Haar Classifier Based Face Recognition and Tracking in a Video Stream using Real Time Computer Vision with OpenCV

Samudith Nanayakkara
Information Technology
General Sir John Kothelawala Defence University
Colombo, Sri Lanka
samudithnanayakkara97@gmail.com

Ashen Wanniarachchi
Information Technology
General Sir John Kothelawala Defence University
Colombo, Sri Lanka
ashenw@kdu.ac.lk

***Abstract*—In the past few decades, the interest on computer vision has increased drastically with the development and popularity of machine learning and deep learning concepts. Human face recognition and detection from an image or a video stream source is a popular research topic in the field of biometrics. Face detection and recognition technology have captured a greater attention due to the successful implementation in various applications and real-world consumer related market products. With the development in technology and hardware capabilities, face detection/recognition and emotional recognition has become a popular research area related to computer vision and algorithm-based image analysis. The paper further emphasizes a real-world implementation using a program which is developed using various libraries of Open CV, Matplotlib and Python framework along with video stream using a High Definition input source. A critical analysis is provided on Haar-Classifier which is an algorithm used in OpenCV for face detection and tracking purposes.**

Keywords— face detection, Haar-Classifier, OpenCV

I. INTRODUCTION

Machine learning builds under the background of Artificial intelligence that could transform information to knowledge. In the past few years people used to explore more and more data, but it would be useless if it cannot be analysed and find the pattern hidden in it. Machine Learning (ML) techniques has the potential to figure out the underlying pattern in a large/complex dataset which would be extremely difficult to discover by humans. Once the hidden pattern is identified and with knowledge on specific problem, predictions could be provided on events in the future that might be helpful in future decision making. [1]

Computer vision is a rapidly evolving research area devoted to modifying, analysing and in-depth high-level understanding of individual or series of vector and raster images. Computer vision is mainly intended to determine the feed from an external image or video source and use that understanding to manipulate and create a response on a programme or trigger an event.[2] Computer vision technology are used in many different applications in the current world such as on biometrics, surveillance systems, augmented reality, user authentication systems and many more. The main goal of the paper is to emphasize the implementation of face detection and recognition using OpenCV libraries with the aid of a visual stream input from a web camera or any other input source. Once the program is initiated based on the input stream the algorithm will search, detect, and recognize any human face/group of faces based on various facial features on a face of a typical human being.[3]

II. CHALLENGES AND COMPUTER VISION APPLICATION

Computer vision is highly computational oriented, since most computer vision application must be executed in real time, it requires lot of computational and processing power. At minimum level, a single frame should be processed and completed within 35-45 milliseconds. Modern photography is embedded with computer vision by which cameras are capable of automatically detect and focus a person face and automatically trigger the shutter when a person smile. With rapid revolution in the camera technology, modern photography cameras, CCTV cameras and smart phone cameras are embedded with powerful lens, aperture, shutter capacity and low light sensors. These developments have improved the computing power on image processing which ultimately increases the demand on computer vision applications. Optical Character Recognition uses computer vision technology to identify the text and characters from a scanned document and which they can read them aloud using an audio synthesizer. Modern automobile industry uses computer vision to maintain the vehicle in the road lanes which is used with cruise control system, here the driver can allow the vehicle to travel with in a specific lane. Robotics at the manufacturing industry depends on computer vision to map real world objects and its parameters when performing specified tasks accordingly. [4]

III. OPEN CV

OpenCV (Open Source Computer Vision Library) is a computer vision library available on open source platform.[5] This is developed using C language and could be implemented on any kernel platform. Open CV is developed by Intel Corporation. The libraries are mainly developed to be used in real time processing of images. Originally the OpenCV libraries are developed using C which makes it portable to some platforms. OpenCV can be implemented using many different languages such as python, Java etc. With the introduction of OpenCV 2.0 a new C++ interface is embedded with the traditional C interface. With the introduction of new version and new interface, vision functionalities could be implemented in an application with

ISSN 2756-9160 / November 2020.
International Conference on Advances in Computing and Technology (ICACT–2020) Proceedings

few numbers of code lines and reduces the programming errors such as memory leaks.[6]

IV. LITERATURE REVIEW

A. Face Detection Classifiers in OpenCV

Various algorithms or classifiers are used to perform the face detection in OpenCV. The algorithms/classifiers are used to determine whether the input stream consist of -face (Positive)/Consist of a face or face (Negative)/Does not contain face. OpenCV consist of two pretrained classifiers which could be used in face detection and recognition applications.[7] The two type of classifiers are-

- Haar Classifier
- LBP Classifier or Local Binary Pattern

On 2002 with the paper titled "Multiresolution Grayscale and Rotation Invariant Texture Classification with Local Binary Patterns" Local Binary Pattern (LBPs) was re-introduce and populated.[8] Just like Haar classifier, LBP classifier is trained using several images. Human face consists of micro visual patterns where LCP visual descriptor features used feature vectors, which recognize a human face from a non-face object.[9] Since the current facial detection and recognition application is developed using Open CV Haar Classifier, it further elaborates the algorithm using the Haar feature based cascade classifier.

B. Haar feature based cascade Classifier

Haar name originated from the Haar wavelets which is used by earlier real time face detectors. Haar-feature classifier is based on the Haar wavelets which was first used by Paul Viola and Michael Jones in their paper titled "Rapid Object Detection using a Boosted Cascade of Simple Features". The classifier was mainly used in computer vision applications such as for face detection.[10]

Haar classifier utilizes a machine learning approach for object detection in visual input streams and which can perform image processing at high speeds with higher detection rates. A large amount of positive and negative datasets of images are used to train the classifier. Positive images are datasets of images which the classifier needs to detect. While the Negative images are datasets of various other images of objects which the classifier does not want to detect.[10]

This can be attributed to three main reasons:

Haar classifier uses "Integral Image" concept to quickly compute the features detected by the detector. This concept reduces the image processing time which is important in computer vision related applications.

The Algorithm used in learning is based on "AdaBoost", this selects limited number of most important features from a large data set and provides the most efficient classifier.

Many more classifiers are combined to create a "cascade" to avoid the focus on non-face regions in a video stream/image, where more computation is focus or spent on object like regions or areas.

C. Methodology of the Haar feature based cascade Classifier

1) Haar Classifier features extraction

A large amount of training data as video stream or images are fed to the algorithm. Then the classifier begins to extract the Haar related features from each and every fed image or inputted video stream. After words Haar features are used to primary detect whether a relevant feature is present in the input video feed or the image. Haar features are similar to square shaped windows which are placed on images or run across the video stream to compute a feature. The feature is a single value obtained by subtracting the total pixels under the white area and that under the black area.[11]

2) Integral Images algorithm

The Integral Images algorithm was introduced by Viola Jones. The algorithm uses a "24x24" base rectangle shaped window, which can calculate over 180,000 features. This algorithm uses the values of four corners of the rectangle to calculate the total number of pixels under it.[10]

3) AdaBoost

As elaborated above, even though by 24x24 window more than 180,000 values of features could be resulted, all the features might not be helpful on detecting a human face.(Goyal et al., 2017) To determine the best feature out of the bulk, the Ada boost algorithm would be used. AdaBoost algorithm is used to filter the features which are helpful to increase the accuracy of the classifier. After the filtering process the number of features would gradually drop from 180,000 to 6000.[10]

4) Cascade of Classifiers

Cascade of Classifiers is another way proposed by Viola Jones, that would contribute the algorithm to process faster. Cascade Classifier consist of various stages where at each stage it consists of a strong classifier. A major benefit of the cascade classifier is it eliminates the requirement of applying all the features at once in a window. Separate sub window groups of features are created, and at each stage classifier determines whether sub window detect a face or not. In the absence of a face the sub window is discarded with the respective features of the window. If the classifier is passed by the sub-window, then at the next stage the second stage of features are applied.[10]

V. TECHNOLOGIES UTILIZED

The program is developed using the Python programming language along with Matplotlib 2.0 and OpenCV 3.2.0 dependencies. The program is developed and run using Python 3.6.8 64bit version. Since this version is more stable when executing OpenCV projects. The cascade files of Haar algorithm could be downloaded from OpenCV GitHub page. The video stream is sourced to the program using HP True Vision HD inbuild Web camera.[12]

ISSN 2756-9160 / November 2020.
International Conference on Advances in Computing and Technology (ICACT–2020) Proceedings

VI. IMPLEMENTATION

The program is mainly developed using the Haar Classifier function which is capable of analysing and detecting human face or group of faces using the HD video stream input from the laptop web camera. After successfully detecting the face/group of faces, program outputs a green colour square around the face/ group of faces by tracking the position of the face in the video stream.[10] Initially after installing the OpenCV 3.2.0, Python v3.6.8 and Matplotlib 2.0, the required libraries/dependencies should be imported and added to the program module. [13] The list of required imports is as follow- **import numpy as np ,import cv2, import matplotlib.pyplot as plt, %matplotlib inline**

The program should be fed with a video source to analyse and perform the face detection. The following code is used to load the live video stream from the web camera of the laptop to the program-**camera=cv2.VideoCapture(0).** After adding the input source, the Haar Cascade files should be imported. OpenCV contains number of pre trained classifiers for various instances. Some of the classifiers could be used to detect the eyes, face, object etc. OpenCV classifier file is an XML file. The xml files are added to the root of the project directory. The classifier xml file is as follow-**clsfr=cv2.CascadeClassifier('haarcascade_frontalface_default.xml')**

The detection algorithm only works with grayscale colour scheme. The following line of code is added to transform the colour images/video stream to grayscale. **gray=cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)**

To implement the Face detection, in the classifier a module name "detectMultiscale" must be added to the code. When the above module is added the function will return a square with coordinates(x,y,w,h) around the face detected. With the function three important parameters are passed based on the data. The three parameters passed are image or input video source of type CV_8U, "scaleFactor" and "minNeighobors". "ScaleFactor" is passed to determine the amount of reduction or compression in input image or video source under each scale."minNeighbors" parameter is passed to specify the amount of neighbors each and every candidate rectangle should have to retain and depict the face. The quality of the faces detected will be affected with this parameter. The following line of code is added to implement the above functionality-**faces=clsfr.detectMultiScale(gray)**

An infinite loop is added to the code since the input stream is sourced from the web camera which is a video stream. The loop runs over all the coordinates returned and these coordinates are represented using a square in OpenCV. A green square will be drawn around the face if detected with a width of 2CM. The code used to add the loop and green rectangle functionality to the program-**for (x,y,w,h) in faces: cv2.rectangle(img,(x,y),(x+w,y+h),(0,255,0),2)**

With the green square representation, a text called "Face" will be outputted in HERSHEY font type. The following line of code is used to add this functionality to the program. **cv2.putText(img,'FACE',(x,y-10),cv2.FONT_HERSHEY_SIMPLEX,1,(0,255,0),2)**

Finally, in order to determine whether face/group of faces has been correctly detected, the original inputted video stream is outputted in colour format. The following line of code is used to add this functionality to the program-**cv2.imshow('LIVE',img)**

VII. RESULT AND DISCUSSION

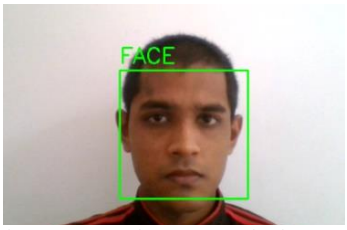

Fig. 1. Output of the program after a successful face detection

The final outcomes of the programme are referred using the Fig 1. These HD video frames are sourced from a web camera. Occasionally face detection algorithm would detect another face within the face detection rectangle, although there is only one face physically existed. In the said cases post image processing is used to derive the coordinates of the exact face using OpenCV Haar Classifier. When system falsely output more than 1 square to a face, distance of centre points of the square is been calculated. If the calculated distance is smaller than the pre-set threshold, Then the final position of the detected face will be determined by calculating the average of these squares. Although the currently implemented method is successful, various bugs and limitations have been identified when testing the face detection algorithm due to the performance limitation in Python language. After evaluating the program based on factors such as Individual face/group of faces detection capability, Detection speed, Efficiency of detecting under various lighting conditions, CPU and resource utilization while the program is running, the face detection algorithm proposed by Viola and Jones is more suitable when implementing real time face recognition/detection.

VIII. CONCLUSION

Face detection/tracking and recognition is important on building commercial and industrial applications. The paper presents an in-depth explanation of Haar Classifier algorithm implemented using OpenCV libraries. Various face recognitions algorithms could be utilized based on various demands and factors affecting to the application. Advantages of implementing face detection using Haar classifier features over other features are high calculation speed. Accuracy and speed are the features which determine the success of a face detection algorithm. To provide a better face detection and recognition, the program will be further enhanced in the future to implement the solution in a practical real-world application using Internet of Things.

REFERENCES

[1] Machine Learning—Fundamentals. Basic theory underlying the field of… | by Javaid Nabi | Towards Data Science. (n.d.). Retrieved

July 29, 2020, from https://towardsdatascience.com/machine-learning-basics-part-1-a36d38c7916

[2] Emami, S., & Suciu, V. P. (2012). Facial Recognition using OpenCV. Journal of Mobile, Embedded and Distributed Systems, 4(1), 38–43.

[3] Pulli, B. K., & Baksheev, A. (2012). OpenCV_CACM_p61-pulli. 61–69.

[4] Computer Vision Applications in 10 Industries—Algorithm-X Lab. (n.d.). Retrieved July 29, 2020, from https://algorithmxlab.com/blog/computer-vision/

[5] About. (n.d.). Retrieved July 29, 2020, from https://opencv.org/about/

[6] OpenCV. (n.d.). Retrieved July 29, 2020, from https://opencv.org/

[7] Python | Haar Cascades for Object Detection—GeeksforGeeks. (n.d.). Retrieved July 29, 2020, from https://www.geeksforgeeks.org/python-haar-cascades-for-object-detection/

[8] Face detection using OpenCV and Python: A beginner's guide—Blogs SuperDataScience—Big Data | Analytics Careers | Mentors | Success. (n.d.). Retrieved July 28, 2020, from https://www.superdatascience.com/blogs/opencv-face-detection

[9] Local Binary Patterns with Python & OpenCV - PyImageSearch. (n.d.). Retrieved July 29, 2020, from https://www.pyimagesearch.com/2015/12/07/local-binary-patterns-with-python-opencv/

[10] Face Detection with Python using OpenCV - DataCamp. (n.d.). Retrieved July 29, 2020, from https://www.datacamp.com/community/tutorials/face-detection-python-opencv

[11] Goyal, K., Agarwal, K., & Kumar, R. (2017). Face detection and tracking: Using OpenCV. Proceedings of the International Conference on Electronics, Communication and Aerospace Technology, ICECA 2017, 2017-Janua(4), 474–478. https://doi.org/10.1109/ICECA.2017.8203730

[12] Review #1308 about "HP Truevision HD" | Webcam Reviews | Webcam Test. (n.d.). Retrieved July 29, 2020, from https://webcamtests.com/reviews/1308

[13] Face Detection in 2 Minutes using OpenCV & Python | by Adarsh Menon | Towards Data Science. (n.d.). Retrieved July 29, 2020, from https://towardsdatascience.com/face-detection-in-2-minutes-using-opencv-python-90f89d7c0f81