Indonesian Journal of Electrical Engineering and Computer Science
Vol. 20, No. 3, December 2020, pp. 1513~1521
ISSN: 2502-4752, DOI: 10.11591/ijeecs.v20.i3.pp1513-1521

Face recognition using Viola-Jones depending on Python

Khansaa Dheyaa Ismael[1], Stanciu Irina[2]
[1]College of Information Engineering, Al-Nahrain University, Iraq
[2]Faculty of Engineering in Foreign Languages, University Politehnica of Bucharest, Romania

Article Info

Article history:

Received Jan 13, 2020
Revised Apr 5, 2020
Accepted Jun 20, 2020

Keywords:

Adaboost
Cascading
Face recognition
Haar-based features
Integral Image
Opencv
Viola-Jones object detection framework

ABSTRACT

In this paper, the proposed software system based on face recognition the proposed system can be implemented in the smart building or any VIP building need security interring in general. The human face will be recognized from a stream of pictures or video feed, this technology recognizes the person according to the specific algorithm, the algorithm that employed in this paper is the Viola–Jones object detection framework by using Python. The task of the proposed facial recognition system consists of two steps, the first one was detected the human face from live video using the webcamera in the computer, and the second step recognizes if this face allowed to enter the building or not by comparing it with the existing database, the two steps depending on the OpenCV python by importing cv2 method for detecting the human face, the frames can be read or written to file with the cv2.imread and cv2.imwrite functions respectively Finally, this proposed software system can be used to control access in smart buildings as a rule and the advancement of techniques connected around there, Providing a security system is one of the most important features must be achieved in the smart buildings, this proposed system can be used as an application in a smart building as a security system. Face recognition is one of the most important applications using today for practical facial recognition, The proposed software system, depending on using OpenCV (Open Source Computer Vision) is a popular computer vision library, in 1999 this library started by Intel. The platform library sets its focus on real-time image processing and includes patent-free implementations of the latest computer vision algorithms. OpenCV 2.3.1 now comes with a programming interface to C, C++, Python, and Android. OpenCV library of python, the three algorithms that will be used in this proposed system. The currently available algorithms are: Eigenfaces → createEigenFaceRecognizer (), Fisherfaces → createFisherFaceRecognizer (), Local Binary Patterns Histograms → createLBPHFaceRecognizer (), Finally the proposed system provide entering to the building just for the authorized person according to face recognition algorithem.

Corresponding Author:

Khansaa Dheyaa Ismael,
College of Information Engineering,
Al-Nahrain University, Baghdad 64074, Iraq.
Email: dhkhaljafaar@gmail.com, khansaa.dhia@coie-nahrain.edu.iq

1. INTRODUCTION

Our world has become like a network today, our building must be protected, especially in a very important location, there are several ways of detecting the identity of the person, such as fingerprint, eye print but the human often use faces to recognize the identity of the person. "A face recognition system is a computer application capable of identifying or verifying a person from a digital image or a video frame from a video source. One of the ways to do this is by comparing selected facial features from the image and a face database.''[1, 2]This document displays Viola–Jones object detection framework algorithm [1] using Python

to implement face recognition system, detect the object (face), Mach the object (face) with the objects in the database, and then the system decide if this person can enter into the building or not [3, 4]. The proposed software system for face recognition is implemented in two phases: the first phase detects the face from live video depending on the web camera. The proposed software system extracts the human facial features and classifies these features using Integral Image, Adaboost, Cascading, classifier. In the second-phase the system will compare the captured face from live video with a database of human face building by using OpenCV library in python [5]. Human face identity is more accurate in comparison with the existing methods. The proposed software system uses Bio ID-face-database because the human face can be used as a standard image database for the purpose of face identification and detection [6].

2. RELATED WORK

T.Deshpande and S.Ravishankar [7], their proposed methodology depends on to detect and identify the human face in the given image, using Viola–Jones object detection framework and Linear Discriminative Analysis and Artificial Neural Network to detect and identify the human face in a clustered image. Several recent studies [8], carried out the new study about the parts of the human face information analyzer, by dealing with the face as a promising model they depending on locating the facial features in images for detecting faces. They build from complex background images a fully automated human facial measurement system. Their work detects facial features such as nose, eye, and mouth is an important step for analyzing human face tasks. Their study presents a simple model approach depends on Viola–Jones object detection framework algorithm concerted with symmetric information and geometry of the face parts of the image. Several recent studies [9], carried out a new proposes which improve the approach for face detection by using Open CV, Viola–Jones algorithm based on coding eyes which removes the falsely detected faces. The Haar training module in Open CV is an implementation of the Viola–Jones framework, the input represents a training group of positive and negative images, the algorithm work to generate strong features in the format of an XML file for detecting the wanted face and eyes in a given image, to speed up Haar-like features calculation for each image the integral image is used to collect the weak classifiers and produce a strong classifier dataset and the Adaboost algorithm is implemented. By using a classifier cascade process, the speed and accuracy of the face detection system are increased. Singh1, Kaur2 [10], they carried out that human face detection is the process of detecting region of the face from a picture of one or multiple persons together. That depends on the viola-Jones algorithm to detect and process the face. The correlation model is the use of recognition face in the proposed model. The face recognition process can detect the person among the database of faces without knowing any other details about the person-specific.

3. METHODOLOGY

The proposed system presents an efficient algorithm for detecting and recognizing a human face from live video by applying the Viola–Jones algorithm [3, 11] depending on python. The flow chart for the proposed software system is as shown in Figure 1.

3.1. Inputs and outputs

The first step will get video inputs and frames in the right formats; the video feed must be transferred from the camera into the frame using the statement (camera = VideoCamera()) by taking 20 pictures for every face detection by using the function (cv2.namedWindow) to create a window that can be used as a placeholder for images and track bars.

3.2. Detecting faces from a live video feed

This step would involve using the Viola–Jones object detection framework used to provide competitive object detection rates in real-time proposed in 2001 by Paul Viola and Michael Jones [12]. To detect and capture faces from a live video feed as shown in Figure 2.

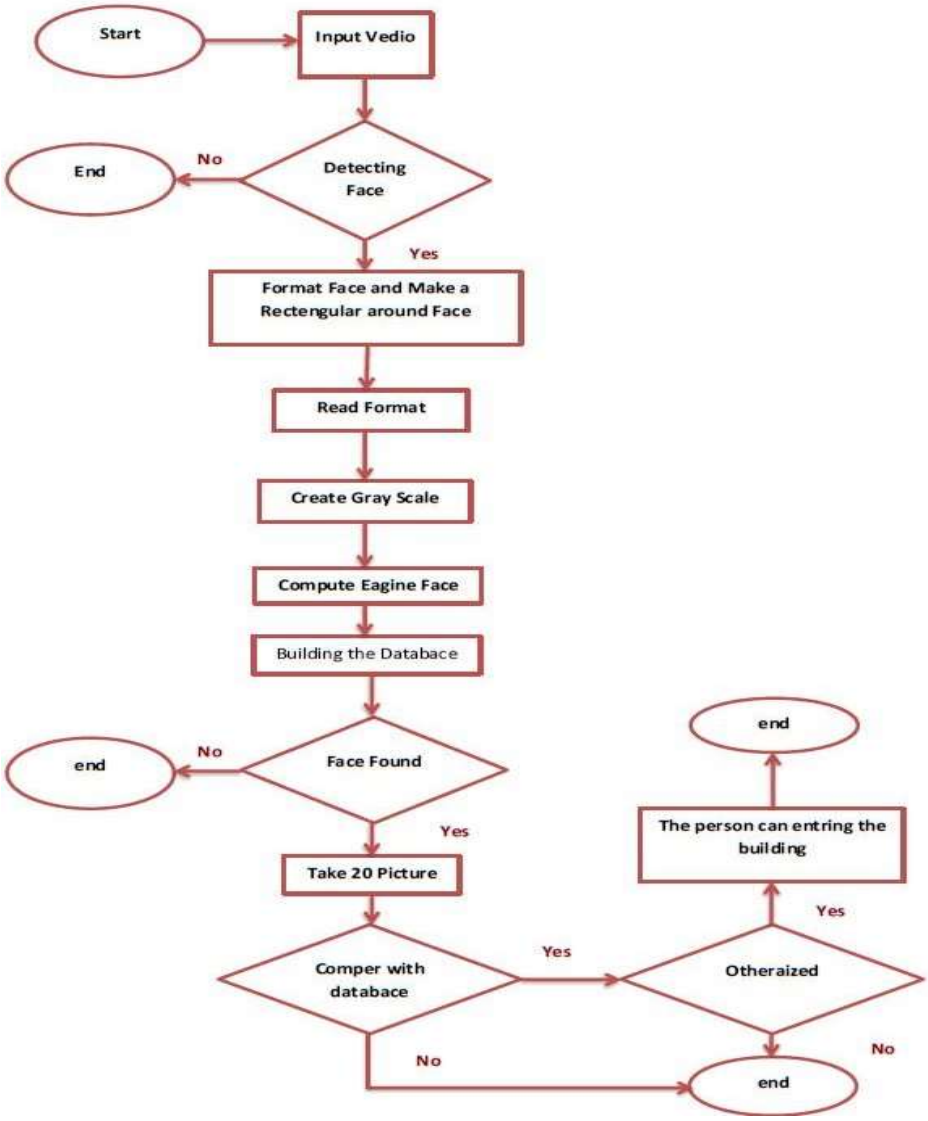

Figure 1. Face detection methodology

This implementation phase will be achieved in the processes enumerated in Figure 2:

Figure 2. Detecting face from a live video feed

3.3. Normalizing images

To get the most out of the images detected, we need to perform some operations to normalize the images from the videos [13]. First, a rectangle cuts around the face to get the most of the face while reducing background noise as shown below. The picture is converted to grayscale with the complete grayscale spectrum, [0-255], used to normalize the image attributes in terms of color, size, and intensity, the intensity is also enhanced. Finally, our pictures are taken to a particular size.

3.4. Building the database

This would involve storing images from different persons in a database. In which each image will take a name ‘’as a label’’ to recognize the authentication persons.

3.4.1. Imports

OpenCV is a library in python [14] which is an open-source vision and machine learning software. The library includes a comprehensive set of robust algorithms. For this scenario, some algorithms we'll consider include detecting faces, recognizing faces, and finding similar images from an image database. It has support for many programming languages and can run on many operating system platforms. The other libraries will be employed in the proposed software system to gate a suitable development such as ***Numpy***, ***matplotlib***, ***math***, ***IPython*** [15], and some other custom classes and functions useful for the program.

3.4.2. Video inputs and frames

The built-in camera in a normal computer system will be used to stream videos to the proposed system. The web camera is instantiated by calling the ***VideoCapture ()*** method. The quality of the video depends on the resolution (in megapixels) of the camera we are using. The video feed is displayed in an external window. To get pictures, the video is split into frames; the consecutive presentation of frames constitutes the video. Each frame in which the video is split into is a numpy array and can plot using the ***matplotlib*** library [16]. ***OpenCV***, by default, deals with videos and pictures in BGR format [17]. For consistency, we convert the videos and pictures to RGB. The Table 1. Shows that by using cv2.imread() to read the image (load from live video) in the file type .jpg, or .png, all the image file will be in RGB (red, green, blow) all the image file will read as numpy array form depending on ***OpenCV library,*** in this steep every image must be in the order of colors is BGR (blue, green, red), because the images reading by using ***OpenCV library,*** on the other hand by using cv2.imwrite() to write (save) the image in different types of files, all the image file will save in the RGB (red, green, blow).

Table 1. OpenCV read and write

Image File	→	OpenCV	→	Image File
.jpg .png, etc.	cv2.imread()	numpy array	cv2.imwrite()	.jpg .png etc
RGB	→	BGR	→	RGB

3.4.3. Reading and writing images

The proposed system can read or write the frames to file with the ***cv2.imread*** and ***cv2.imwrite*** functions respectively [18], These values below indicate the modes in which the proposed system can read the images.

➔ 1 = cv2.IMREAD_COLOR

➔ 0 = cv2.IMREAD_GRAYSCALE

➔ -1 = cv2.IMREAD_UNCHANGED

Images can also be written in different formats as desired.

3.4.4. Detecting faces

We can implement the detecting face using the following algorithms.

– Haar-based features

In Figure (3) the features of each object to give a high output when the regions are similar.

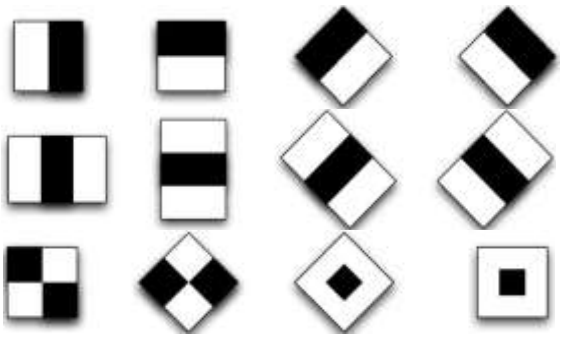

Figure 3. Edge features, line features, center surround features [19]

The output generated is given by the formula: Output = Σ (pixels in the black area) - Σ (pixels in white area) Problems arise though, because the numbers of computations that need to be carried out are very high. For instance, a 24 by 24" could give over 160,000 features and the sum of pixel intensities needs to be calculated each time any of these features are applied [6, 20].

– Integral image

This algorithm tends to solve the problem described above - reducing the number of calculations being carried out. For instance, if we have to find the integral image, S, of a square in the image below, the formula is given by:

$$S = \sum D + \sum A - \sum B - \sum C$$

As shown in figure (4) where S has four limits C, A, B, D the idea is to convert each pixel intensity by the sum of all pixel intensities to the left and above it prior to applying the Haar features. This reduces our computation to just 4 numbers for each square **Error! Reference source not found.**.

In Figure 5 the idea is to convert each pixel intensity by the sum of all pixel intensities to the left and above it prior to applying the Haar features. This reduces our computation to just 4 numbers for each square.

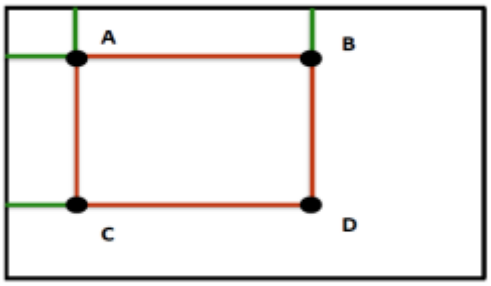

Figure 4. Finding the integral image [19]

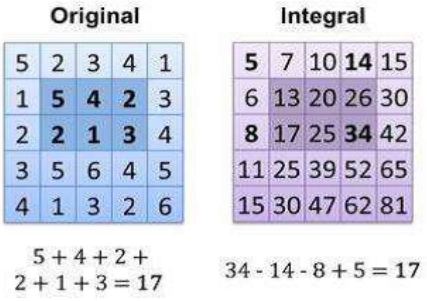

Figure 5. Integral image algorithms [21]

– Adaboost

Even after getting the integral image, the features are still quite a lot. Adaboost solves this problem by reducing the number of features. This is achieved by the formula:

$$F(x) = \alpha 1 f 1(x) + \alpha 2 f 2(x) + \alpha 3 f 3(x) \ldots + \alpha nf n(x)$$

Weak classifiers: $f n(x)$, Strong classifier: $F(x)$, and the bigger the weight α the more relevant the feature is. If the total amount of features, for instance, were 160000+, after Adaboost, the features can reduce to 6000.

– Cascading

In Figure 6 finally, cascading separate features in different classifiers and discard features that are not of interest if really sure. The proposed methodology will decide if the captured image represents a human face or not.

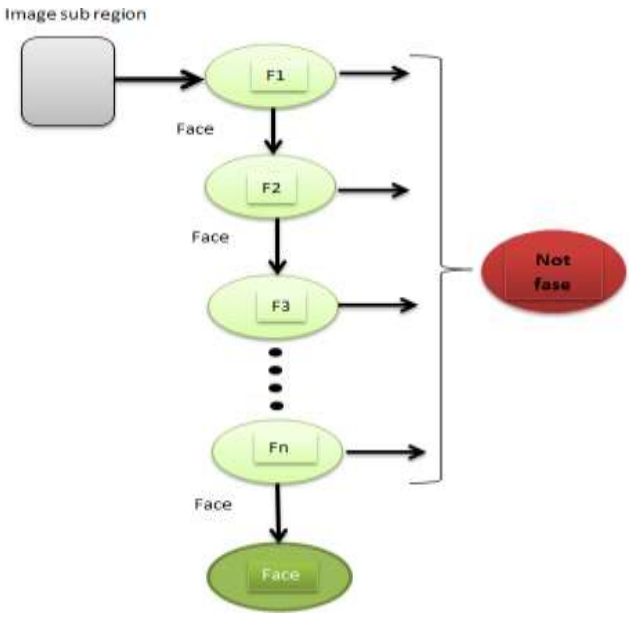

Figure 6. Cascading

There is a list of pre-trained models available for the detection of faces. For this proposed methodology, we'll be employing the *haarcascade_frontalface_alt* model. The Picture with 70% of the face and the rectangle to cut around the human face detected.

3.4.5. Normalizing images

To get the most out of the images detected, some operations must be performed to normalize the images from the videos [22]. First, a rectangle cuts around the face to get the most of the face while reducing background noise as shown in Figure 7. The picture is converted to grayscale with the complete grayscale spectrum, [0-255], used to normalize the intensity, the intensity is also enhanced, finally, the capture pictures are taken to a particular size.

3.4.6. Building the database

After detecting and normalizing the images, the images are saved in a directory. To build the database and to aid learning of the models, the proposed application takes 20 pictures for each person, which is stored in JPG format [23].

3.5. Recognizing faces from a live video feed

The final phase would detect faces from a live feed and comparing them to the images stored in the database to get a match. When our database of images is set, the proposed system train models to recognize images from a live feed [24]. To accomplish that, recognition models will be employed. For this proposed

system, the following recognition models employed - Eigen Faces, Fisher Faces, and LBPH Faces. They are instantiated by calling the functions in the ***OpenCV*** library [25]. The models are trained by accessing the pictures that have been stored in the database created. A ***Numpy*** array of images is built by the trainer and a list of labels corresponding to the saved images is also built [14]. To make a prediction using ***OpenCV*** 3.1.0, the model recognizes a picture by making a prediction and attaching the corresponding label to it. The prediction is made by getting the confidence value. The functions are described below.

Collector = cv2.face. MinDistancePredictCollector ()
Recognizer. Predict (image, collector)
Confidence = collector.getDist ()
Prediction = collector.getLabel ()

The result will present in three different numbers along with the name of the person. Each number corresponds to a function (**Eigen Faces, Fisher Faces and, LBPH Faces**)

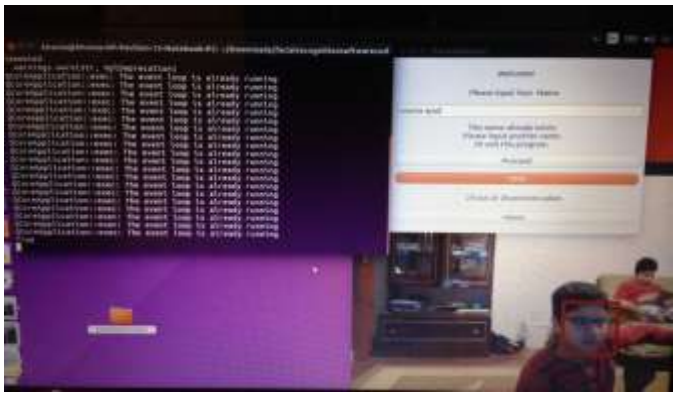

Figure 7. Detecting Faces from live video

4. RESULTS AND ANALYSIS

The results of the proposed system can be present in two-phase, the first phase will be recognizing and detecting a face from live video, and the second phase will be matching the human face with the database depending on the calculation of three functions that explain in the methodology (**Eigen Faces, Fisher Faces and, LBPH Faces**)

4.1. Recognizing and detecting a face from live video

The proposed methodology can recognize the human face and matching it with the stored database even when the person with classes, and bring the time stamp for the person even the processing for the image 64%, show in Figure 8.

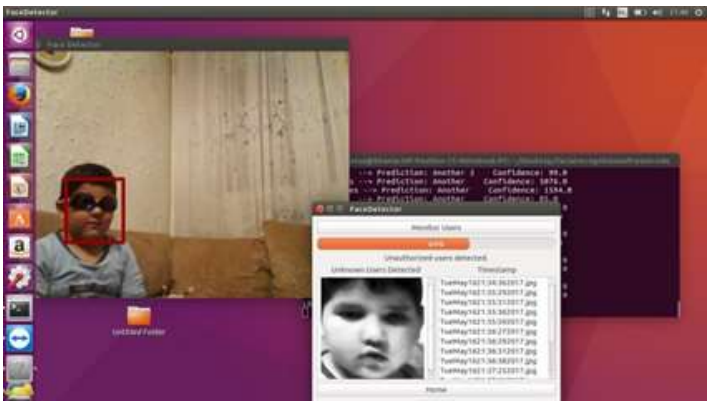

Figure 8. Recognize the human face with a class

4.2. Matching the human face with the database

The proposed methodology can recognize the face and matching it with the stored database even when the person takes a different way and different views, and bring the time stamp for the person. When the proposed system reaches to the 64% from the processing method for the new a human face that captured from the live video will bring the name of the person and the time stamp from the database if the person was interred in a previous time, if not the proposed system will make a new folder for the new person, the process is schown in Figure 9.

Figure 9. Macthicng the human face with the database

5. CONCLUSION

The proposed system has presented a type of biometric system, depending on a face recognition, which can employ in the smart building environment. The proposed software system found that the face recognition technology is the suitable prevent method in public institutions, due to the possibility of employing it in several implementations such as access control, surveillance, security system, etc., the proposed system is suitable because it is a very low consuming way of detecting and recognition face, especially our proposed system doesn't require external devices, to implement the system must have a computer with memory (RAM) at least 8 GB to ensure the fast process to the live video feed from the webcamera, if the memory (RAM) less than 8 GB the process of the system will be very slow and sometimes cannot get a good result. For future work, this proposed software system will be an application by making multi interfacing using python.

REFERENCES

[1] P. 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,* vol. 1, no. 511–518, p. 3, 2001.
[2] K. Delac and M. Grgic, *Face recognition*. 2007.
[3] M. Castrillón, O. Déniz, D. Hernández, and J. Lorenzo. "A comparison of face and facial feature detectors based on the Viola–Jones general object detection framework." *Mach. Vis. Appl.*, vol. 22, no. 3, pp. 481–494, 2011.
[4] T. S. Arulananth, M. Baskar, and R. Sateesh. "Human face detection and recognition using contour generation and matching algorithm." *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 16, no. 2, pp. 709–714, 2019.
[5] G. Bradski and A. Kaehler, *Learning OpenCV: Computer vision with the OpenCV library*. "O'Reilly Media, Inc." 2008.
[6] R. W. Frischholz and U. Dieckmann. "BiolD: a multimodal biometric identification system." *Computer (Long. Beach. Calif)*, vol. 33, no. 2, pp. 64–68, 2000.
[7] N. T. Deshpande and S. Ravishankar. "Face Detection and Recognition using Viola-Jones algorithm and Fusion of PCA and ANN." *Adv. Comput. Sci. Technol.*, vol. 10, no. 5, pp. 1173–1189, 2017.
[8] A. Gupta and R. Tiwari. "Face detection using modified Viola jones algorithm." *Int. J. Recent Res. Math. Comput. Sci. Inf. Technol.*, vol. 1, no. 2, pp. 59–66, 2015.
[9] A. M. A. Hossen, R. A. A. Ogla, and M. M. Ali. "Face Detection by Using OpenCV's Viola-Jones Algorithm based on coding eyes." *Iraqi J. Sci.*, vol. 58, no. 2A, pp. 735–745, 2017.
[10] R. Singh and M. Kaur. "Face Recognition and Detection using Viola-Jones and Cross Correlation Method." International Journal of Science and Research (IJSR), vol. 4, no. 1, pp. 2498- 2501, January 2015.
[11] R. I. Bendjillali, M. Beladgham, K. Merit, and A. Taleb-Ahmed. "Illumination-robust face recognition based on deep convolutional neural networks architectures." *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 18, no. 2, pp. 1015–1027, 2020.

[12] D. Peleshko and K. Soroka. "Research of usage of Haar-like features and AdaBoost algorithm in Viola-Jones method of object detection." in *2013 12th International Conference on the Experience of Designing and Application of CAD Systems in Microelectronics (CADSM)*, pp. 284–286, 2013.
[13] M. J. Canty, *Image analysis, classification and change detection in remote sensing: with algorithms for ENVI/IDL and Python*. Crc Press, 2014.
[14] J. Minichino and J. Howse, *Learning OpenCV 3 Computer Vision with Python*. Packt Publishing Ltd, 2015.
[15] W. McKinney, *Python for data analysis: Data wrangling with Pandas, NumPy, and IPython*. "O'Reilly Media, Inc." 2012.
[16] S. Raschka, *Python machine learning*. Packt Publishing Ltd, 2015.
[17] D. L. Baggio, *Mastering OpenCV with practical computer vision projects*. Packt Publishing Ltd, 2012.
[18] A. Mordvintsev and K. Abid. "Opencv-python tutorials documentation." *Obtenido https//media. readthedocs.Org/pdf/opencv-python-tutroals/latest/opencv-python-tutroals.pdf*, 2014.
[19] C. ennehar Bencheriet. "New face features to detect multiple faces in complex background." *Evol. Syst.*, vol. 10, no. 2, pp. 79–95, 2019.
[20] P. I. Wilson and J. Fernandez. "Facial feature detection using Haar classifiers." *J. Comput. Sci. Coll.*, vol. 21, no. 4, pp. 127–133, 2006.
[21] G. Medioni and S. B. Kang, *Emerging topics in computer vision*. Prentice Hall PTR, 2004.
[22] T. Ephraim, T. Himmelman, and K. Siddiqi. "Real-time viola-jones face detection in a web browser." in *2009 Canadian Conference on Computer and Robot Vision*, 2009, pp. 321–328.
[23] J. E. Solem, *Programming Computer Vision with Python: Tools and algorithms for analyzing images*. "O'Reilly Media, Inc." 2012.
[24] B. Vaidya, A. Patel, A. Panchal, R. Mehta, K. Mehta, and P. Vaghasiya. "Smart home automation with a unique door monitoring system for old age people using Python, OpenCV, Android and Raspberry pi." in *2017 International Conference on Intelligent Computing and Control Systems (ICICCS)*, pp. 82–86, 2017.
[25] P. Joshi, *OpenCV with Python by example*. Packt Publishing Ltd, 2015.

BIOGRAPHIES OF AUTHORS

Khansaa Dheyaa Aljafar, B.E degree in Control and Computer Engineering, Science University of Technology, Baghdad, Iraq, in the year of 2003. Master Degree - in Software System Engineering Politehnica University of Bucharest, Faculty of FILS "The Faculty of Engineering in Foreign Languages" June 2017, Bucharest (Romania) Specialization "Control Access in Smart Building, Using Face Recognition". The interested research includes Security Engineering, Embedded Systems, and communication systems, Currently working an Assistant Lec. In AL-Nahraine University, College of Information Engineering, Department of Systems Engineering, Iraq.

Irina Stanciu, Diploma in Mathematics and Computer Science University of Bucharest, Faculty of Mathematics and Computer Science, Bucharest (Romania) October 2004–June 2006 Master Degree - in Applied Mathematics Politehnica University of Bucharest, Faculty of Applied Science (2004 - 2006), Bucharest (Romania) Specialization "Models of Decision, Risk and Prognosis" October 2006–November 2011 Ph.D. in Electronics Engineering Politehnica University of Bucharest, Faculty of Electronics, Telecommunications and Information Technology, Bucharest (Romania) Thesis subject: "Evaluation of performance variations of microfluidic systems as a function of the microfabrication dispersion"