Real Time Face Detection and Tracking Using OpenCV

Prof. P Y Kumbhar[1], Mohammad Attaullah[2], Shubham Dhere[3], Shivkumar Hipparagi[4]

[1]Faculty, Department of Electronics and Telecommunication Engineering, Walchand Institute of Technology, Solapur, Maharashtra, INDIA. E-Mail: pravinkumbhar03@gmail.com
[2]Student, Department of Electronics and Telecommunication Engineering, Walchand Institute of Technology, Solapur, Maharashtra, INDIA. E-Mail: mohammadattaullah13@gmail.com
[3]Student, Department of Electronics and Telecommunication Engineering, Walchand Institute of Technology, Solapur, Maharashtra, INDIA. E-Mail: shubhamdhere007@gmail.com
[4]Student, Department of Electronics and Telecommunication Engineering, Walchand Institute of Technology, Solapur, Maharashtra, INDIA. E-Mail: shivahipparagi08@gmail.com

ABSTRACT

In this paper, we intend to Implement a real-time Face detection and tracking the head poses position from high definition video using Haar Classifier through Raspberry Pi BCM2835 CPU processor which is a combination of SoC with GPU based Architecture. SimpleCV and OpenCV libraries are used for face detection and tracking the head poses position. The experimental result computed by using computer vision SimpleCV and OpenCV framework libraries along with above mentioned hardware results were obtained through of 30 fps under 1080p resolutions for higher accuracy and speediness for face detection and tracking the head poses position.

Keywords—Raspberry Pi; Haar Filter; OpenCV; Processing; Servos.

1. INTRODUCTION

Face detection is a computer technology that determines the locations and sizes of human faces in arbitrary (digital) images. It detects facial features and ignores anything else, such as buildings, trees and bodies. Human face perception is currently an active research area in the computer vision community. Human face localization and detection is often the first step in applications such as video surveillance, human computer interface, face recognition and image database management. Locating and tracking human faces is a prerequisite for face recognition and/or facial expressions analysis, although it is often assumed that a normalized face image is available.In this paper we intend to implement the Haar-Classifier for Face detection and tracking based on the HaarFeatures.

2. RELATED WORKS

Robust and real-time face detection plays a vital role in many of the application scenarios like in biometrics, often as a part of (or together with) a facial recognition system. It is also used in video surveillance, human computer interface and image database management. Some recent digital cameras use face detection for autofocus. Face detection is also useful for selecting regions of interest in photo slideshows that use a pan-and-scale Ken Burns effect. Face detection is gaining the interest of marketers. A webcam can be integrated into a television and detect any face that walks by. The system then calculates the race, gender, and age range of the face. Once the information is collected, a series of advertisements can be played that is specific toward the detected race/gender/age. This paper shows prototype or partial implementation of this type of work. Face detection is also being researched in the area of energy conservation [Energy Conservation]

3. DESCRIPTION OF TOOLS

In this section, the tools and methodology to implement and evaluate face detection and tracking using OpenCV are detailed.

OPENCV

OpenCV (Open Source Computer Vision Library) is a library of programming functions mainly aimed at real time computer vision, developed by Intel. The library is cross-platform. It focuses mainly on real-time image processing.

Figure 3: Object Detection Pattern using OpenCV

The library was originally written in C and this C interface makes OpenCV portable to some specific platforms such as digital signal processors. Wrappers for languages such as C#,

Python, Ruby and Java (using JavaCV) have been developed to encourage adoption by a wider audience. However, since version 2.0, OpenCV includes both its traditional C interface as well as a new C++ interface. This new interface seeks to reduce the number of lines of code necessary to code up vision functionality as well as reduce common programming errors such as memory leaks (through automatic data allocation and deallocation) that can arise when using OpenCV in C. Most of the new developments and algorithms in OpenCV are now developed in the C++ interface. Unfortunately, it is much more difficult to provide wrappers in other languages to C++ code as opposed to C code; therefore the other language wrappers are generally lacking some of the newer OpenCV 2.0 features.

FACE DETECTION

In this section, the base algorithm used to detect the face is discussed. AdaBoost algorithm is discussed first then feature selection is discussed.

ADABOOST

In 1995, Freund and Schapire first introduced the AdaBoost algorithm . It was then widely used in pattern recognition.

The AdaBoost Algorithm

1. **Input**: Give sample set S = **(**x1, y1), ...(xn, yn) xiϵ X, yiϵ Y = {-1, +1}, number of iterations T

2. **Initialize**: $w_{i,j} = \frac{1}{N} i = 1, \dots N$

3. **For** t = 1, 2,…,T,

i) Train weak classifier using distribution Wt.

ii) Calculate the weight (w_i) training error for each hypothesis.

$$h_n \varepsilon_t = \sum_{i=1}^{N} W_{t,i} |k_i - y_i|$$

iii) Set:$a_t = \frac{1}{2} log \frac{1-\varepsilon t}{\varepsilon t}$

iv) Update the weights:

$$W_{t+1,i} = 1 + \frac{W_{t,i}}{Z_t} \times \begin{cases} e^{-a_t} \\ e^{a_t} \end{cases}$$

$$= \frac{w_{t,i} \exp(-a_t y_i h_t(x_i))}{Z_t}$$

4.Output: the final hypothesis, also the stronger classifier.

$$H(x) = sign\left(\sum_{t=1}^{T} a_t h_t(x)\right)$$

Feature Selection using Haar like Features

In the implementation of face detection, Xi contains a huge number of face features, and some of the features with low ϵ i to train our strong classifier are selected. By AdaBoost algorithm this can be achieved automatically. For each iteration ϵ i with each feature in Xi can be calculated and then the lowest one is what we need. For doing this, the face detection rapid could be very fast. In next part, you will find there are many haar-like features, so it is hard to make use of all them. Face features are abstracted from the input image and are used to train the classifier, modify weights.

Face features are abstracted from the input images and are used to train the classifiers, modify weights as mentioned . In 2001, Viola et al. first introduced the haar-like features. The haar-like features are rectangle features and value is that **the**

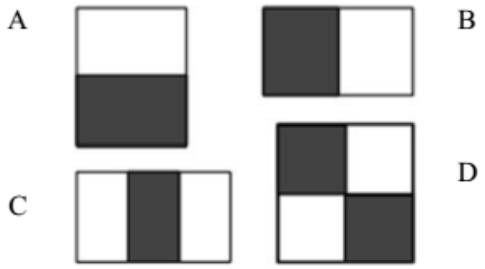


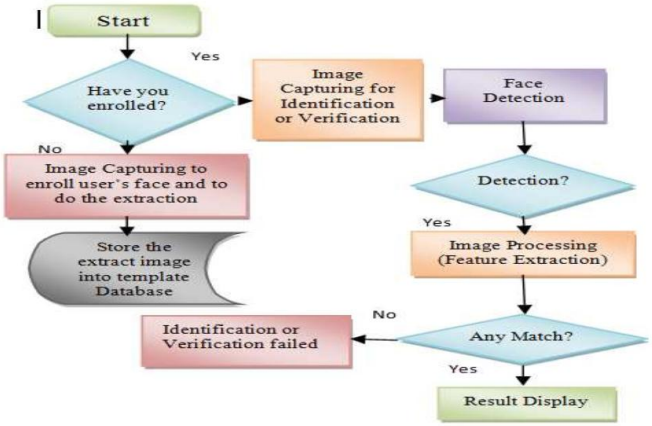
Figure 6: Haar-like Features Introduced in Viola's Paper

sum of pixels in black district subtracts the sum of pixels in white district. Rainer Lienhart had done an extended set of haar-like features which significantly enrich the basic set of simple haar-like features, and can get a better hit rate. Two-rectangle features are „A" and „B" . „C" is three rectangle feature and „D" is four-rectangle feature. At a size of 24x 24, there are more than 180,000 rectangle features.

4. FACE DETECTION AND DESIGN ANALYSIS

This section will describe about the Face detection with itself which has several modules that are working together as one to make the system runs smoothly. The phase consists of capture image; Detect faces in the image, feature extraction, template comparison, declaration of matching template. The acquisition of face images can be done by acquiring the real-time image from the OV5647 CMOS Image sensor interfaced with Raspberry pi High speed processor with GPU Processing. Furthermore, the acquisition can also be done through real

time remote monitoring either with Ethernet connectivity. The function of the face detection module is to clarify whether the face is available during real time monitoring for detection or not. The face detection is done by scanning up an image for different scales and looking for some simple patterns. When the system detects the face, it will produce an sub-image and this sub-image is scaled such that the face appears in the center and presented at a uniform size. OpenCV already provide an algorithm to locate faces in still image and videos stream. Haar classifier algorithm scans the image and creates a bounding box as returns for each detected face.

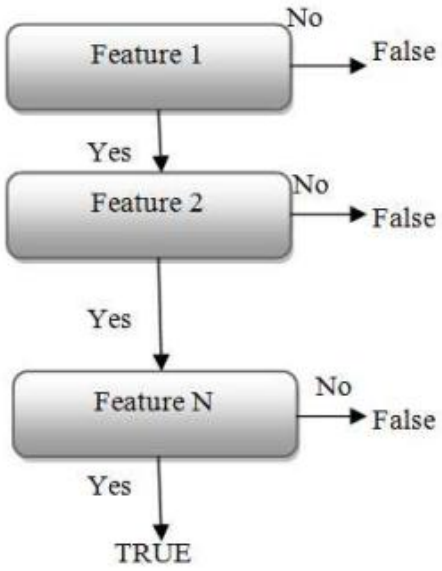


The feature extraction in face detection is done by localizing of the characteristics of face components (i.e., eyes, mouth, nose etc.) in an image. In other terms, the feature extraction is a step-in face detection and recognition where the system locates certain points on the faces such as corner and center of the eyes, tip of the nose, mouth etc. It analyzes spatial geometry of differential feature of a face. Result of this analyzing is a set of template generated for each face. The template consists of a reduced set of data which represent the real-time face detected in bounded box. The template comparison is done with the template stored in the database. Two phases are there in this phase identification and verification. These two-term identification to detect the face in real time video and verification application for face recognition which scope out of this paper. The final phase of face detection is to declare the highest matching score resulted in the previous step. The configuration will determine how the application should behave based on the desired security and operational consideration. The face detection methodology is shown in figure.

A. Face detection

This System is capable of detecting the faces from the captured image from HD Video for the purpose of analyzing and detecting the face. From the above Section IV, face detection determines where in an image, a face is located and it is being done by scanning the different image scales and extracting the exact patterns to detect the face. The Prototype is built with Haar-Like Feature function from OpenCV. Haar classifier face detection is used to create a search window that slide through a image and check whether a certain region of an image looks likes face or not. Haar like features and a large set of very weak classifier use a single feature to define a certain image as face or non-face. Each feature is described by the template and its coordinate relative to the search window which is the origin of the size of the feature.

The search window quickly scans the first classifier on the cascade as shown in the Figure 9, if the classifier returns false then the computation on that window also ends and results no detected face (false). Moreover, if the classifier returns true, then the window will be passed down to the next classifier in the cascade to do the exact same thing. When all classifiers return true for that window, then the result will return true also



for that certain window face is detected.

Figure . Decision tree based on Haar –like features (Cascade of classifier)

Software Required

OpenCV 2.3.1 super pack for windows, Raspbian OS, Putty,etc.

Hardware Required

PC preferably running windows, Rapberry Pi 3 model B , standard servos *2, webcam w/usb interface, breadboard, jump wires, hobby wire to tie pan/tilt servos and webcam together. Breadboard is used to make connections. The various connections required are as given below

SERVOS :

A **servomotor** is a rotary actuator or linear actuator that allows for precise control of angular or linear position, velocity and acceleration. It consists of a suitable **motor** coupled to a sensor for position feedback.

WEBCAM:

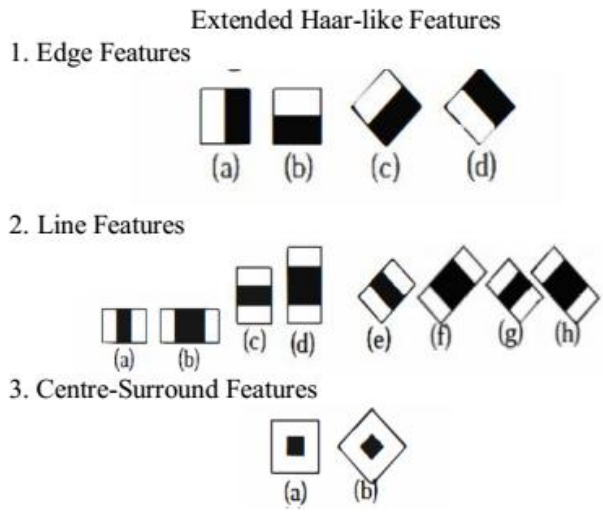
The webcam's USB goes to the pc. The code will identify it via a number representing the USB port its connected.

RASPBERRY PI :

All models feature a broadcom system on a chip (SoC), which includes an ARM compatible central processing unit (CPU) and an on-chip graphics processing unit (GPU, a VideoCore IV). CPU speed ranges from 700 MHz to 1.2 GHz for the Pi 3 and on board memory range from 256 MB to 1 GB RAM. Secure Digital (SD) cards are used to store the operating system and program memory in either the SDHC or MicroSDHC sizes. Most boards have between one and four USB slots, HDMI and composite video output, and a 3.5 mm phone jack for audio. Lower level output is provided by a number of GPIO pins which support common protocols like I²C. The B-models have an 8P8C Ethernet port and the Pi 3 has on board Wi-Fi 802.11n and Bluetooth.

5.IMPLEMENTATION

After a classifier is trained, it can be applied to a region of interest (of the same size as used during the training) in an input image. The classifier output is "1" if the region is likely to show the face and "0" otherwise. To search for the object in the whole image one can move the search window across the image and check every location using the classifier. Here we



use two different codes for face detection and tracking respectively. The algorithm used for both the codes (Processing & Raspberry Pi).

Implementation of Software

Processing takes the video input from the webcam and uses the OpenCV library to analyze the video. If a face is detected in the video, the OpenCV library will give the Processing sketch the coordinates of the face. The processing sketch will determine where the face is located in the frame, relative to the centre of the frame, and send this data through a serial connection to the Raspberry Pi. The Raspberry Pi will use the data from the Processing sketch to move the servos connected the Servo setup.

a) Basically haar-cascade classifier is used for detecting the faces.

b) The input video frame is read from camera and temporary memory storage is created to store

c) A window is created to capture the display frame and frame is continuously monitored for its existence.

d) A function is called to detect the face where the frame is passed as parameter.

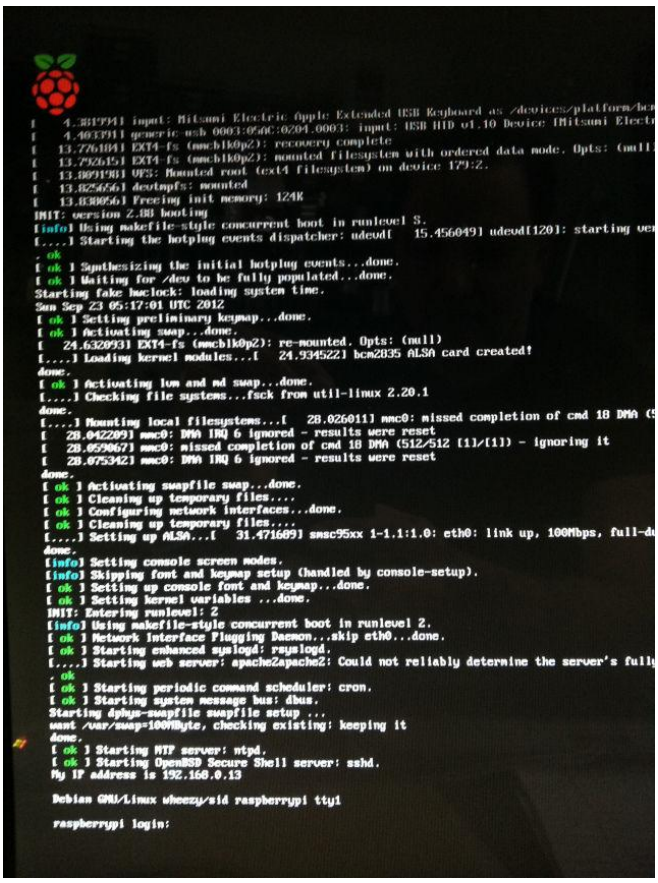
e) Steps b-d is kept in a continuous loop until the user defined key is pressed.

f) The classifier, frame, memory storage & the window are destroyed.

g) The (X, Y) coordinate of the image is plotted according to movement of face.

g) The (X, Y) coordinate of the image is plotted according to movement of face.

h) The difference between face position and centre is calculated and sent to Raspberry Pi serially.



6. RESULT AND ANALYSIS

The result of face detection is shown in Figure. Those are the frames extracted from the HD video streaming. Sometimes, face detection algorithm may get more than one result even there is only one face in the frame. In this case, a post image processing is been used for extracting the exact face coordinates with OpenCV and SimpleCV Haar Classifier libraries. If the system output provides more than one

rectangle, which indicates the position of the face, the distance of center points of these rectangles has been calculated. If this distance is smaller than a pre-set threshold, the average of these rectangles will be computed and set as the final position of the detected face. In this paper we also implement the face tracking application in Python language by using face detection. This method is verified and the limitations of the scheme are observed through testing and debugging our codes. And then, limited by Python performance, we shift to OpenCV to evaluate the speed of this face tracking scheme, We found the Viola and Jones face detection is more suitable for real-time face detection since they requires less CPU resource and costs shorter time.

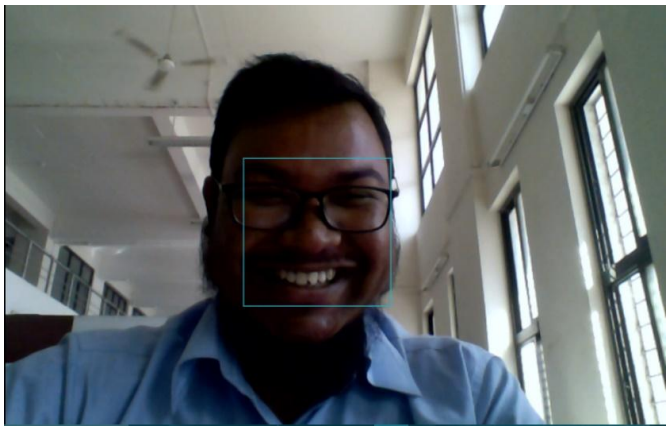


Fig. Output of Algorithm Showing the face detection

7.CONCLUSION

Face detection and tracking is being a challenge for many researchers with real time Image sensor. With the advancement the real time face detection in remote monitoring is helpful for building many efficient industrial and commercial applications. Moreover such technology can be useful in tracking the lost object under dynamic environment. Further enhancement of this work can be extended with stereo depth analysis of face detection using two image sensor interfaced with High speed Processor.

REFERENCES

[1] A. Faizi (2008), "Robust Face Detection using Template Matching Algorithm,", University of Toronto, Canada.
[2] P. Feng (2004), "Face Recognition based on Elastic Template,", Beijing University of Technology, China.
[3] L.H. Liang, H.ZH. Ai & G.Y. Xu (2002), "A Survey of Human Face Detection," J.Computers. China, Vol. 25, Pp. 1–10.
[4] K.J. Wang, SH.L. Duan & W.X. Feng (2008), "A Survey of Face Recognition using Single Training Sample", Pattern Recognition and Artificial Intelligence, China, Vol. 21, Pp. 635–642.
[5] Z. Zhang (2008), "Implementation and Research of Embedded Face Detection using Adaboost", Shanghai JiaoTong University, China.
[6] L. Guo & Q.G. Wang (2009), "Research of Face Detection based on Adaboost Algorithm and OpenCV Implementation", J. Harbin University of Sci. and Tech., China, Vol. 14, Pp. 123–126.
[7] CH. Y. Lu, CH.SH. Zhang & F. Wen (1999), "Regional Feature based Fast Human Face Detection", J. Tsinghua Univ. (Sci. and Tech.), China, Vol. 39, Pp. 101–105.
[8] H. J. Jiang (2007), "Research on Household Anti-Theft System based on Face Recognition Technology", Nanjing University of Aeronautics and Astronautics, China.
[9] P. Viola & M. Jones (2001), "Rapid Object Detection using a Boosted Cascade of Simple Feature", Conference on Computer Vision and Pattern Recognition. IEEE Press, Pp. 511–518.
[10] G. Bradski & A. Kaebler (2009), "Learning OpenCV", China: Southeast Univ. Press.
[11] Ben Fry & Casey Reas (2007), "Processing: A Programming Handbook for Visual Designers and Artists", MIT.
[12] Open Source Computer Vision Library Reference Manual-intel
[13] Gary Bradski & Adrian Kaehler O" Reilly (2008), "Learning OpenCV", O'REILLY Media.
[14] "Introduction to OpenCV", [Online] Available: www.cse.iitm.ac.in/~sdas/courses/CV_DIP/PDF/INTRO_C
[15] "DCRP Review: Canon PowerShot S5 IS", Available: http://www.dcresource.com/reviews/canon/powershot_s5-review/
[16] "Energy Conservation", Available: http://portal.acm.org.offcampus.lib.washington.edu/citation.cfm?
[17] Sarala A. Dabhade & Mrunal S. Bewoor (2012), "Real Time Face Detection and Recognition using Haar - based Cascade Classifier and Principal Component Analysis", International Journal of Computer Science and Management Research, Vol. 1, No. 1.
[18] Faizan Ahmad, Aaima Najam & Zeeshan Ahmed (2013), "Image-based Face Detection and Recognition: State of the Art", IJCSI International Journal of Computer Science Issues, Vol. 9, Issue. 6, No. 1.
[19] Hussein Rady (2011), "Face Recognition using Principle Component Analysis with Different Distance Classifiers", IJCSNS International Journal of Computer Science and Network Security, Vol. 11, No. 10, Pp. 134–144.
[20] S. Sankarakumar, Dr.A. Kumaravel & Dr.S.R. Suresh (2013), "Face Detection through Fuzzy Grammar", International Journal of Advanced Research in Computer Science and Software Engineering, Vol. 3, No. 2.

