SVU-International Journal of Engineering Sciences and Applications (2023) 4(1): 79-84
Print ISSN 2785-9967 | Online ISSN 2735-4571
DOI 10.21608/svusrc.2022.161634.1072

Gamma Effect in Face Detection Methods

Mahmoud Hardan[1,✉], Mohamed Abdel-Nasser[2], Adel B. Abdel-Rahman[1,3]

Abstract Face detection could be a technology capable of finding a person's face in an image or video frame. Face detection is one of the foremost widely used computer vision applications. it's a fundamental problem in computer vision and pattern recognition. Face detection could be a critical beginning in face recognition systems, with the reason for localizing and extracting the face location from the background. Within the last decade, multiple face feature detection methods have been introduced. In python, there are plenty of face detection methods like OpenCV library, dlib library, MTCNN algorithm, and ….etc. During this paper, we present a test compression between the face detection methods. There are many face detection methods. We decide on the foremost used like OpenCV with Haar cascade and LBP models, MTCNN algorithm and dlib with CNN model and HOG model in python and therefore the effect of gamma value on its accuracy. The chosen face detection methods measured the period of time for each gamma value.

Keywords:. Face Detection, Facial Recognition, Python, Gamma, MTCNN, Haar cascade, dlib, HOG.

1 Introduction

Face detection [1] is one in every of the foremost widely used computer vision applications. It's a fundamental problem in computer vision and pattern recognition. Face detection may be a technology employed in a spread of applications to acknowledge faces in digital images. Face detection will be considered a special case of target detection. In object detection, the task is to search out the position and size of all objects of a given class within the image. Face detection is the first stage of face recognition. Fig. 1 shows the relation between face detection and face recognition.

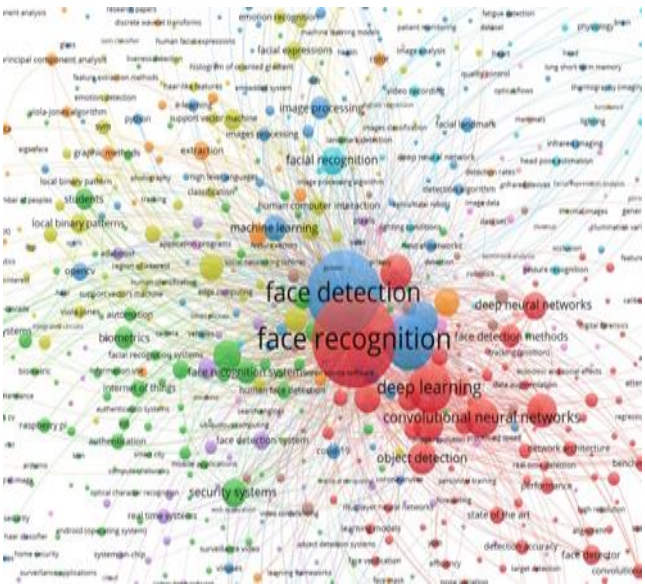

Fig. 1 the relation between the face detection and face recognition

In the last decade, multiple face feature detection methods

Received: 8 September 2022/ Accepted: 23 December 2022
✉Corresponding Author
1. Electrical Department, Faculty of Engineering, South Valley University, Qana, Egypt
engmahfah2005@gmail.com
2. Faculty of Engineering, Aswan University
Aswan , Egypt, mohamed.abdelnasser@aswu.edu.eg
3. School of Electronics, Communications and Computer Engineering ,E-JUST, Egypt, adel.bedair@ejust.edu.eg

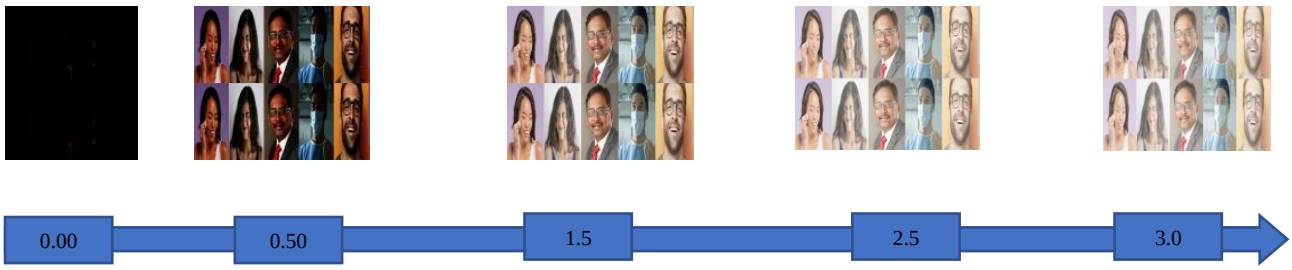

Fig. 2 the effect of gamma on the image. Gamma=0

are introduced. In python, there are lots of methods for face detection like as OpenCV library [2, 3], dlib library [4, 5], MTCNN algorithm [6, 7], and …. etc. Face detection may be a critical opening in face recognition systems, withthe reason for localizing and extracting the face location from the background. It additionally has quite few functions in areas like content-based photo retrieval, video coding, video conferencing, crowd surveillance, and clever human–computer interfaces. The goal that we seek during this paper is to get the effect of adjusting the gamma factor with different libraries and different models like OpenCV with Haar cascade, OpenCV with LBP, dlib with HOG, and dlib with CNN and also the average time it takes for every model to detect faces, find the simplest model will be used and best value of gamma is work on that.

1.1. Gamma Factor

Gamma—gamma-correction—simply refers to the operation to encode the linear values the camera records into a non-linear relationship (or the reversal of this process in decoding). the explanation we must gamma correct images lies within the historical must accommodate the exponential output response of the old ray Tube (CRT) displays. The luminance would arc up from black to white because the input voltage increased [8]. Fig. 2 show the effect of gamma on the image.

1.2. OpenCV Library

OpenCV is that the huge open-source library for the pc vision, machine learning, and image processing and now it plays a significant role in real-time processing which is extremely important in today's systems. By using it, one can process images and videos to spot objects, faces, or maybe handwriting of a people. When it integrated with various libraries, like NumPy, python can process the OpenCV array structure for analysis. to spot image pattern and its various features we use vector space and perform mathematical operations on these features.

1.2.1. Haar Cascade classifiers

Haar Cascade classifiers are a good way for object detection. Haar Cascade could be a machine learning-based approach where lots of positive and negative images are wont to train the classifier. Positive images – These images contain the pictures which we would like our classifier to spot. Negative Images – Images of everything else, which don't contain the article we would like to detect [9]. Fig.3 show how the Haar Cascade classifiers work.

Fig 3 Haar Cascade classifiers work [9]

1.2.2. Local Binary Pattern (LBP)

Local Binary Pattern (LBP) may be a simple yet very efficient texture operator which labels the pixels of a picture by thresholding the neighborhood of every pixel and considers the result as a binary number, Fig. 4. LBP may be a visual descriptor it may be used for face recognition tasks[10]. Parameters: the LBPH uses 4 parameters:

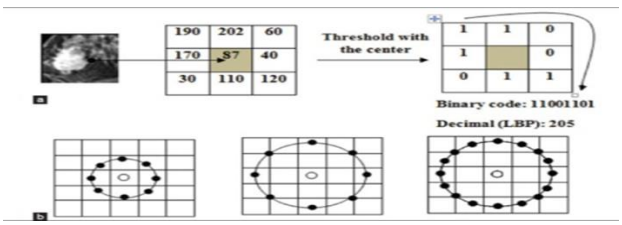

Fig. 4 Local Binary Pattern (LBP) method [4]

• Radius: the radius is employed to make the circular local binary pattern and represents the radius round the central pixel. It's usually set to 1.

• Neighbors: the amount of sample points to make the circular local binary pattern. Detain mind: the more sample points you include, the upper the computational cost. It's usually set to eight.

• Grid X: the quantity of cells within the horizontal direction. The more cells, the finer the grid, the upper the dimensionality of the resulting feature vector. It's usually set to eight.

• Grid Y: the quantity of cells within the vertical direction. The more cells, the finer the grid, the upper the dimensionality of the resulting feature vector. It's usually set to eight.

1.3. Dlib Library

Dlib could be a modern C++ toolkit containing machine learning algorithms and tools for creating complex software in C++ to resolve world problems. it's utilized in both industry and academia in a very big selection of domains including robotics, embedded devices, mobile phones, and huge high performance computing environments. Dlib's open-source licensing allows you to use it in any application, freed from charge [11].

1.3.1. Histogram of oriented gradients (HOG)

Histogram of Oriented Gradients, also called HOG, may be a feature descriptor just like the Canny Edge Detector; SIFT (Scale Invariant and have Transform). it's utilized in computer vision and image processing for the aim of object detection. The technique counts occurrences of gradient orientation within the localized portion of a picture[12]. Figure [5,6] show the Using of HOG.

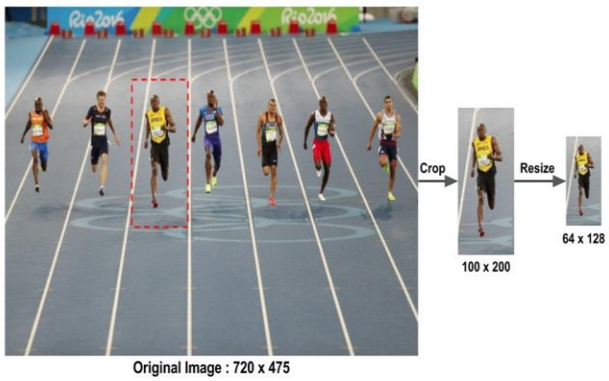

Fig. 5 Input image to HOG

Fig. 6 Left : Absolute value of x-gradient. Center : Absolute value of y-gradient. Right : Magnitude of gradient [12]

1.3.2. Convolutional neural network model (CNN):

CNN (Convolutional Neural Network), Fig.7 could be a class of deep networking, it works rather well for non-frontal faces at odd angles where HOG based detector isn't good at it. Dlib with CNN based face detection, trained with innumerable images, and stored the trained model in an exceedingly "mmod_human_face_detector.dat" file. Dlib with CNN can to detect another object with another '.dat' file [13].

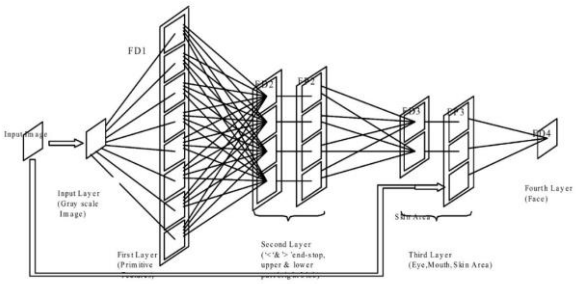

Fig. 7 CNN for face detection [13]

1.4. MTCNN Algorithm

Multi-Task Cascaded Convolutional Neural Networks [6] could be a neural network that detects faces and facial landmarks on images. MTCNN is one in all the foremost popular and most accurate face detection tools today. The MTCNN work by resizing the image, scale the first image to different scales, and generate a picture pyramid. Then the pictures of various scales are sent to 3 sub-networks for training Fig .8, the aim is to detect faces of various sizes, to attain multi-scale target detection. The three sub-networks are: -

1.4.1. P-Net (Proposal Network): -

P-Net may be a candidate network for the face region. The input of the network may be a 12x12x3 image. After 3 layers of convolution, it's judged whether there's a face

within the 12x12 image, and also the regression of the face frame and also the person are given. Key points of the face.[6].

1.4.2. R-Net (Refine Network): -

As is seen from the network diagram, it's only due to the difference between the network structure and also the P-Net network structure that a totally connected layer is added, so it'll achieve a more robust suppression of false-positive effects. Before inputting R-Net, it has to be scaled to 24x24x3. The output of the network is that the same as that of P-Net. the aim of R-Net is to get rid of an oversized number of non-face frames [6].

1.4.3. O-Net (Output Network): -

As are often seen from the network diagram, this layer has yet one more convolutional layer than the R-Net layer, therefore the processing result are more refined. The input image size is 48x48x3, and therefore the output includes the coordinate information of the N bounding boxes, the score and also the position of the key points [6].

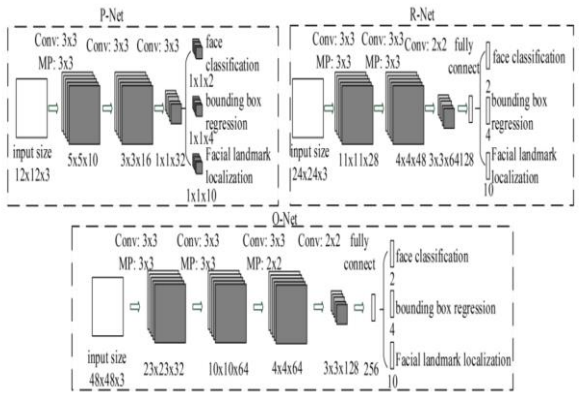

Fig. 8 MTCNN Design [6]

In this study, in the domain of face detection, we compare the face detection methods in python. The rest of the paper organizes as follows: In section 2, we present the methodology used to test the face detection methods under test. In section 3, we discuss the results and discuss our experiments. Finally, section 4 concludes the paper.

2. Methodology

This section describes the methodology which we used to compare the face detection methods (OpenCV with Haar cascade, OpenCV with LBP, dlib with HOG, and dlib with CNN). Table (1) describes the computer specifications used in this experiment.

Table 1 Computer Specification

Processor	Intel(R) Core (TM) i5-2430M CPU @ 2.40GHz 2.40 GHz
RAM	12.0 GB
GPU	GF119-CORES 48-TMUS 8-ROPS 4-MEMORY SIZE 1024 MB-MEMORY TYPE DDR3-BUS WIDTH 64 bit

The following table (2) shows the steps of the algorithm that was used in this paper:

Table 2 Step of Algorithm

step	
1	Set gamma=0
2	Set timer=0
3	Adjust the image using gamma value
4	Using one of test face detection model to detect faces
5	Stop timer
6	gamma=gamma+0.1
7	Repeat steps 2-6 while gamma<=3

3 Results

In this section, we will describe the results that were extracted from the models that were used. Our process relied on inserting an image consisting of 5 faces that were repeated twice (5*2) and examining the results and the time used to discover the faces. Figure (2) shows the image used as input. Fig. (9) show the accuracy resulting from all models concerning gamma.

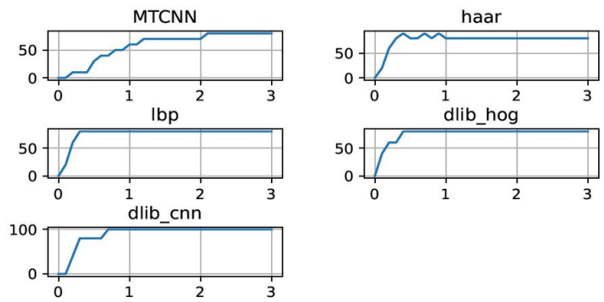

Fig. 9 The Accuracy of Face Detection Models

Fig.10, which shows the efficiency of each model with changing gamma, shows us with small values of gamma from 0.0 to 0.1; the detection rate is the lowest possible value. Whereas higher gamma values from 1.0 to 3.0 indicate higher efficiency. But the best model with lower gamma values is the Haar cascade, where its efficiency reaches 60% when the gamma value is 0.2 and then rises to a value ranging from 80% to 90%.

The best model with high gamma values was dlib with CNN model where its value ranged from 90% starting from gamma equal to 0.3 until gamma equal to 0.6 and the value settled at 100% starting from gamma 0.7 to gamma of 3.0

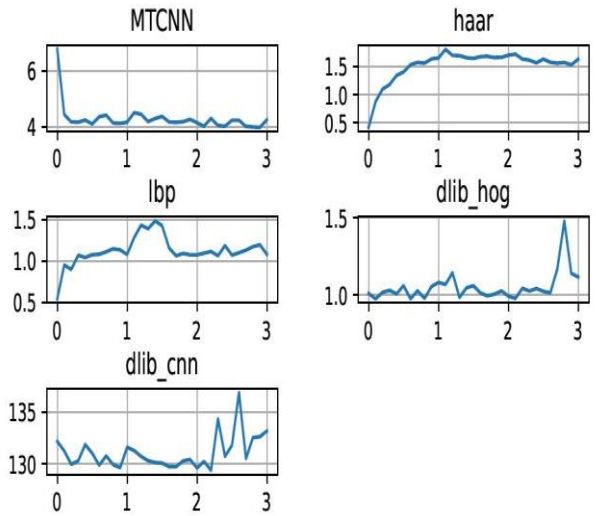

Fig. 10 Elapsed Time of Face Detection Models

Finally, Figure (11) shows the time used for each model to detect faces concerning gamma. As it is clear from the figure that the average time elapsed for each of the Haar cascade, LBP, and finally dlib with HOG is 1.521853281,1.119600397,1.051145784 seconds in order and the lowest value that is always considered a gamma value less than 0.2 and this is shown in Table (3) where it does not exceed a second One for these models, and in cases of high gamma, it is only less than two seconds.

Whereas dlib with CNN, which is the most efficient, takes a time ranging from 129 seconds for the lowest gamma value and 136 seconds for the highest gamma value.

As for MTCNN, which is one of the most common nowadays in face detection, the time used by it to detect faces is no more than 7 seconds and not less than 3.5 seconds. But it was noticed that the time in MTCNN is inverted, the higher the gamma, the lower the time needed to detect faces

Table 3 Avarage Elapsed Time of Face Detection Models (second)

	Average	Min	Max
MTCNN	4.290299	3.982811	6.780323
HAAR	1.521853	0.419168	1.805049
LBP	1.1196	0.538264	1.481038
DLIB_HOG	1.051146	0.974717	1.481223
DLIB_CNN	131.0512	129.3556	136.8949

Another point that emerged from the relationship between the time it takes for the model to find faces and change gamma values is that MTCNN takes longer time in low gamma values less than 0.5 than it takes in gamma values higher than 0.5. As for OpenCV with LBP, the highest time taken was from A gamma value of 1.0 to a gamma value of 1.5 as well as OpenCV with Haar cascade the highest time spent was starting from a gamma value of 1.0 to a gamma value of 1.2. In the opposite direction, dlib wth HOG starts from a value of 2.5 to 3.0 and dlib with CNN starts from a value of 2.2 to a value of 3.0.

4 Conclusion

As a clear result, changing the gamma value affects the performance of the different models. The fastest model is OpenCV with LBP and the slowest is dlib with CNN with an average time of 1.1196,131.0512 sec in the same order. Changing the gamma value affects the efficiency of the different models and affects the time that each model needs to detect faces. It leads to an increase in the efficiency of the models with an increase in the gamma value and vice versa. And dlib with CNN is the best because with values less than one gamma, its efficiency rose to 100%, followed by dlib with HOG. But the best in time and efficiency is dlib with HOG, its efficiency is not less than 80% with values starting from 0.5 gamma value and the elapsed time does not exceed 1.5 seconds

References

[1] E. Hjelmås and B. K. Low, "Face detection: A survey," Computer vision and image understanding, vol. 83, no. 3, pp. 236-274, 2001.

[2] R. Laganière, OpenCV 3 Computer Vision Application Programming Cookbook. Packt Publishing Ltd, 2017.

[3] D. S. Vasantha, "FACE MASK DETECTION SYSTEM USING DEEP LEARNING."
[4] "http://dlib.net/python/." (accessed.
[5] N. LATTRAG and A. BAGHDADI, "Face Recognition Based Access Control Systems," UNIVERSITY of M'SILA, 2022.
[6] N. Zhang, J. Luo, and W. Gao, "Research on face detection technology based on MTCNN," in 2020 international conference on computer network, electronic and automation (ICCNEA), 2020: IEEE, pp. 154-158.
[7] P. Kavitha, D. Shanmugam, T. Akash, T. KaranRaj, and A. G. Gowtham, "PAYMENT TRANSACTION USING FACE-RECOGNITION," 2022.
[8] "Understanding Gamma in Photography." https://www.japanistry.com/understanding-gamma-in-photography/ (accessed.
[9] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition. CVPR 2001, 2001, vol. 1: Ieee, pp. I-I.
[10] T. Ahonen, A. Hadid, and M. Pietikäinen, "Face recognition with local binary patterns," in European conference on computer vision, 2004: Springer, pp. 469-481.
[11] "DLib C++ libraty." http://dlib.net/ (accessed.
[12] O. Déniz, G. Bueno, J. Salido, and F. De la Torre, "Face recognition using histograms of oriented gradients," Pattern recognition letters, vol. 32, no. 12, pp. 1598-1603, 2011.
[13] K. Grm, V. Štruc, A. Artiges, M. Caron, and H. K. Ekenel, "Strengths and weaknesses of deep learning models for face recognition against image degradations," Iet Biometrics, vol. 7, no. 1, pp. 81-89, 2018.