Real Time Face Detection and Recognition

Amril Prasad
Computer Science
University of Canterbury
Christchurch, New Zealand
apr107@uclive.ac.nz

Richard Green
Computer Science
University of Canterbury
Christchurch, New Zealand
Richard.green@canterbury.ac.nz

***ABSTRACT* – This paper proposes a method for real time face detection and recognition without any prior training. This Face detection technique can be used as biometrics in various applications like biometrics for attendance, biometrics for opening door. Faces are located by a Haar cascade frontal face model which is a pre trained model. The facial data is collected and stored in numpy array. Then faces are recognized by implementing K-Nearest Neighbors (KNN) which uses neighborhood classification to give a predictive value at the instance. The value of k was chosen where the recognition was most precise, keeping in mind not to choose very high k value as it may cause overfitting. With upto 3 registered users, method can achieve upto 100% recognition rate and average accuracy of one person detection to 73%. The proposed method has a method for accuracy calculation, output collection in a video file and better face detection by mapping co-ordinates as compared to previous research.**

***Keywords* – Haar Classifier, face detection, face recognition, K-Nearest neighbors(knn), numpy array**

I. INTRODUCTION

Real time face detection and recognition is one of the most researched fields. This technology can be applied in many domains like attendance in schools, offices, solving crime, biometrics at office or home and minimizing human intervention at any place, identify, finding missing persons etc.

During 1946 and 1965, Bledsoe along with Helen Chan and Charles Bisson laid the foundation of automated human face recognition system [2]. Since then, facial recognition system has seen rapid technological advancement from machine learning algorithms like Support Vector Machine to complex neural networks. Face detection and recognition are two different things. But Face recognition is built on top of Face detection. Factors like light intensity, angle and distance of face from camera are hinderance in achieving a good precision and accuracy. Though with advancements in cameras and training model over complex data with weights have resulted in good accuracy.

Biometrics nowadays plays an important role as a personal identity and recently surge for using facial data as recognition has gone up and based on Markets and Markets research centered around facial recognition market, it will be worth $6.84 billion by 2021 compared to $3.35 billion in 2016 [1]. Since one can lose it id cards and id cards maybe used by another person, problem with fingerprint is when fingers are wet or if we age fingerprint may have difficulty in recognizing, but facial data remains unaffected in many cases.

The prior research [5,6] indicates the use of KNN algorithm by Manvi Tyagi in Oct, 2019. The accuracy was not mentioned, rectangular markers on face during detection needed improvement and video feed was not saved anywhere.

This paper seeks to improve above limitations by calculating precision score manually on data given as output by feed, storing the feed by face recognizer as it is used for surveillance and defining the 3-dimensional co-ordinates which will be covering the face during detection to give more accuracy.

II. BACKGROUND

At present most of the facial recognition system follow a process that is illustrated in Figure (1). When an input live / captured image is passed, the algorithm identifies the required features and co-ordinates and extracts them. Then these are saved in a file which will be further used in recognition.

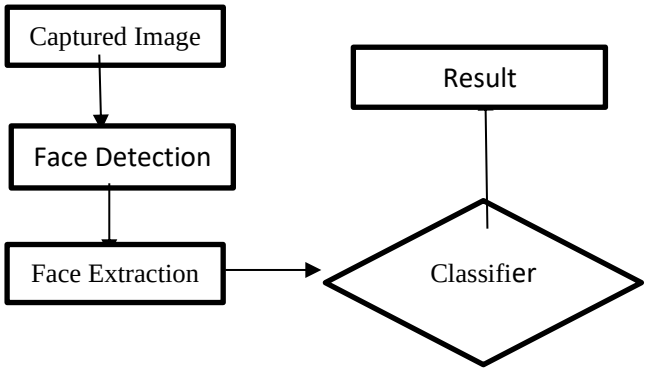


Figure (1) Simple face recognition system

A. Face Data Collection

Viola and Jones Method (Haar Cascade Frontal Face)

In 2001 Viola and Jones first described the cascade classifier for face detection in “Rapid Object detection using a Boosted Cascade of Simple Features”. It is used in finding a human face in the image or frame. The Viola and Jones algorithm will detect the human face in image / frame by calculating the Haar features [7,8,9,10]. In the algorithm three specific types of rectangles are used as shown in Figure (2).

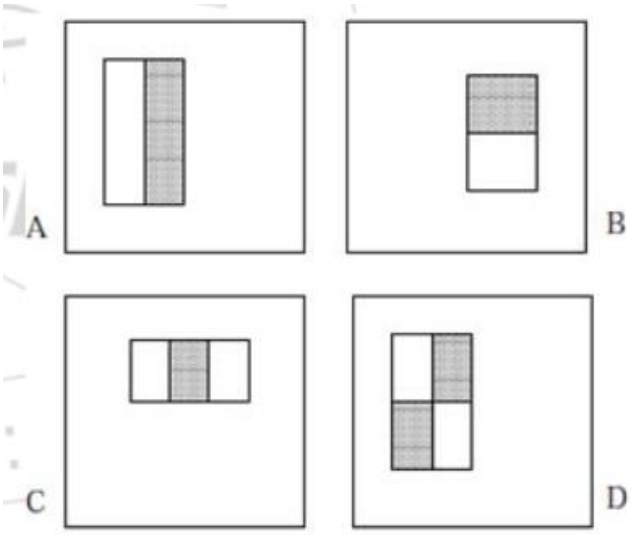


Figure (2) Specific Rectangle Types

The value of a two-rectangle as shown in A and B in Figure (2) is calculated by differencing between the sums of pixels within two regions. C as shown in Figure (2) is three rectangle feature, which is calculated by differencing the sum within the two outer regions and center region. The fourth rectangle (D) as shown in Figure (2) is calculated by differencing between diagonal pairs of regions.

Here the given regions can be of any shape or size.

After the detection of haar features, the second step is to create an integral image which is defined as the summation of the pixel values of the original image. Figure (3) illustrates the internal image, where i(x,y) is the original image.

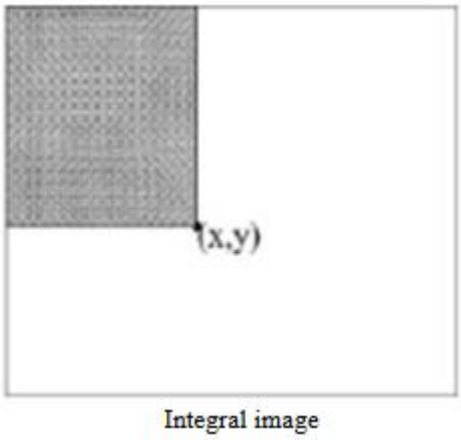


Figure (3) Integral Image

The integral image can be computed (at (x,y location) as: -

$$ii(x,y) = \sum_{x' \le x, y' \le y} i(x',y')$$

The integral image can be computed in one pass over the original image using the pair of recurrences below: -

$$R(x, y) = R(x, y-1) + I(x, y)$$

$$F(x, y) = F(x-1,y) + R(x, y)$$

Here R is initialized as R(x, -1) = 0

F is initialized as F(-1,y) = 0

The next step in the algorithm uses a variant of AdaBoost, which is a learning algorithm. It selects the best optimal features and train the classifier that use them. A strong classifier is constructed using a linear combination of weighted weak classifiers, which is defined by the following: -

$$h(x, f, p, \theta) = \begin{cases} 1 \text{ if } pf(x) < p\theta \\ 0 \text{ otherwise} \end{cases}$$

Where f is considered as the feature and θ is the threshold which is a constant obtained by AdaBoost algorithm.

The final strong classifier is defined as: -

$$C(x) = \begin{cases} 1, & \sum_{t=1}^{T} \alpha_t h_t(x) \ge \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0, & \text{otherwise} \end{cases}$$

The final of the algorithm is stage cascading. The cascade eliminates candidate if it does not pass the first stage and passes to next stage if previous stage is passed. If a candidate passes all the stages, it implies that face detection has been done. Figure (4) represents the stage cascading.

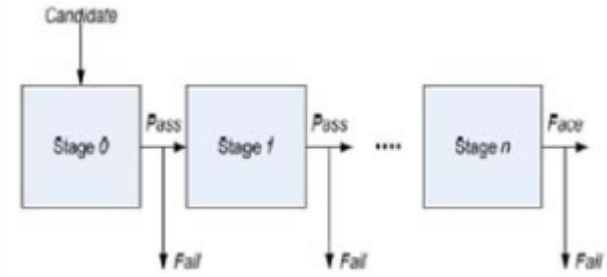


Figure (4) Stage Cascading

Though there are some limitations to this method like it can only recognise using frontal face and when there are deviations by some angle, it does not work. Additionally, it is very sensitive to lightning condition.

Numpy Array

NumPy is an open-source numerical Python library which stands for 'Numerical Python' [13, 14]. It is the fundamental package for scientific computing in Python. It provides a multidimensional array object, various derived objects such as masked arrays and matrices, and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more.
Numpy array is a powerful N-dimensional array object which is in the form of rows and columns. This ndarray object is the core of the NumPy package. We can initialize numpy arrays

from nested Python lists and access its elements. This encapsulates n-dimensional arrays of homogeneous data types, with many operations being performed in compiled code for performance. There are several important differences between NumPy arrays and the standard Python sequences, which works very well for handling image data.
NumPy arrays have a fixed size at creation, unlike Python lists which can grow dynamically. Changing the size of a ndarray will create a new array and delete the original. The elements in a NumPy array are all required to be of the same data type, and thus will be the same size in memory.
NumPy arrays facilitate advanced mathematical and other types of operations on large numbers of data. Typically, such operations are executed more efficiently and with less code than is possible using Python's built-in sequences. AS image data could be very huge, we could reply on Numpy arrays for handling image data.

When we talk particularly about image data, it is stored as three-dimensional numpy array [14].

When we look at this array, it appears in a rectangular format corresponding to image shape, though the order of the co-ordinates is reversed. Pixels in image is represented as (x,y) co-ordinates and color is specified as a RGB format. In an skimage, the co-ordinates will be reversed as (y,x) but RGB format remains the same.

Figure (5) and Figure (6) gives a visual representation of image stored in a numpy array

Figure (5) Original Image

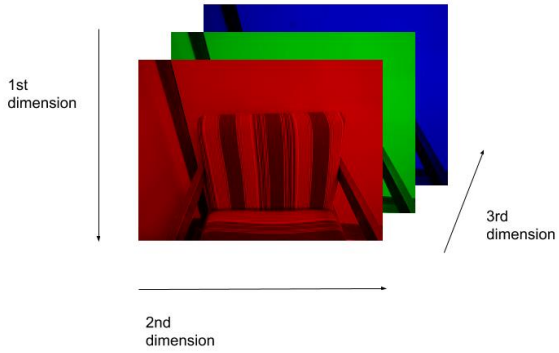


Figure (6) Image stored as numpy array

The first channel stored is the red channel, followed by the green, and then the blue.

B. Face Recognition Algorithm

K Nearest Neighbor (KNN)

The K Nearest Neighbor classifier is an extension of Nearest Classifier (NN), which is a method for classifying the objects based on closest training sample loaded in training data set. KNN is often denoted as lazy learner because it does not need any kind of training [15, 16, 17, 18]. It is an instance-based learning by combining the majority votes of neighbors it is set to. The only hyper parameter that can be tuned is k, which basically means how many nearest distant neighbors our algorithm should look for before classifying an object. Some of the measures to compute nearest distance are Euclidean distance, Manhattan distance, cosine distance.

Euclidean distance between two points (p,q)

$$d(\mathbf{p},\mathbf{q}) = d(\mathbf{q},\mathbf{p}) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \cdots + (q_n - p_n)^2}$$

$$= \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}.$$

$$d_1(\mathbf{p},\mathbf{q}) = \|\mathbf{p} - \mathbf{q}\|_1 = \sum_{i=1}^{n} |p_i - q_i|$$

Manhattan distance between two points (p,q)

$$d_{\cos}(x,y) = 1 - \frac{\vec{x} \cdot \vec{y}}{|x| . |y|}$$

Cosine distance between two points (x,y)

The Euclidean distance has been used to compute distance because of higher accuracy over other distance metrics. Though high performance of KNN algorithm is highly corelated with k value we are using. KNN algorithm uses K closest samples to the query image. Each of these samples belongs to a known class Ci. The query image Iq further categorized to the class Cm which has the majority of votes among the training samples.

The proposed workflow of facial recognition system is illustrated in Figure (7)

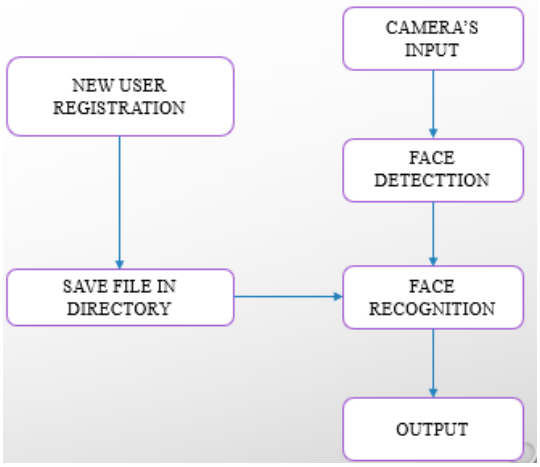


Figure (7) Face Recognition flowchart

The proposed method first detects the presence of face in a real time, maps the face co-ordinates and converts to grayscale. It also asks the name of the person whose face is being registered and finally save in the directory database in a numpy array format. Then after feeding users, face recognizer detects the face and gives the output label (name of the person it recognizes) on the output frame.

A. New User Registration and Data Handling

The algorithm allows registration for new user by taking in account of their respective name as filename. Registered user(s) can also update their data by deleting the previous saved numpy file manually to reduce ambiguity. The frames are taken continuously until 'Q' is pressed and last image is stored as I consider it the best pose a user can give. The registered users numpy file is stored in a directory which can be retrieved while classification of image in real time.

B. Face Detection

The proposed method detects face quite accurately by Haar Cascade Classifier. It identifies the face and crops it as shown in Figure (8).

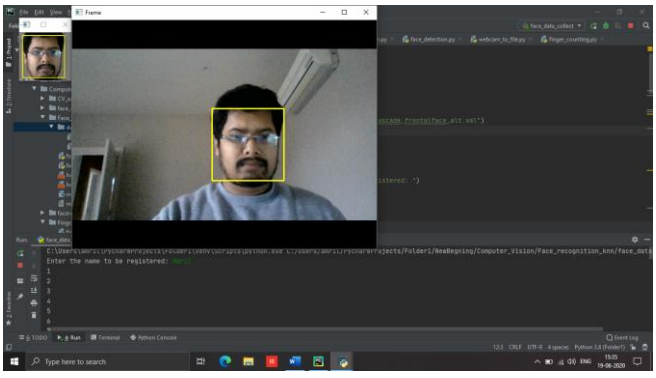


Figure (8) Face Detection using Haar classifier

After identifying and cropping the face, the result is stored as numpy array.

C. Face Recognition

Figure (9) shows the recognized face, when k (nearest neighbor), the hyperparameter was set to 5. Identification of face was proper and recognition accuracy was calculated 100%. The algorithm also gave upto 100% accuracy with two registered users. To test reliability of model, two- or three-people's result in single frame was recorded. Though there was a decline in accuracy but classification was done correct most of the times.

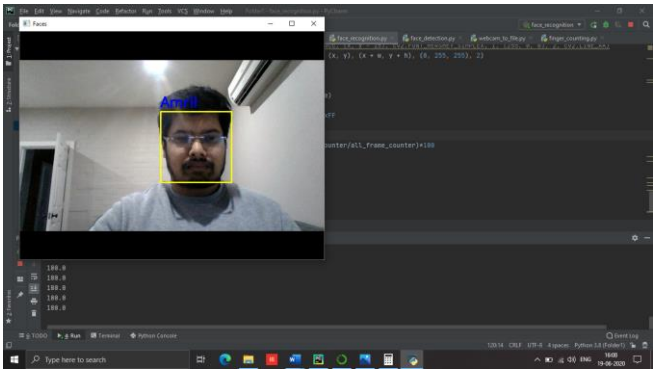


Figure (9) Face recognition with 1 person in frame

III. RESULTS

The machine and setup used to implement this proposed model is shown in Table 1.

Table 1 Environment and setup for proposed model

Equipment	Model / Version
Device	Lenovo ideapad 320
Camera	Webcam
Operating System	Windows 10
RAM	12 GB
Processor	Intel ®Core i5 7th Gen
IDE	Pycharm 2020.1.1
Language	Python 3
OpenCV	4.2.0

A. Output File

After executing the Face Recognition code, the output feed is captured frame by frame and is stored in video format (.avi) file format as shown in Figure (10). This is attained by VideoWriter function present in OpenCV [21,22]. The constructor/ function is initialized as FFMPEG or VFW on Windows. 4 character code is used to compress the frames. The proposed model has used VideoWriter::fourcc('M','J','P','G'), which is a motion-jpeg codec.

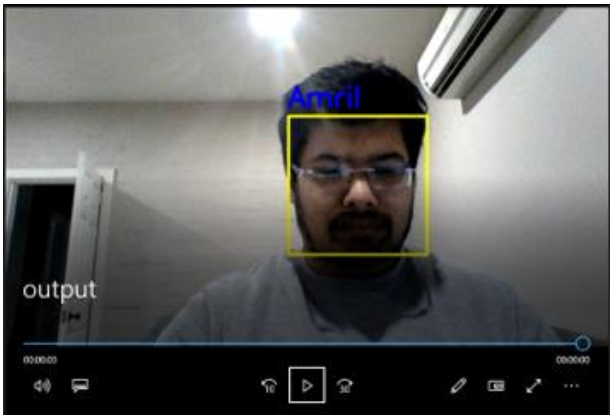


Figure (10) Output Video file

With varying the distance of face from camera, accuracy is affected and it works best with a range of 0.3 to 1.1 metres.

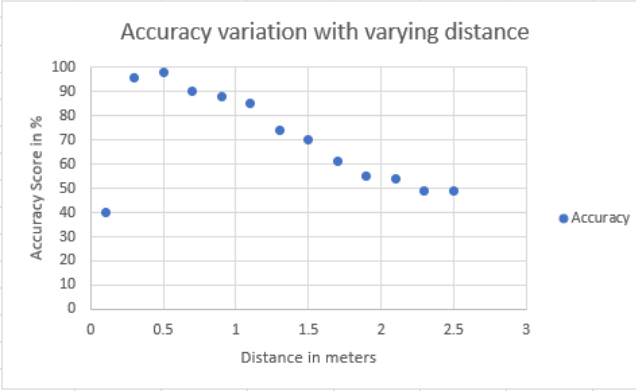


Figure (11) Accuracy score with varying distance

The closer the face is to the camera, Haar Cascade is unable to detect the face in most of the cases. Increasing the distance of face from camera by a great distance has sort of similar effect, like either model does not detect the face or it misclassify.

It was observed that face recognition model performed relatively well with different facial expression and with other things like while wearing spectacles and not wearing them. The model performed fairly with upto 3 persons in a frame, there were misclassification at some points but majorly model classified people correctly.

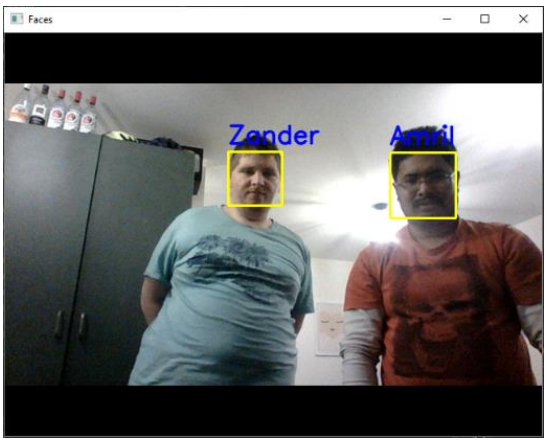


Figure (11) two people in a frame

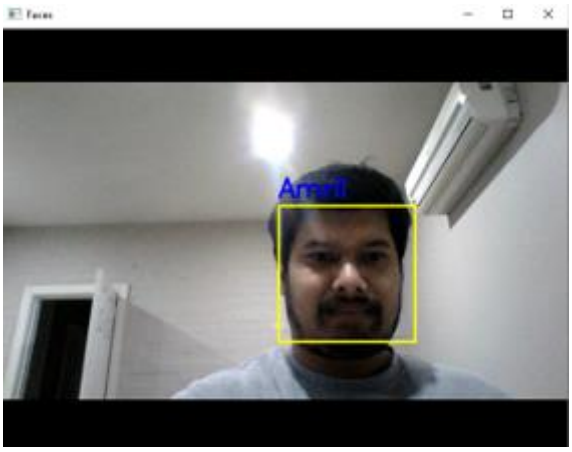
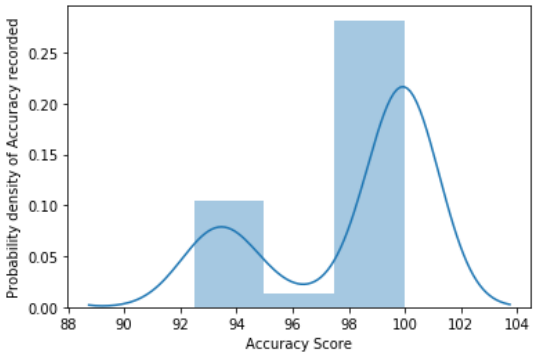


Figure (12) Without Spectacles

The accuracy for single person (Amril) while not wearing spectacles was found to 92.23% which is somewhat less than previous result.

To further determine the preciseness of proposed method, accuracy score was calculated by taking one person at a time. The accuracy of each label obtained from model was recorded and stored in a csv file. The csv file was then imported in Jupyter notebook for analysis and distplot was created with Kernel density function illustrating probability density of Accuracy Score. Figure (13) illustrates the distplot of Accuracy score of one person.

```
Text(0, 0.5, 'Probability density of Accuracy recorded')
```



Figure(13) Accuracy Measure with kde

The above plot was generated by 93 estimating points. By looking at the plot, accuracy measure tends to be in 98-100% bracket maximum times. Lowest accuracy that can be interpreted fall in 92 – 94% bracket and accuracy going that low is quite less as probability is only 0.10. The above result is generated for one person while three users were registered.

IV. CONCLUSION

A model / method to detect and recognize face in real time was implemented. This method uses Haar Cascade and KNN to detect and recognize face.

Haar Cascade and KNN together performed well in detection and recognition of face within a range of 0.3 to 1.1 metre distance. The KNN face recognition was tested and gave accuracy upto 100% for a single person in the frame. Though, the average accuracy for the person falls in 98% to

100% bracket and when there are multiple people in the frame accuracy is averaged to 69%.

This proposed method also covers the limitations of previous model by introducing enhanced co-ordinates for detecting face, a method for accuracy calculation and storing the output in a video file.

A. Future Research

The proposed method works fairly but certain areas needs improvement. The thirst for newer and better technology is unquenchable. Day in and day out, new neural networks are released in the market which have some improvements as compared to the old ones. In real time scenario, model should be able to work properly with huge dataset and reliably. The training dataset should be sufficient in order for the model to classify between a huge number of people.

In order to achieve the above, one would require a better environment setup, a GPU preferably. Siamese neural network is one of a kind which has gained popularity because of its one-shot learning power. There is literally a universe to explore when we step in the realm of deep learning with neural networks.

So, future work will involve training the model with Siamese Neural Network, maintain the data in a cloud environment, adding new features, scenarios and above all working on GPU which is a necessity for all this to achieve powerful computation power and better accuracy.

V. REFERENCES

[1] Kairos, "Face Recognition for Security," [Online]. Available: https://www.kairos.com/face-recognition-for-security.

[2] M. Rovai, "Real-Time Face Recognition: An End-To-End Project," towardsdatascience, [Online]. Available: https://towardsdatascience.com/real-time-face-recognition-an-end-to-end-project-b738bb0f7348. [Accessed 27 April 2020].

[3] S. Raviv, "The Secret History of Facial Recognition," Wired, [Online]. Available: https://www.wired.com/story/secret-history-facial-recognition/. [Accessed 26 April 2020].

[4] R. K. T, *Enhanced Visual Attendance System by Face Recognition uisng K-Nearest Neighbor Algorithm,* Journal of Advanced Research in Dynamical and Control Systems, 2019.

[5] M. Tyagi, "Face Recognition Using Knn & OpenCV," Medium, [Online]. Available: https://medium.com/analytics-vidhya/face-recognition-using-knn-open-cv-9376e7517c9f. [Accessed 1 May 2020].

[6] M. tyagi, Github, [Online]. Available: https://github.com/Manvityagi/Face-Recognition-using-KNN-openCV. [Accessed 1 May 2020].

[7] P. Viola and M. J. Jones, *Robust Real-Time Face Detection,* 2 ed., vol. 57, International Journal of Computer Vision, 2004, pp. 137-154.

[8] D. M, A. A and D. O, *Face Detection Using Viola and Jones Method and Neural Networks,* Abu Dhabi: Information and Communication Technology Research (ICTRC), 2015.

[9] M. K. Dabhi and B. K. Pancholi, *Face Detection System Based on Viola - Jones Algorithm,* vol. 5, Baroda: International Journal of Science and Research (IJSR), 2016.

[10] A. Yap and R. Green, *Real Time Face Detection and Recognition,* Christchurch: Department of Computer Science and Software Engineering, University of Canterbury, 2017.

[11] V. Gupta and D. Sharma, *A Study of Various Face Detection Methods,* 5 ed., vol. 3, International Journal of Advanced Research in Computer and Communication Engineering, 2014, pp. 6694-6697.

[12] "Image Processing with Python," datacarpentry, [Online]. Available: https://datacarpentry.org/image-processing/aio/index.html. [Accessed 17 May 2020].

[13] "What is NumPy?," [Online]. Available: https://docs.scipy.org/doc/numpy-1.13.0/user/whatisnumpy.html. [Accessed 17 May 2020].

[14] N. Community, "Numpy User Guide," [Online]. Available: https://numpy.org/doc/1.18/numpy-user.pdf. [Accessed 18 May 2020].

[15] H. Ebrahimpour and A. Kouzani, *FACE RECOGNITION USING BAGGING KNN.*

[16] D. and M. Kaur, *K-Nearest Neighbor Classification Approach for Face and Fingerprint at Feature Level Fusion,* 14 ed., vol. 60, Chandigarh, Punjab: International Journal of Computer Applications , 2012.

[17] E. Setiawan and A. Muttaqin, *Implementation of K-Nearest Neighbors Face Recognition on Low-power Processor,* 3 ed., vol. 13, TELKOMNIKA, 2015.

[18] R. Liao and R. Green, *Face Recognition with CNN,* Department of Computer Science and Software, University of Canterbury, Christchurch, 2019.

[19] H. Wu, . Z. Xu, . J. Zhang, . W. Yan and . X. Ma, *Face recognition based on convolution siamese networks,* Shanghai, 2018.

[20] G. Koch, R. Zemel and R. Salakhutdinov, *Siamese Neural Networks for One-shot Image Recognition,* Department of Computer Science, University of Toronto. Toronto, Ontario, Canada.

[21] "cv::VideoWriter Class Reference," [Online]. Available: https://docs.opencv.org/3.4.8/dd/d9e/classcv_1_1VideoWriter.html. [Accessed 12 June 2020].

[22] A. Rosebrock, "Writing to video with OpenCV," [Online]. Available: https://www.pyimagesearch.com/2016/02/22/writing-to-video-with-opencv/. [Accessed 12 June 2020].