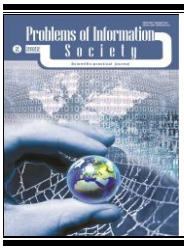

www.jpis.az

13 (2)
2022

Real-time face detection on a Raspberry PI

Leyla G. Muradkhanli[1], Eshgin A. Mammadov[2]

[1,2]Department of Process Automation Engineering, Baku Higher Oil School, Khojaly ave. 30, AZ1025, Baku, Azerbaijan

[1]leyla.muradkhanli@bhos.edu.az

orcid.org/0000-0001-6149-4698

ARTICLE INFO

http://doi.org/10.25045/jpis.v13.i2.05

Article history:
Received 17 January 2022
Received in revised form
18 Mart 2022
Accepted 06 June 2022

Keywords:
Face Detection
Raspberry PI
Histogram of Oriented Gradients
Support Vector Machines
Internet of Things

ABSTRACT

The article describes the implementation of different face detection algorithms to capture human faces from real-time video frames using a Raspberry PI microprocessor. This article examines this issue, proposes the implementation of two distinct real-time face detection algorithms, and presents a comprehensive architectural design. Used methods include Haar Cascades which is known as Viola-Jones algorithm, and Histogram of Oriented Gradients + Linear Support Vector Machines algorithm. The algorithms are implemented with the help of the OpenCV and Dlib libraries, and the Python programming language was used to build the face detection system. The OpenCV and Dlib libraries include a large number of built-in packages that assist with face detection and conduct face operations separately, resulting in reduced processing time and increased efficiency overall. The results confirm that both methods can detect faces in real time with acceptable accuracy and computation time but there are several differences. The Histogram of Oriented Gradients + Linear Support Vector Machines algorithm.method is much more preferable in terms of accuracy, but the image pyramid construction will be computationally demanding.

1. Introduction

Face detection is becoming increasingly important in creating E-commerce applications, and it is a critical link in the automatic face recognition system, which has a broad application situation than the face recognition system. Furthermore, it is extremely crucial for applications such as video identification, content-based retrieval, security systems, etc.

Basically, face detection is the technique of identifying and locating people in a given scene. The algorithms that have been presented are primarily concerned with frontal human faces. For humans, this is not a difficult assignment to complete because they are familiar with the appearance of a face because their brain has been accumulating data since childhood. However, it is a far more challenging process for machines to complete this task. The difficulty is caused by excessive facial expressions, visual fluctuations in picture and a big region to find the face in, and so forth. In addition, the retrieved facial information contains information such as the size, location, posture, expression, and so on, among other things. Face detection process, on the other hand, becomes even more complicated because of some unstable features, such as glasses and a beard, which will impact on the detection effectiveness. Furthermore, various forms and angles of lighting will cause detecting faces to generate different sorts of shininess and varied areas of shadows, which will have an impact on the performance of the face detection algorithm. The result of the detection provides face location characteristics, which may be required in many forms, such as a rectangle containing the central portion of the face, eye centers, or landmarks comprising eyes, nose, etc.

In recent years, there has been an increase in the need for automated surveillance systems. The development of real-time face detection and recognition has become a popular study topic. Optimization of system performance and speed are the key obstacles in the development of a real-time automatic face identification system since each second of live video input involves numerous activities and processing. There are numerous methods which are applied for face detection, such as, SMQT features and SNOW Classifier, Neural Network based Face Identification, and Support Vector Machine based face detection. Due to the computational resources required by these

algorithms, it is relatively simple to implement them on hardware platforms capable of dealing with large-scale computations. With the advent of the Internet of Things, there is an increasing demand for robust and efficient recognition methods that are based on single board operating systems. Therefore, the key issue is to construct facial detection systems on these single board computers, which have limited processing resources.

In this research work, the implementation of real-time face detection is conducted on Raspberry PI microprocessor. First, several previous experiments on the topic of face detection are reviewed and analyzed. Next, the explanation of choosen methodology is explained. Finally, conducted experimental results are shown in detail. The face detection algorithms are modified to take advantage of the limited processing capacity of the hardware platform. Since the speed of detection is the major factor for real-time face detection, two face detection methods are explored, and a comparative analysis is done. The first method is the implementation of the Haar Cascade Classifier based on the OpenCV library, and the second is Histogram of Oriented Gradients (HOG) + Linear Support Vector Machines (SVM) algorithm.using the Dlib library. After testing both methods on a Raspberry PI board, results show that algorithms achieve low false positives, and high true positives and real-time detection performance is obtained with a satisfactory rate of speed.

Several previous experiments on the topic of face detection are reviewed and analyzed.

Xin Zhang and Thomas Gonnot (2017) [1] from the Electrical and Computer Engineering Department of the Illinois Institute of Technology investigate the techniques for real-time face identification and recognition in complex backgrounds that are both efficient and robust. Throughout the research they explore several signal processing methods like LBP, Haar-like feature, and Principal Component Analysis. Here, a cascade classifier uses the Ada Boost algorithm to improve the detection performance of face and eye detectors. Fast face detection is made possible by the LBP descriptor, which extracts facial features. The identified facial image is subsequently processed to adjust the orientation and enhance the contrast, hence keeping the high accuracy of facial recognition. Finally, the PCA method is utilized to recognize faces efficiently. Large databases containing photos of faces and non-faces are used to train and evaluate face detection and recognition algorithms. The aformentioned methodologies are shown in the below flowchart (fig. 1):

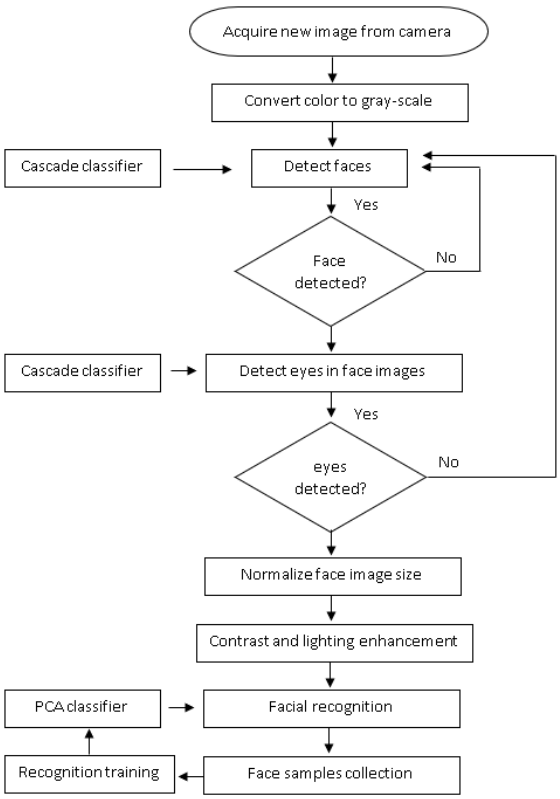

Fig. 1. Real-time face detection and recognition flowchart

Looking at the results, it is seen that the algorithms are executed on a computer with a VGA-resolution Intel Core i7-3537U processor at 2.50 GHz on a single thread. The processing time to detect each face is 11.4 milliseconds, whereas the processing time to detect each pair of eyes inside the facial areas is 15.3 milliseconds. The algorithms achieve an overall true-positive rate of 98.8% for face detection and 99.2% for correct facial recognition. The main advantage of the methods used in this research is that very high detection rate is achieved with minimum computation time.

Pablo Tribaldos and Juan Serrano-Cuerda (2013) [2] from the Albacete university has been offered a computer vision solution for recognizing people in smart spaces. Their method is applicable to both visible and infrared images. HOG is employed for feature extraction in the process of human detection, whilst for human classification linear SVM are utilized. The combination of these two methods allows for highly accurate human detection in both the visible and infrared spectrums. The HOG technique extracts feature vectors from the color images used for training in the suggested databases. Next, a linear SVM generates a hyperplane capable of optimally dividing the feature vectors into two classes. Using the recommended parameters for the feature detector and selecting a few kernels suited for

SVM in the color spectrum, the approach works well in the visible and infrared spectra, with an accuracy of 90.33 percent and a recall of 79.86 percent for automatically annotated images in the spectral region, and an accuracy of 94.64 percent and a recall of 96.91 percent for manually labeled infrared images.

After reviewing several scientific papers [3-10] on the related topic it can be said that the implementation of real-time face detection systems becomes challenging in terms of two factors: the chosen hardware platform and the accuracy and efficiency of used face detection algorithm. Taking these factors into account in this research work, two face detection methods, Haar Cascades and HOG + Liner SVM implemented in Raspberry Pi microprocessor since they are suitable for working with low computation power platforms.

2. Research Methodology

Since the whole project is carried out on Raspberry PI microprocessor, it is necessary to develop a real-time face detector, which is fast, less computation power demanding, and accurate. Taking these factors into account, two face detection methods are chosen based on other scientific researches, and here a detailed explanation of these methods will be provided.

2.1 HOG + Linear SVM algorithm

The HOG is utilized in this technique for feature extraction during the process of face detection. On the other hand, Linear SVM is utilized for the purpose of face classification. The idea behind HOG is that the intensity distribution of gradients can be used to characterize the shape of an object that is present in an image. This is the core concept behind it. A detector that is created by utilizing HOG descriptors has the significant benefit of being invariant to changes in rotation, translation, scaling, and illumination. This is a significant advantage. As a result, it is utilized effectively in photographs taken in the visible spectrum, as well as in the infrared spectrum. Following the implementation of HOG descriptors, SVMs are typically employed for classification. SVMs are a series of supervised learning methods developed for both linearly separable and non-linearly separable data. SVMs are applied to classification and regression issues in numerous domains, including text recognition, bioinformatics, and object recognition. They are also utilized effectively in the detection of individuals.

The HOG system enables the calculation of the histogram. In fact, each significant point has its own HOG feature. Each nearby area is separated into small chunks known as cells. Within each cell, a local 1-D histogram of gradient patterns is computed for each pixel. The descriptor is, therefore, the mixture of all these histograms. Vector gradient is computed in the following manner:

$$G_x(x,y) = H(x+1,y) - H(x-1,y)$$
$$G_x(x,y) = H(x,y+1) - H(x,y-1)$$

Here, Gx(x, y) represents the horizontal gradient of the image pixel, while Gy(x, y) denotes the vertical gradient. The following equations show the gradient amplitude and pixel(x, y) direction accordingly:

$$G(x,y) = \sqrt{G_x(x,y)^2 + G_y(x,y)^2}$$
$$a(x,y) = tan^{-1}(\frac{G_x(x,y)}{G_y(x,y)})$$

The given frame is subdivided into circular or rectangular shaped cells of N*N pixels. Also, gradient feature vectors are produced for each cell (fig. 2). These vectors are then combined to produce the feature vector for a particular frame. Lastly, all gradient feature vectors collected from distinct images are concatenated to form the HOG feature vector, a single long vector. The latter will serve as the input for the SVM classifier.

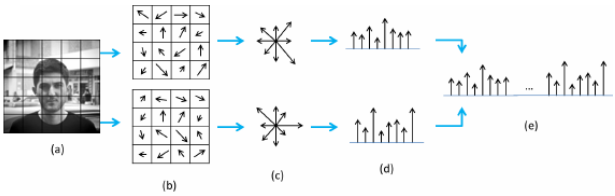

Fig. 2. HOG feature extraction

An SVM takes the features of two different objects and searches for a hyperplane that optimally separates one set of features from the other set of data. An SVM is designed to maximize the margin of separation between 2 classes, which results in a hyperplane with one side including all of the objects that belong to one class and the other side containing all of the other objects. Support vectors are the vectors that are employed in classification, and they are the vectors that are closest to the margin of separation. In the circumstance that data are not normalized, the precision of an SVM could potentially suffer as a result. Both the degree of input features and the kernel level are viable options for performing normalization. In order to complete the classification task, it is first required to split the data into phases of training and testing. In the training set, each instance has a specific value or class label, as well as a sequence of attributes, such as the observed features. These attributes are used to train the model. The purpose of SVMs is to build a model out of training data that can

estimate the key parameters of a test dataset based simply on the attributes of the objects in the test dataset.

3. Results

As discussed above two main methodologies, Haar Cascade and HOG + Linear SVM face detection algorithms will be performed on a Raspberry PI board. First, general structure of the system will be shown, and then the setup process in terms of both hardware and software will be explained and obtained face detection results will be shown. Lastly, results will be analyzed and compared for both methods using relative tables and graphs.

3.1. System design and implementation

As mentioned before, Raspberry PI is selected as a platform for the implementation of real-time face detection system. The Raspberry Pi, developed by the Raspberry Pi Foundation, is a credit card-sized, open-source, Linux-based computer board. The Raspberry Pi is an intriguing and accessible way for people of all ages to improve their computing and programming skills. The Raspberry Pi can perform many of the same tasks as a desktop computer, including internet browsing and video playback, when connected to a TV or monitor, a keyboard, and when programmed correctly. Pi is also fantastic for experimenting with unique projects; the processing capability of newer models makes them suitable for Internet of Things projects.

From a wide range of models, Raspberry PI 3 model B+ is selected for the project (fig. 3).

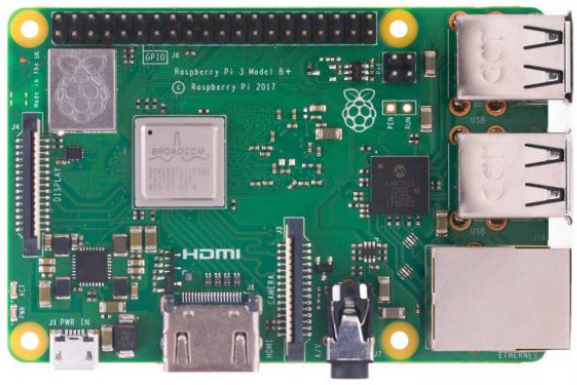

Fig 3. General view of Raspberry PI 3 model B+

This portable microprocessor is a Raspberry pi board of the third generation, which is quicker than earlier generations. It is a credit card-sized, low-cost microcomputer that connects to a computer display or television and uses a conventional keyboard and mouse. The specifications of this board are listed below:

- Broadcom BCM2837B0, Cortex-A53 64-bit SoC @ 1.4GHz processor;
- 1GB LPDDR2 SDRAM, Full size HDMI;
- 2.4GHz and 5GHz IEEE 802.11.b/g/n/ac wireless LAN and Bluetooth 4.2, BLE on board;
- 4 USB 2.0 ports, extended 40-pin GPIO header
- MIPI CSI camera port for connecting a Raspberry Camera;
- Micro SD format for loading operating system and data storage;
- 4 pole stereo output and composite video port.

In order to create a working prototype for the face detection system, there are some hardware requirements that are necessary. These components are listed below:

- Raspberry PI: Model 3B+;
- Raspberry PI Camera module v2;
- 16GB MicroSD Card;
- microSD Adapter for pc connection;
- Micro HDMI to HDMI cable;
- LCD monitor;
- Mouse and Keyboard.

The schematic view of connection of these components are represented in fig. 4.

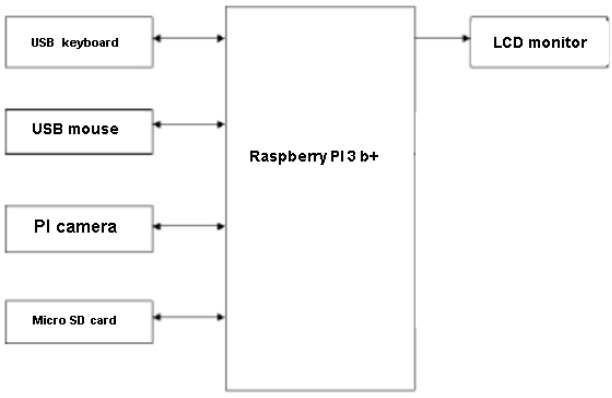

Fig. 4. Connection of hardware components for the system

As mentioned previously, for video capturing external camera module v2 is used. It has a Sony IMX219 8-megapixel sensor and can be used to capture both still images and high-definition video. It is simple to use for beginners, but offers advanced users a wealth of resources for expanding their knowledge. There are numerous examples of its use for time-lapse, slow-motion, and other video effects on the Internet.

Additionally, you can use the included libraries to create effects.

Once the setup process of hardware parts has finished, the next step is to create a working platform. Since the implementation of real-time face detection system will be build in python environment, some procedures should be completed first. The steps for obtaining a working platform in terms of software phase are listed below:

Installing the Buster OS to microSD card for Raspberry pi:

- Updating the System;
- Installing the Dependencies (imutils);
- Installing Python 3;
- Installing OpenCV and Dlib libraries.

The fig. 5 shows the components used in this project, as well as finished version of the setup configuration part.

Fig. 5. Experimental setup and hardware components

3.2. General Architecture

The following flowchart (fig. 6) shows the overall procedures of created real-time face detection system. We start by configuring the Raspberry PI camera and apply RGB to gray conversion when the camera starts recording. Next, face detection method is applied, and if face or faces are found in the detection phase then rectangular box will be drawn around the face. This process continues until the program is stopped.

3.3. Testing the camera

Before starting the video capturing, we should make sure that camera support is enabled for the Raspberry pi operating system. Camera support can be enabled from the Raspberry pi software configuration tool (fig. 7).

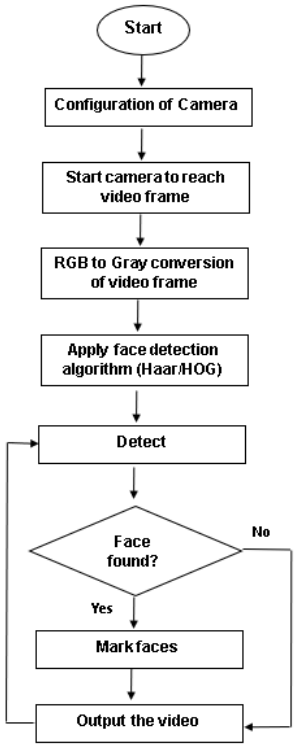

Fig. 6. The flow chart of system

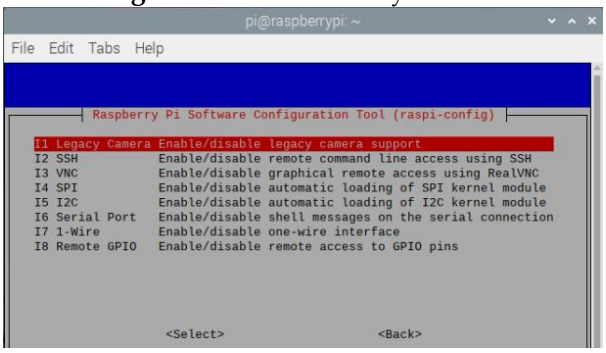

Fig. 7. Raspberry pi configuration tool

After successfully configuring the Picam and starting the video monitoring, it is necessary to convert the video frame to gray mode, since the proposed algorithms work with gray mode (fig. 8).

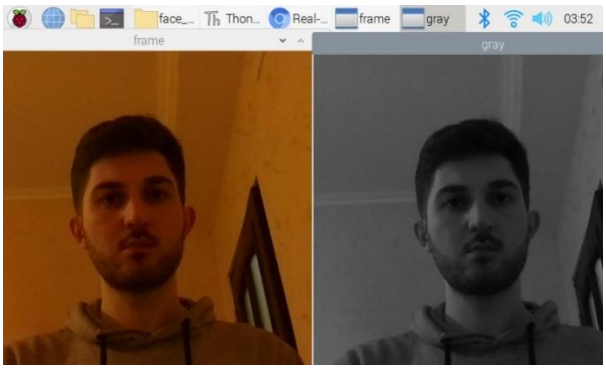

Fig. 8. Picamera video frame in RGB color and gray modes

3.4. Experimental results of Haar Cascades algorithm

Experiment is carried out by using two face detection methods. First method is Haar Cascades algorithm. In order to begin training the classifier, the algorithm requires a significant amount of both positive (positive data points demonstrate locations where there is a face) and negative (these data points are instances of regions without a face.) images. After that, we will need to derive features from it. In this step, in order to train the classifer I used OpenCV Haar training module. It contains a large number of positive and negative images which are used to generate a strong classifier for face detection system and available as XML file (haarcascade_frontalface_default.xml). After implementing this pre-trained module to our python code we can start to test the face detection system. The following figure (fig. 9) is captured from the real-time video frame during the executing of Haar face detector.

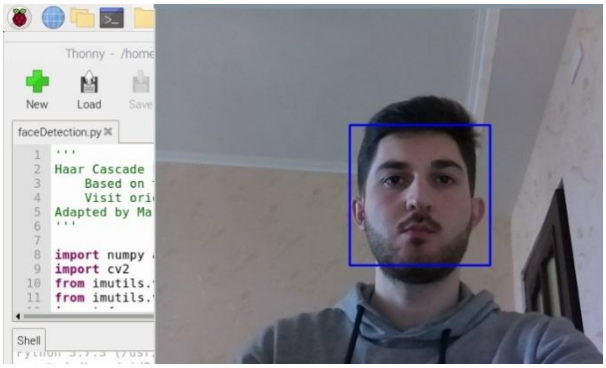

Fig. 9. Detected result of Haar face detector

It can also detect faces from a distance (up to 5 meters). Below are the examples of test results for distance face detection (fig. 10).

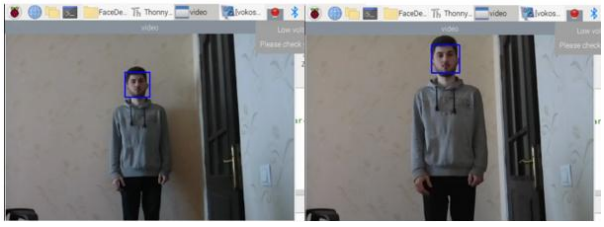

Fig. 10. Distance results of Haar face detector

One of the conclusions from the test results is that covering the eyes and mouth area partially will result in positive detection, but if one eye region is covered completely then algorithm will fail to detect face. We can see this in the following examples (fig. 11).

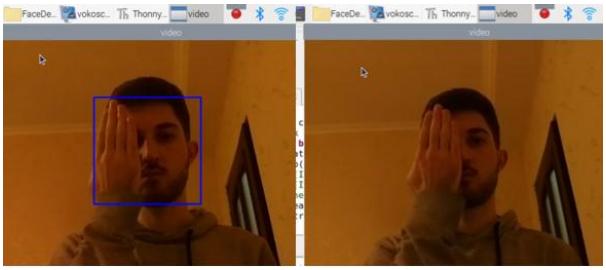

Fig. 11. Detection results when covering facial areas partially and completely

In low light conditions, depending on the visibility of the face, face detection will be either positive or negative. If most of face is visible, then system will detect the face.

Examples are shown below (Fig. 12).

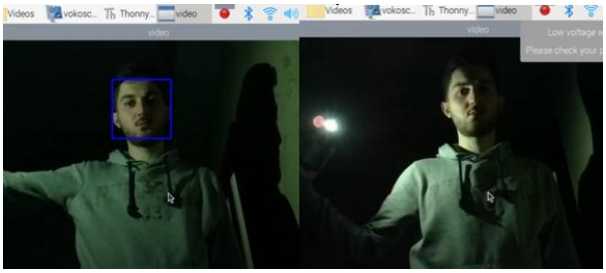

Fig. 12. Haar face detection in low light condition

The face detection system is also able detect not only one face but multiple faces in a given video frame. In the next examples we can see the test results of detection of multiple faces (fig. 13).

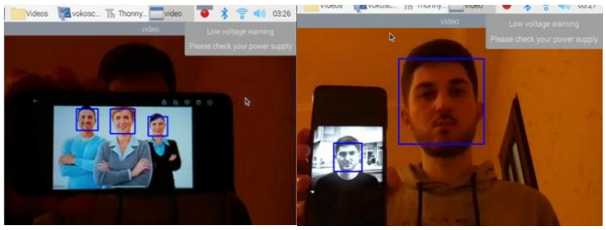

Fig. 13. Detection of multiple faces in Haar face detector

One of the downsides of the Haar face detection method is that sometimes it draws a rectangle around nonface areas. It usually happens due to the low light conditions and distance. Below we can see the examples of this situation (fig. 14).

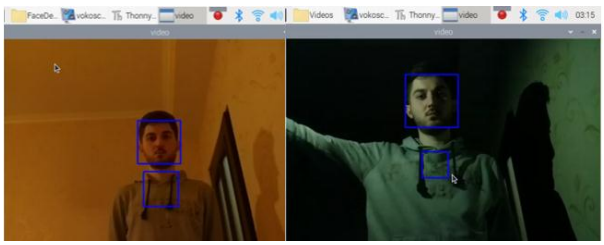

Fig. 14. False detection cases of Haar method

3.5. Experimental results of HOG + Linear SVM algorithm

The second method used in our face detection system is the implementation of HOG+ Liner SVM face detector. The detector is available in the Dlib library. Davis King initially introduced Dlib as a C++ library for machine learning. Later, however, a Python API is also created, that can be easily installed via the "pip" package manager. For the face detection dlib.get_frontal_face_detector () function is used. This function does not take any parameters and a call to it returns the dlib library's pre-trained HOG + Linear SVM face detector. Another consideration here is that after installing both OpenCV and Dlib libraries through pip package manager conversion of bounding boxes from dlib to OpenCV should be implemented, because they are represented in two libraries differently. In OpenCV, bounding boxes are represented as a 4-tuple of beginning x-coordinate, beginning y-coordinate, width, and height, while Dlib demonstrates bounding boxes through a rectangle entity with left, top, and right characteristics. After the conversion of bounding boxes, implementation of dlib.get_frontal_face_detector () function in python script is done and camera is configured for video monitoring. The system now is ready to detect faces. The following figure (fig. 15) is captured from the real-time video frame during the executing of HOG face detector.

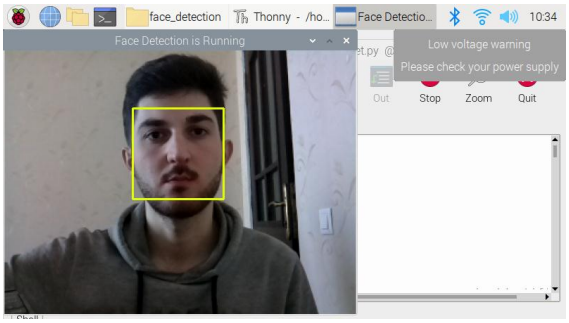

Fig. 15. Detected result of HOG face detector

As for the previous method, this face detector is also able to detect faces from a distance (fig. 16).

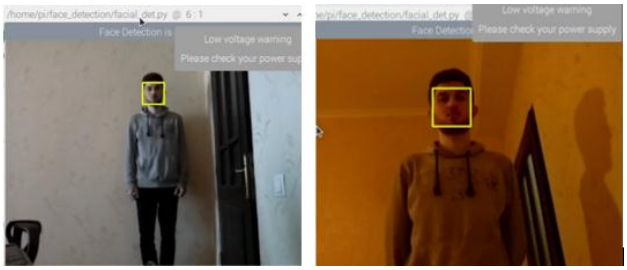

Fig. 16. Distance results of HOG face detector

In low light conditions, detection accuracy is similar to the Haar face detector. Again depending on the visibility of face region results will be either positive or negative. Test results are shown below (fig. 17).

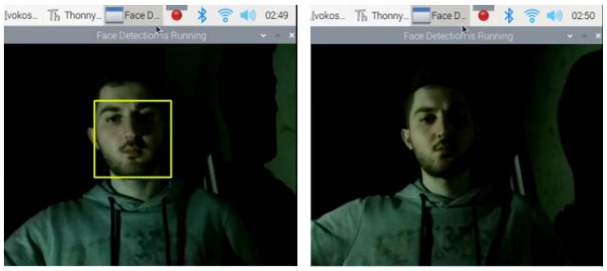

Fig. 17. HOG face detection in low light condition

Lastly we can test whether this face detector can detect multiple faces or not. Results show that algorithm is able to spot multiple faces from video frame with good detection accuracy. The test results are shown in the below figure (fig. 18).

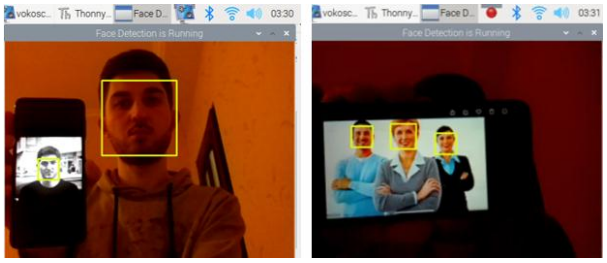

Fig. 18. Detection of multiple faces in HOG face detector

3.6 Comparison of face detection methods

After implementing both face detection algorithms, it is possible to make some conclusions regarding their performance, detection accuracy, etc. In table 1, we can see the detection accuracy, average fps count and detection speed of both methods. It can be seen that in terms of accuracy HOG face detection outperforms Haar face detector averaging 95% accuracy rate. However, in performance factors, the Haar face detection method obtains higher scores, resulting 2 times more fps count and 4 times faster face detection speed.

Table 1. Analysis of used face detection methods

Face detection method	Face detection accuracy	Average frame per second (FPS)	Face detection speed (millisecond)
Haar Cascade	~90%	15 fps	~175 ms
HOG +Linear SVM	~95%	7 fps	~740 ms

From the observed results some advantages and disadvantages of both methods can be mentioned. The pros of the Haar Cascade method are that it is extremely fast and able to run in real-time and also it has minimum requirements in terms of computational power, therefore it is possible to implement it in devices with limited resources, such as the Raspberry Pi. However, it is extremely susceptible to false-positive detections which is the main disadvantage.

In case of HOG + Linear SVM method, it can be said that higher accuracy and stability are main advantages of this face detection system. However, it just functions on frontal viewpoints of the face; it will not any profile faces because the HOG descriptor is sensitive to rotation and viewing angle changes. Also, due to image pyramid formation and sliding windows, it is actually extremely computationally demanding.

4. Conclusion

In this article, implementation of real-time face detection system was experimented on a Raspberry Pi platform. The proposed methodologies were determined after analyzing and taking the main pinpoints of previously conducted researches and experiments based on the actual topic. Two factors were essential in choosing the above discussed algorithms. One reason is that since the working hardware platform is Raspberry Pi microprocessor, the computational requirements of these platforms must be taken into account. Another reason is the detection accuracy factor of the choosen method and performance characteristics of it. In this regard, Haar Cascade and HOG +Linear SVM methods were explored to build the robust and accurate real-time face detection system. Experiments were conducted on Raspberry Pi 3 model B+ board which had a decent CPU power. During the process real-time video footage captured the faces and based on the used algorithmic method detection was carried out in certain speed and accuracy. Testing of face detection process were conducted in various conditions, including low light environments, in cases when certain areas of face were covered, distance monitoring, and also situations where multiple faces were present in a video frame.

Overall, all the experiments were successful for both methods, but there were several differences in terms of detection accuracy, detection speed and frame count per second. To explain in more detail, the HOG face detection method was the most accurate one between the two, detecting faces with up tp 95% accuracy, while the Haar face detector achieved a score of 90%. Also false positive rate was minimal for this face detector, meaning that the algorithm was much more stable than the other one. However, the Haar face detection method was superior in terms of performance measures. With average 15 frame counts per second and 175 ms detection speed, the Haar face detector outperformed the other HOG face detector 2 times and 4 times, respectively.

References

1. Xin, Zhang, Thomas, Gonnot. (2017). Real-Time Face Detection and Recognition in Complex Background. Journal of Signal and Information Processing , 8(2), 1-5.
2. Pablo, Tribaldos, Juan, Serrano-Cuerda. (2013). People Detection in Color and Infrared Video Using HOG and Linear SVM, 1-10. https://www.semanticscholar.org/paper/People-Detection-in-Color-and-Infrared-Video-Using-Tribaldos-Serrano-Cuerda/c933c4bef57be3585abb13bacb74aca29588a6ac
3. Jyotirmaya, Ijaradar, Jinjing, Xu. (2022) . A Cost-efficient Real-time Security Surveillance System Based on Facial Recognition Using Raspberry Pi and OpenCV. Current Journal of Applied Science and Technology, 41(5), 1-12.
4. Klaus, Kollreider, Hartwi, Fronthaler. (2008). Real-Time Face Detection and Motion Analysis with Application in "Liveness" Assessment. IEEE Transactions on Information Forensics and Security, 2(3), 548 – 558.
5. Yi-Qing Wang. (2014). An Analysis of the Viola-Jones Face Detection Algorithm. Image Processing On Line, 4, 128-148.
6. Daniel Hefenbrock, Jason Oberg. (2010). Accelerating Viola-Jones Face Detection to FPGA-Level using GPUs. IEEE Xplore . Conference: Field-Programmable Custom Computing Machines (FCCM), 2010 18th IEEE Annual International Symposium on 11-17.
7. Haoxiang Li, Zhe Lin. (2015). A Convolutional Neural Network Cascade for Face Detection. IEEE Explore, 1-10.
8. Li Cuimei1, Qi Zhiliang. (2017). Human face detection algorithm via Haar cascade classifier combined with three additional classifiers. International Conference on Electronic Measurement and Instruments, ICEMI, IEEE.
9. José Ignacio Rodríguez Molano. (2015). Internet of Things: A Prototype Architecture Using a Raspberry Pi. Lecture Notes in Business Information Processing, 224, 618-631.
10. Junjie Yan, Xuzong Zhang (2013). Face detection by structural models. Image and Vision Computing, Elsevier, 1-10.