute of Technology.
[2]Computer Science and Engineering, Hindustan Institute of Technology and Science.

**To Cite this Article**
Shaik Mohammed Imran, Dr. Ranjith P.N. and Farooq Mohammmed, "Face Detection System in Python using HoG Importance of Dynamic Analysis for RCC Structures", *International Journal for Modern Trends in Science and Technology*, 6(8): 285-290, 2020.

**Article Info**
Received on 24-July-2020, Revised on 12-August-2020, Accepted on 18-August-2020, Published on 25-August-2020.

## ABSTRACT

*Face-recognition technology is commonly recognized as having played a key role in the CCTV surveillance system because it does not require the object's cooperation of the individual. It is generally accepted that the face recognition had played a significant and important role in surveillance system. We use deep learning to extract facial embedding's from each face, train a face-recognition model on the embedding's, and then finally recognize faces with Open CV in both images and video streams. We may use a simple k-NN model while classifying. This paper has been implemented using the HoG& K-Nearest Neighbors Algorithm and is a very simple type of face recognition.We can do using cnn in both ways one way is to detect faces in an image so we can use HOG with SVM (linear) or HAAR cascade like in this paper or by deep metric cnn which takes an input face and then computes 128d quantification of an image.*

***KEYWORDS:****embedding, hog, k-nearest neighbor, train*

## I. INTRODUCTION

Face recognition technology is widely acknowledged and commonly accepted that it has played a crucial role in cctv closed circuit television surveillance system Facial recognition system was even used to help for confirming the identification of Laden after he was killed in a U.S. raid Here we implement face recognition for both still images and video streams and we are outputting a real-valued feature vector instead of a single label. In our paper, we propose a face detection and recognition framework that uses python programming language along with HoG method. For the method of the face identification here in this paper we have used different techniques using dlib library and haar cascade classifier. "Either we can implement dlib for face detection, which using a mixture of HOG i.e.,(Histogram of Oriented Gradient) or OpenCV's" [3]. The biggest downside to this face tracking algorithm is that each and every input frame must have a separate face detector running on.

## II. METHODOLOGY

This study is an applied research in terms of purpose and is a quasi-experiment in terms of data collection method

1.**Robust and real-time face detection** plays a vital role in many of the application, often as a part of a facial recognition system. It is also used in video surveillance, image database management. Some recent digital cams like DSLR use face detection for autofocus.

Facial recognition process generally involves two stages:

- Face Detection where the image/video is searched to seek out a face, then the corresponding image is processed to crop and extract the person's face for easier recognition.

Face Recognition where that detected and processed face is compared to a database of known faces, to make a decision like who that person is.

The Eigen face technique is taken into consideration that the only simplest method of accurate face recognition, but many other methods are much more complicated methods or combinations of multiple methods are slightly more accurate. The face space is defined by the 'eigen faces', which are the eigenvectors of the set of faces[5].

"OpenCV was started at Intel co-operation in 1999 by Gary the person named Bradski that is for the needs of enhancing the research in and commercial applications of computer vision within the world and, for Intel"[13].

OpenCV has the huge advantage of being a multi-platform framework; it supports both Windows and Linux OS.

**2.Feature Selection using Haar like Features**

In 2001, Viola et al. first introduced the haar-like features. Haar cascade is a machine learnigHoG for face detection[11]. The haar-like characteristics are rectangle attributes and value is that the sum of pixels in black district subtracts the sum of pixels in white district. Rainer Lien hart had done an extended set of straightforward haar-like features which significantly enrich the essential set of simple haar like features, and may get a far better hit rate. Two-rectangle features are A‟ and B‟, C‟ is three rectangles attribute and D‟ is four-rectangle attribute. At a size of 24x 24, there are quite 180,000 rectangle features. Face attributes are abstracted from the input image and are used to train the classifier, modify weights.

**3. HoG for face detection**

The advancements in CV with Deep Learning has been constructed and perfected with time, essential over one specific algorithm — a Histogram of Oriented Gradients. The distribution (histograms) of gradient directions (oriented gradients) is used as features in the HOG function descriptor. Gradients (x and y derivatives) of an image are useful because the gradient magnitude is high around edges and corners (regions with sudden changes in intensity) and we know that edges and corners pack in far more details about the shape with artifacts than flat regions.

**4. k-NN Algorithm for classification.**

K-NN works by finding the distances between the query and all the examples in the data, by selecting the number examples found (K) closest to the query, and then by voting for the most regular label in the case of classification[12].

The k-Nearest Neighbor classifier is by far the most simple machines learning/image classification algorithm.

Within, this algorithm relies simply on the distance between feature vectors, just like constructing a search engine for images.. Here simply put, the k-NN algorithm categorizes the unknown data points by locating the most common class among the k-closest examples. Each data point in the k closest examples casts a vote and the category with the most votes wins fairly.

We need to specify a distance metric or similarity function to apply the k-nearest Neighbor classification by Euclidean distance:

$$d(\boldsymbol{p}, \boldsymbol{q}) = \sqrt{\sum_{i-1}^{N} (q_i - p_i)^2}$$

Figure 1: The Euclidean distance.
p,q: are Euclidean vectors

## III. PROPOSED SOLUTION

**1.dlib facial recognition-**

The output function vector for the dlib facial recognition network is 128-d i.e., it's a list of 128 real-valued numbers which is used to determine the face. Dlib is a repository of general use applications. We can render real world machine learning applications using dlib toolkit. Dlib is a general-purpose software repository. Python provides face_recognition API which is made through dlib's face recognition algorithms.

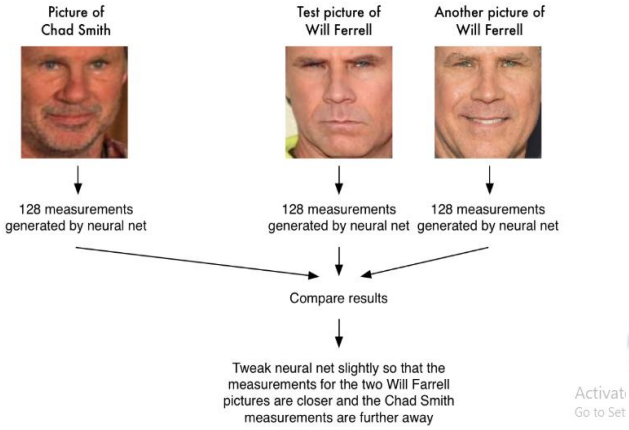


Figure 2 Using triplets the training to the network.

Here the triplet consists of 3 unique face images — 2 of the 3 are the same person. The Neural Network generates a 128-d vector for each of the 3 face images.

For the 2nd face images of the same person, we improve the neural network weights to make the vector closer through distance metric. In the above we provide three images to the neural network:

Our neural network quantifies the faces, constructing the 128-d embedding (measure) for each. Two of these images are the same person's faces for example. The third image from our dataset is a random face and is not the same individual as the other two images.

This face recognition API allows us to achieve face detection, real-time face tracking and face recognition applications. Here can we use two different technics CNN and HoG for recognition

based on dlib's face recognition system with using face_recognition.

Our neural network construction for face recognition is based on ResNet-34, but with fewer layers and the number of filters reduced by half [1].

Davis King trained the network itself on a database of some ~3 million images. The network contrasts with other outstanding approaches on the Labeled Faces in the Wild (LFW) dataset, achieving 99.38 per cent accuracy [2].

For dataset the training code is obviously also available, since that sort of thing is basically the point of dlib. You have to first find all details on training and then model.

**2.Install the libraries for face detection-**

A library called dlib which is used to construct our face embeddings used for the actual recognition process .and face _ recognition facial recognition functionality.

**3.Installing dlib without GPU support-**

Here the GPU is nothing but Graphics processing unit GPUs are optimized for training artificial intelligence and deep learning models as they can process multiple floating point computations simultaneously. Architecturally, the CPU consists of only a few cores and lots of cache memory that can accommodate a couple of program threads at once.

A GPU is smaller than a CPU but tends to have more logical cores i.e., arithmetic logic units or ALUs, control units and memory cache. In contrast, a GPU is composed of hundreds of cores that can handle thousands of threads simultaneously [7].

They have a huge number of cores, which allows for better calculation of multiple parallel processes. Additionally, calculations in deep learning need to handle large chunks of data — this makes a GPU's memory bandwidth most suitable.

There are a few deciding parameters to determine whether to use a CPU or a GPU to train a deep learning model like:

Memory bandwidth,
Dataset size, and Optimization.

**4.Face detection dataset-**

Here we apply face recognition to a fragment of the characters from films like in 2018 Jurassic World: Fallen Kingdom was released.

So for that we use API from Bing Image Search API to build an image dataset for deep learning which is part of Microsoft's Cognitive Services used to bring AI to vision, speech, text etc. We parse two command line arguments:

First Querry: The image search query you're using, which could be a search based upon keyword for sample.

Second Output: The output directory for your images in the form of sub-directories

After gathering this dataset of images we'll then create the 128-d embedding's for each face in the dataset and also we will use these embeddings to recognize the faces of the characters in both images and video streams.

**STEPS:**

Step 1 Build a dataset by using the Bing API.

Step 2 is for Encodings (128-d vectors) for faces are built.

Step 3 is to recognize faces based on encodings from your dataset in a single image.

Step 4 for recognizing the faces in a live video stream from your webcam and output a video.

Step 5 for recognizing the faces in a video file residing on the disk either a pen drive or a hard disk and output the processed video to disk.

Step 6 for facial recognitions as well as encodings which are generated from your dataset that we build.

And finally for building the embedding's for recognizing the faces after a dataset of images is created through search bing API key.

**5.Encoding the face**

Here it's easier to use the pre-trained network and then use it to construct 128-d embedding for each of the 218 faces in our dataset. During classification, here we can use a simple k-NN model with votes to make the final face classification.

OpenCV uses color channels in BGR, but the dlib expects RGB. The face recognition module uses dlib. Here we pass two limits for recognition using face recognition. Face location method:

Rgb(Red, Blue, Green): Our RGB image.

Model: Either cnn or hog. As we know the CNN method is more precise but it is slower. HOG is faster but less precise.

Here we encode and name to the appropriate list like known Encodings and known Names similarly its done for 218 datasets[6].

**6.Recognize faces**

After building our 128-d face embedding for every image in our dataset, we are now ready to recognize faces in image using OpenCV, Python, and deep learning[10].

For calculating the Euclidean distance between the candidate embedding and all faces in our dataset:

When the distance is below some threshold level, the greater the threshold will be, the more strict will be our facial recognition system and we will return true, indicating that the faces will match. If the distance is above the threshold then we must return false as the faces don't match.

If the distance is above the acceptance threshold then we must return false as the faces do not match.

The CNN face recognizer must only be used in real-time that too if you are working with a GPU then only you can use it with a CPU, but can expect < 0.5 FPS which makes for a rough video. If you are using a CPU, you should use the HoG method and expect adequate speeds[4].

Importantly in this we read the frame, preprocess, and we detect the face bounding boxes and then calculate encodings for each bounding box. Then we attempt to match the face. If matches are found then we count the votes in the dataset for every name. Instead we collect the highest count of votes and that is the name associated with face.

Here we applied the recognition from the movie Jurassic Park and my own web cam.

## IV. IMPLEMENTATION

### A. Encode Face

```
# to develop as well as parse the arguments

ap = argparse.ArgumentParser()

ap.add_argument("-i","--dataset", required=True,

help="path to input directory of faces+ images")

ap.add_argument("-e","--encodings",
required=True,

help="path to serialized db of facial encodings")
```

```
ap.add_argument("-d","--detection-method",
type=str, default="cnn",

help="face detection model to use : either `hog`")

args = vars(ap.parse_args())
```

**B. Recognize Image**

```
# load the known set of faces as well embeddings

print("[INFO] loading encodings...")

data = pickle.loads(open(args["encodings"],
"rb").read())

# load the input image and convert it from Blue
#Green Red to Red Green Blue

image = cv2.imread(args["image"])

rgb=cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

# for detecting the coordinates (x , y) of the
#bounding boxes corresponding to the attempt to
#fit each face of the known encoding input image

matches =
ace_recognition.compare_faces(data["encodings"],

encoding)

name = "Notknown"

# drawing predicted face name to be on the image

Cv2.rectangle(image, (left, top), (right, bottom),
(0, 255, 0), 2)

Cv2.FONT_HERSHEY_SIMPLEX,0.75, (0, 255,
0), 2)
```

**C. Recognize Video**

```
# loading the known faces and embeddings

print("[INFO] loading encodings...")

data = pickle.loads(open(args["encodings"],
"rb").read())

# initialize the video stream and pointer to
#output video file, then allow the camera

print("[INFO] starting video stream...")

vs = VideoStream(src=0).start()

writer = None

time.sleep(2.0)

# detect the (x, y) co-ordinates of the box

# of every face in the input frame, then we
# measure the embedded facials

boxes = face_recognition.face_locations(rgb,

model=args["detection_method"])

encodings

= face_recognition.face_encodings(rgb, boxes)

names = []
```

**D. Search API bing**

```
# set your Microsoft Cognitive Services API key

# (1) the maximum number of results

# (2) and finally the group size

API_KEY = "INSERT_YOUR_API_KEY_HERE"

MAX_RESULTS = 100

GROUP_SIZE = 50

# while downloading the images the requests
#library number of exceptions that can be
#thrownso list of them now so we can filter

EXCEPTIONS = set([IOError, FileNotFoundError,

exceptions.RequestException,
exceptions.HTTPError,

exceptions.ConnectionError,
exceptions.Timeout])
```

## V. RESULTS

The result of face detection is shown in Figure. Those are the frames extracted from the HD video streaming. Sometimes, face detection algorithm may get more than one result even there is only one face in the frame.

In this paper we also implement the face tracking application in Python language by using face detection. This method is confirmed and the restrictions of the scheme are observed through testing and debugging our codes. And then, limited by Python performance, we shift to OpenCV to evaluate the speed of this face tracking scheme.

In this paper we have developed a system for face

detection and recognition using HoG. It is used to detect and recognize human faces. The images of the persons are the datasets which are defined and trained before recognizing.


Figure 3 Output from video


Figure 4 Output from webcam

For better face recognition and detection small features can be improved. In the coming future, as technology advances, more advance features will be added to the system.

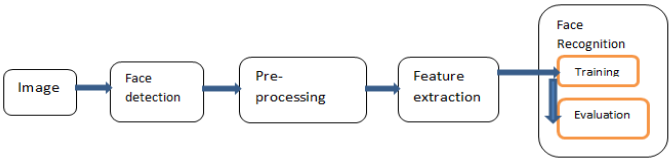


Figure 5: Architecture of face recognition system
Here the process is as follows under:

Image→Facedetection→Pre-processing→Feature →Extraction→Face Recognition

## VI.DISCUSSION

In this paper we established a face detection and recognition framework using opencv. It is designed for the identification and recognition of human faces. Person's images are the datasets that are described and trained prior to recognition. Smaller features can be enhanced for better facial recognition and detection.

## REFERENCES


[1] Deep Residual Learning for Image Recognition Kaiming He Xiangyu Zhang Shaoqing Ren Jian Sun Microsoft Research Computer Vision Foundation https://www.cvfoundation.org/openaccess/content_cvpr_2016/papers/He_Deep_Residual_Learning_CVPR_2016_paper.pdf.

[2] Labeled Faces in the Wild http://vis-www.cs.umass.edu/lfw/.

[3] https://pythonprogramming.net/haar-cascade-object-detection-python-opencv-tutorial/.

[4] http://thursday.moecoin.com/cnn-image-recognition.

[5] M.A.Turk and A.P. Pentaland, "Face Recognition Using Eigenfaces", IEEE conf. on Computer Vision and Pattern Recognition, pp. 586-591, 1991.

[6] https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53.

[7] Hong-W. Ng, S. Winkler. A data-driven approach to cleaning large face datasets. Proc. IEEE International Conference on Image Processing (ICIP), Paris, France, Oct. 27-30, 2014.

[8] https://towardsdatascience.com/what-is-a-gpu-and-do-you-need-one-in-deep-learning-718b9597aa0d#:~:text=Why%20choose%20GPUs%20for%20Deep,computation%20of%20multiple%20parallel%20processes.

[9] Open Source Computer Vision Library Reference Manual-intel[Media].

[10] "Introduction to OpenCV", [Online] Available:www.cse.iitm.ac.in/~sdas/courses/CV_DIP/PDF/INT.

[11] Paul. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001, Kauai, HI, USA, 2001, pp. I-I, doi: 10.1109/CVPR.2001.990517.Michael.

[12] https://en.wikipedia.org/wiki/Knearest_neighbors_algorithm.

[13] http://foss2serve.org/index.php/MouseTrap_Dev_Help/opencv.

[14] http://blog.dlib.net/2017/02/high-quality-face-recognition-with-deep.html.