Face Detection System for Criminal

Modhale Pallavi[1] Kshirsagar Amruta[2] Suryavanshi Arti[3]
[1,2,3]Department of Computer Engineering
[1,2,3]H.S.B.P.V.T College of Engineering, Kashti, India

Abstract— In this paper a criminal detection agenda that could help police to know the face of a criminal or a suspicious is planned. The agenda is a client-server video based face recognition investigation in the actual. The agenda smears face detection and tracking using Android mobile devices at the client side and video based face recognition at the server side. This paper attentions on the growth of the client side of the future agenda, face detection and tracking using Android mobile devices. For the face detection phase, strong Viola-Jones algorithm that is used. The face tracking phase is founded on Optical Flow algorithm. Optical Flow is applied in the future agenda with two feature extraction methods, Fast Corner Features, and Regular Features. The planned face detection and tracking is executed using Android studio and OpenCV library, and verified using Sony Xperia Z2 Android 5.1 Lollipop Smartphone. Trials show that face tracking using Optical Flow with Regular Features attains a higher level of accurateness and effectiveness than Optical Flow with Fast Corner Features.

Keywords: Face Detection System for Criminal

I. INTRODUCTION

Computer revelation emphases on repetition or imitation of human visualization. Therefore, it includes many methods that used to obtain, procedure, examine, and recognize images, and it uses a camera to analyze and understand acts in the real world. Human face is the maximum thing got the attention of the investigators because of the numerous critical applications associated to the human face reaching from shadowing systems to acting applications.

Face detection achieves a real-time performance through Viola-Jones framework where its detection rates are competitive with some of the best methods to date in terms of both performance and running time. Face tracking provides a solution to handle the real-time conditions and video attributes as a temporal continuity attribute but the time consuming of the most tracking algorithms makes tracking problem an open area of research. The recent research seeks to take a benefit of AdaBoost approach to set an initial window of a target object for a tracking method. Then, the tracking method is responsible for tracking the face by distinctive features, but in a fast manner optical flow is used.

To track objects, it is an adaptive algorithm based on the result of the previous frame. Optical flow method gets current key points and a homography transformation between the previous and the current frames.

In the technical context, Android developers tend to support Android mobile devices by biometric applications including face tracking. Face tracking using a hand-held camera of a mobile device must consider both the motion of the camera and the face object, and must be able relatively to handle the blurring resulting from shaking or significant displacement of the face to keep the tracking accuracy as high as possible. In addition, mobile devices also have many limitations in hardware resources like computing resources. These limitations make tracking problem on mobile devices an open area of research. Android platform has the ability to get benefits of Open Computer Vision library. It is a programming library mainly aimed a real-time computer vision. OpenCV provides Viola-Jones detector for detecting multiple faces under the real-time conditions.

In this paper a criminal detection framework is proposed. This framework, as shown in fig. 1, is a client-server video based face recognition surveillance in the real-time. The framework scenario is as following: the policeman capture a video for a criminal or a suspect using his Smartphone camera, a real time face detection and tracking is done at the client side. Then the video frames containing the detected and tracked face are sent to the server where a video based face recognition is done at the server side. The personal information record for the recognized person is sent back from the server to the policeman mobile phone. This paper focuses on the development of the face detect-track system on Android platform at the client side. The face detection stage uses Viola-Jones detector supported by OpenCV. The face tracking stage is based on Optical Flow algorithm, which is implemented using java. The performance of the proposed face detect-track system is compared using two feature extraction methods, Fast Corner Features, and Regular Features with Optical Flow algorithm.

The rest of this paper is organized as follows: Section II presents some of the related work. The proposed system that is implemented using Viola-Jones and Optical Flow algorithms on Android platform at the client side is described in Section III. Section IV discusses the results in terms of accuracy and time efficiency. Finally, Sections V concludes this work and highlights the future work.

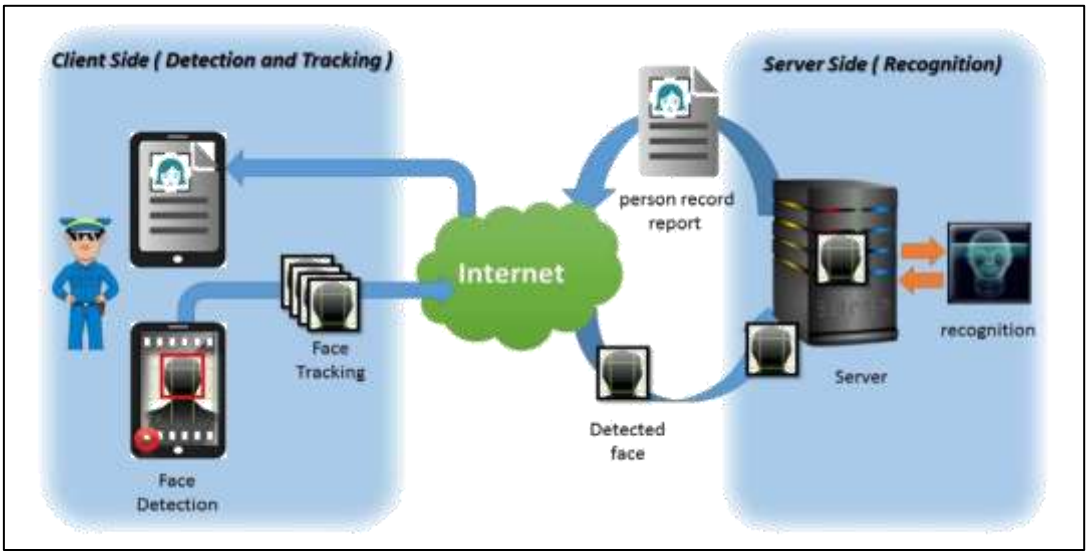

Fig. 1: The Proposed Client-Server Criminal Detection Framework

II. RELATED WORK

Face recognition is applied for criminal detection in many researches. The research work in [8] proposed the design of an Android application which compares multiple faces. The system matches faces using Eigenfaces algorithm based on Principal Component Analysis. At the client side, the Android mobile user takes an image or a video as an input and then passes it to the web server using HTTP method. At the server side, an image or video match with the existing criminal information record in a database is done. The proposed system in [9] recognizes the criminal using images from live streaming CCTV. The system compares these images with the criminal information in database records, then displays a specific information if the image matches with the database content. In the face detection stage, it uses the HAAR's algorithm and for recognition stage it uses the EIGEN values.

There are many researches applied in the computer vision field handled real-time face detection and tracking. The Research work in presented a model that is composed of two-mode tracking: short range tracking mode, and long range tracking mode. This model is used to deal with unstable real-time face tracking situations, Like face scaling, face pose changes, and face abrupt movements. The first mode is used for changes in scaling and appearance. The second mode is used to capture fast and abrupt motion by particle filter and Continuously Adaptive Mean Shift. In the face detection stage they used Viola and Jones to locate the face beforehand and give the tracking system the important face features including corner points and color histogram. Optical Flow and CAMShift are used in the short-range tracking to address the variance face size and appearance and make tracking step more accurate. If a failure case detected by any of the previous methods, the initial position of CAMShift will be computed by Particle Filter to track fast movement of interest object. The authors in used three basic detectors that trained by local binary pattern and boosting algorithm. They are expanded for multi-view face detection. To accelerate the face tracking process, they proposed a robust facial pose estimation algorithm and the face matrix partition scheme. The authors in developed a real-time head pose estimation solution. Their proposed system has been implemented on Mobile Platforms. Three algorithms are used: Viola-Jones for face detection, color tracking, and an efficient head pose estimation algorithm.

Optical Flow and Viola-Jones algorithms are widely used in real time systems. The research work in proposed a real-time face tracker using viola-Jones and Optical Flow algorithms. The system builds a likelihood map from the intermediate results of the Viola-Jones algorithm. Then the likelihood map is extrapolated using Optical Flow. The research work in proposed a face tracking algorithm using real time camera. They used Haar-like features to detect the face object, Optical Flow for tracking stage, and Shi and Tomasi algorithm to extract feature points.

III. THE PROPOSED CLIENT SIDE FACE DETECT-TRACK SYSTEM

At the client side, the aim is to detect a face object and track it during the capturing operation, and enable the users to preview the face tracking functionality in real-time conditions. Viola-Jones algorithm is used for face detection. For tracking stage, Optical Flow algorithm is used because it can handle features deformations like changes in shape, orientation, and can reduce the level of false positive resulted by AdaBoost cascade. Fig. 2 shows the proposed face detection and tracking pipeline diagram where tracking result is updated through a face detect-track cycle.

A. The proposed Face Detect-Track Cycle

The proposed face detect-track cycle deals with frames generated in the real-time in which N is the number of the captured frames. The detection takes its role at the first frame and at every m frames to allow other faces to be detected and then tracked. The detection stage determines faces windows, and removes any faces not found longer. In every tracking iteration the face is preserved by distinctive and efficient features. The features extraction method produces the facial points to be used in the optical flow function.

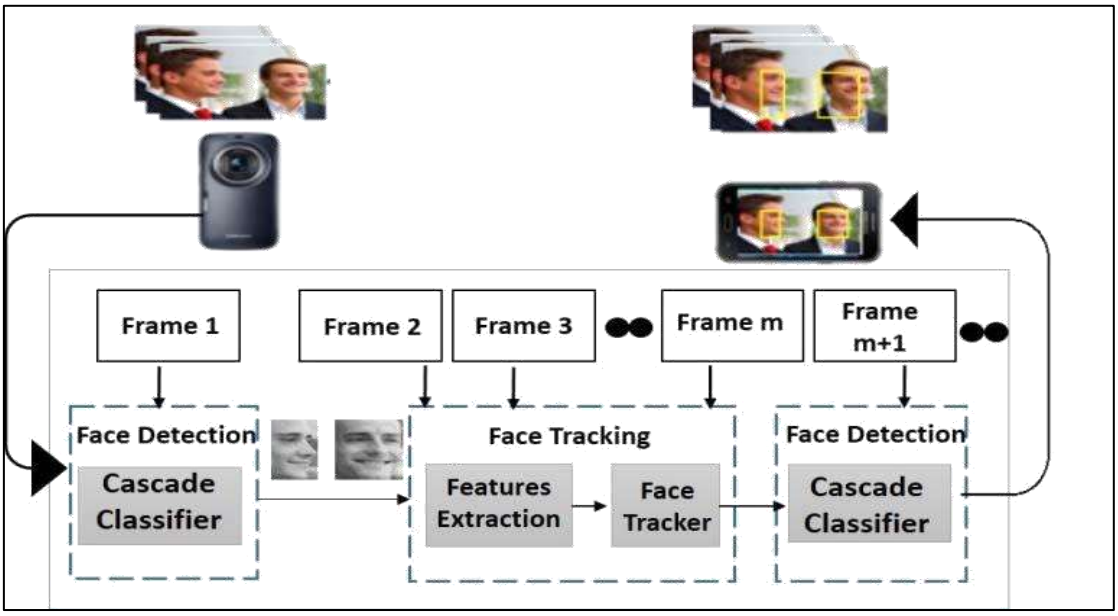

Fig. 2: The Proposed Face Detection and Tracking Pipeline Diagram

Optical Flow function takes the previous frame, the face points, and the next frame as inputs to predict the face location in the next frame. After m frames, the system will start a new detect-track cycle, as shown in fig. 3.

B. Face Detection

In the face detection stage, Viola-Jones algorithm is used which is AdaBoost-based approach. This algorithm achieves real-time performance because it computes integral images for the cascaded windows. Viola-Jones algorithm provided as a built-in function by OpenCV library to detect a face object.

C. Face Tracking

The face tracking stage uses the Optical Flow algorithm. Optical Flow is a pattern of apparent motion of image objects. This motion is caused by the movement of the object or the camera between two successive frames. It is a two dimensional vector field, where each vector represents the difference between the position of an object before and after its movement as shown in Fig 4.

For a pixel I (x, y, t) in the first frame. In the next frame, after time, this pixel moves a distance (dx,dy). If those pixels are the same and there is no change in their intensities, then,

$$I(x ,y ,t) =(x+dy ,y + dy ,t +dt) \quad (1)$$

To get the Optical Flow equation (2), take the Taylor series approximation of the right hand side of equation (1), remove
the common terms, and divide by

$$fxu+fyv+ft=0 \quad (2)$$

Where:

$$Fx=df/dx;fy=df/dy \quad (3)$$

$$U=dx/dt;v=dy/dt \quad (4)$$

and are image gradients. is the gradient along time. And in equation (2) are unknown. Several methods are provided to solve the Optical Flow equation. Lucas-Kanad is one of these methods which is used in the proposed system.

The Optical Flow needs feature extraction as a preprocessing step. The feature is a unique pattern that helps for tracking. In the proposed framework, two methods to extract the facial features are used, Fast Corner method and Regular Feature method.

1) Fast Corner Method

Fast Corner Method is used for feature point's detection in a face window. In Fast Corner Method, a pixel p is classified as a center point by analyzing a circle of sixteen pixels around it, as shown in Fig 5. Then, finding at least adjacent pixels with intensities larger than the intensity of the pixel p from these sixteen points. These n points present key feature points in the detected face window, and are used to track the face by Optical Flow. The Fast Corner method steps are listed in Algorithm 1.

2) Regular Features Method

To extract Regular Features [18], the face window is represented by a matrix × where n is the number of rows and m is the number of columns. Then, select the features as points within the height and width of the face window. The Regular Features Method steps are listed in Algorithm 2.

IV. RESULTS AND DISCUSSION

A. Software and Hardware Tools

Android Platform supports many APIs introduced by Google Company. Android studio is used as an environment for system implementation. It is supported by Android Software Development (SDK) [19]. In addition, we used OpenCV library to take the benefits of optimized implementation of its functions. OpenCV is a programming library aims to focus on real -time image processing. It deals with multiple programming languages like Java and platforms like Android. However, OpenCV library cannot work in Android platform without Android OpenCV manager which manages OpenCV library binaries on the end users' devices. The most important built-in function used in the proposed system is Viola-Jones.

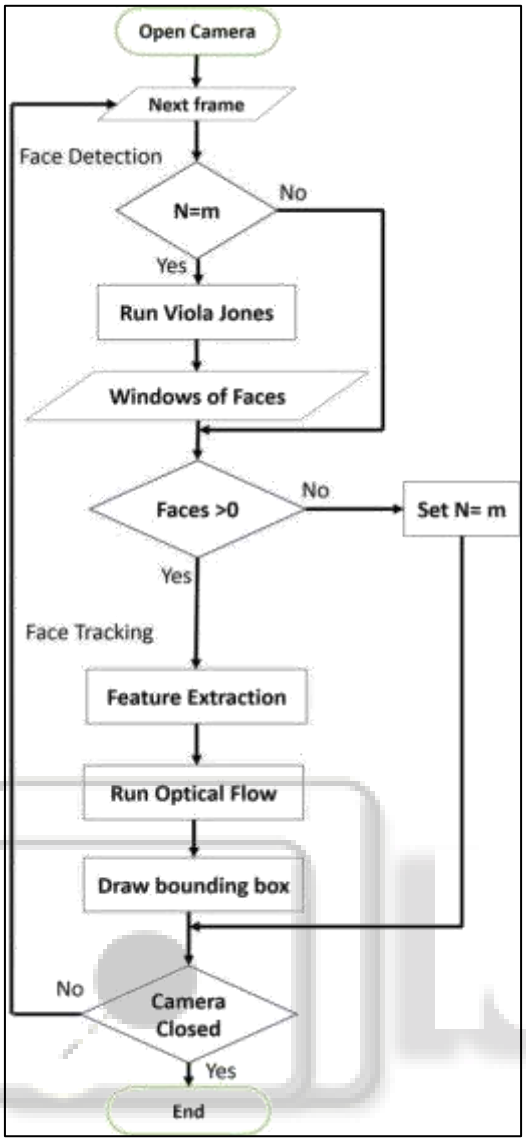

Fig. 3. The proposed Face Detect-Track Cycle

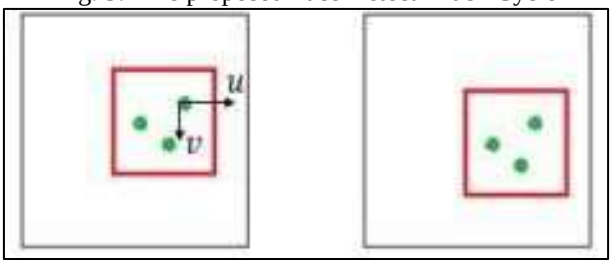

Fig. 4: Points movement between two successive frames

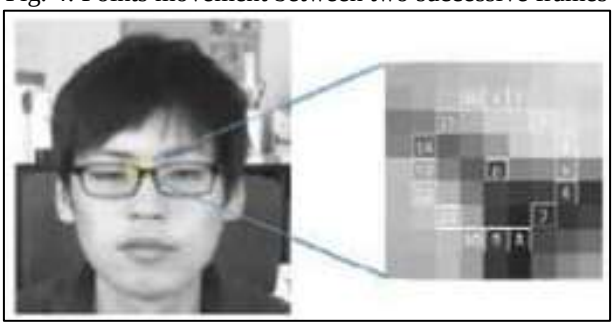

Fig. 5: Fast Corner Features Extraction [5].

The proposed client side face detect-track system is tested on Sony Xperia Z2 Android 5.1 Lollipop Smartphone, which has a 20-megapixel camera, and a Quad-core 2.3 GHz Krait 400 processor.

B. Performance Evaluation

A set of parameters are used to measure the performance of the proposed real time face detect-track system. These parameters are: Frame Per Second (FPS), which represents the real-time system speed or time efficiency, False Negatives (FN), which is the number of face objects that have not been detected, and False Positives (FP), which is the number of non-face objects that have been detected.

C. Face Detect-Track Experimental Results

The proposed face detect-track system on Android platform is tested on real time captured videos. In order to lower the demand of computational power of the proposed system, the Viola-Jones face detection algorithm is executed every 15th frame. The Optical Flow tracking is executed using the two feature extraction methods, Fast Corner features, and Regular features. Fig. 6, and fig. 7 show sample results of two real time captured videos. Table I shows the system performance in terms of FPS, FN rate, and FP rate when tested on real time videos. From Table I, it is shown that using Optical Flow with Regular Features, the system real time speed is 7 FPS, the FP rate is 38%, and the FN rate is 6%. Using Optical Flow with Fast Corner features, the system real time speed is 3 FPS, the FP rate is 48%, and the FN rate is 41%. The system performance using Optical Flow with Regular Features is better than when using Optical Flow with Fast Corner features. The reason of the Optical Flow with Fast Corners has a lower rate is due to the huge operations carried out on a single frame. It tests sixteen pixels around each single pixel in the image. In general, Optical flow does not works well with fast motion as it lost the face location, as shown in fig. 6 (e), fig. 6 (f), and fig. 7 (e).

V. CONCLUSION AND FUTURE WORK

In this paper, a criminal detection framework is presented. This framework is a client-server video based face recognition surveillance in the real-time. This paper focuses on the implementation of the client side face detection and tracking on Android smartphones. The face detection is implemented using Viola-Jones ready-to-use function provided by OpenCV library. Optical Flow is used for tracking stage. We compared between two techniques for extracting features that used by Optical Flow algorithm, Fast Corner method and Regular

Feature method. Regular Features achieved better results as it processes 7 FPS while the fast corner processes 3 FPS.

Although the proposed system presents an acceptable tracking function, we are looking forward to enhance its accuracy and efficiency. Cooperating Optical Flow with CAMShift algorithm will handle large, and fast movements, and they will improve the system accuracy. In addition, we will work on the implementation and evaluation of the face recognition at the server side.

A. Algorithm 1: Extract Features by Fast Corner Method

Input: Face window
Output: List of features

1) if (List of features == null) Do
2) for i=0 To FaceArrayLength Do
3) Define the coordinates of face region by two points (x1, y1) and (x2, y2)
4) for x1+3 To x2-3 Do
5) for x1+3 To x2-3 Do
6) Define center point p
7) Define 16 points n1, n2, n3……. n16 around center p
8) Check all points intensities that are greater than center point p
9) if (greater Points >14) Do
10) Add points to list of features
11) else keep the previous List of features
12) end
13) end
14) end
15) end
16) end
17) return List of features

B. Algorithm 2: Extract Features by Regular Features Method

Input: Face window, integer a, and b.
Output: List of features

1) // Matrix rectFace [0.. n; 0..m]
2) generate upper left corner point (x, y)
3) rowStep ← a // a = 20
4) colStep ← b // b = 25
5) for j← 0 To n-1/ rowStep Do
6) for k← 0 To m-1/colStep Do
7) Add point (x + n∗j, y + m∗k) to list of features
8) end
9) end
10) return List of feature

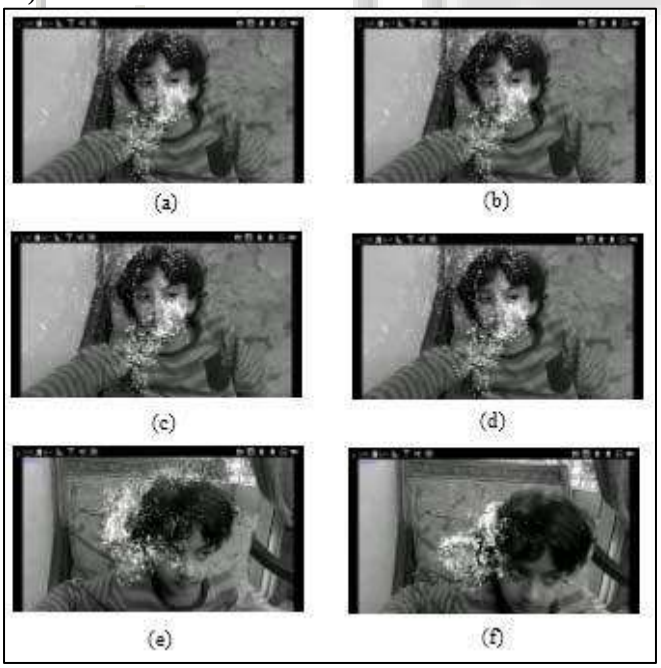

Fig. 6: System results using optical flow based on fast corner method (a) shows the first frame when detecting the face, and run the tracker, (b),(c), and (d) show the result of the tracker on some frames, (e), and (f) show false positive, and false negative.

Fig. 7: system results using optical flow based on regular feature method (a) shows the first frame when detecting the face, and run the tracker, (b), (c), and (d) show the result of the tracker on some frames, (e) shows a false positive, and (f) shows a false negative.

	Frame size	#of Frames	FPS	FP rate	FN rate
Optical Flow with Fast Corner features	1280 * 720	200	3	48%	41%
Optical Flow with Regular Feature		127	7	38%	6%

Table 1: The Proposed Face Detect-Track System Performance

REFERENCES

[1] R. Klette, Concise Computer Vision, Springer London, 2014.
[2] P. VIOLA and M. JONES, "Robust Real-Time Face Detection," International Journal of Computer Vision, vol. 57, no. 2, pp. 137-154, 2004.
[3] A. Ranftl, "Face Tracking Using Optical Flow," Halmstad University, Halmstad, 2014.
[4] L. Montanini, E. Cippitelli, S. Spinsante and E. Gambi , "Low complexity head tracking on portable android devices for real time message composition," Journal on Multimodal User Interfaces, vol. 9, no. 2, pp. 141-151, 2015.
[5] V. Q. Nhat, S.-H. Kim, J. H. Yang and G. Lee, "Real-time Face Tracking with Instability using a Feature-based Adaptive Model," Intrenational Journal of Control, Automatin and Systems, vol. 13, no. 3, pp. 725-732, 2015.

[6] Qi, Cao; Ruishan, Liu;, "Mobile Computer Vision," 8 Jun 2015. [Online]. Available: http://web.stanford.edu/class/cs231m. [Accessed 8 December 2016].
[7] "OpenCV | OpenCV", Opencv.org, 2017. [Online]. Available: http://opencv.org/. [Accessed: 09- Sep- 2015].
[8] Viraj, M. Pradip, T. Pankaj and M. hweta, "Criminal Detection Using Eigenfaces Approach on Android Device," International Journal of Computer Science and Information Technologies, vol. VI, no. 1, pp. 539-541, 2015.
[9] M. Prathamesh, R. Vedant, M. Suraj, K. Avinash and S.V.Wankhade, "Criminal Tracking System using CCTV," Imperial Journal of Interdisciplinary Research, vol. II, no. 7, pp. 206-208, 2016.
[10] J. L. a. K. W. Lei Xu, "REAL-TIME AND MULTI-VIEW FACE TRACKING ON MOBILE PLATFORM," in IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Prague, pp. 1485-1488, 2011.
[11] M. R. N. K. L. E. Jianfeng Ren, "Real-Time Head Pose Estimation on Mobile Platforms," SYSTEMICS, CYBERNETICS AND INFORMATICS, vol. 8, no. 3, pp. 56-62, 2010.
[12] G. Divya and J. Arunkant, "FACE DETECTION AND TRACKING AT DIFFERENT ANGLES IN VIDEO," ARPN Journal of Engineering and Applied Sciences , vol. 10, no. 17, pp. 7678-7683, 2015.
[13] R. Andreas, A.-F. Fernando and K. Stefan, "Face Tracking Using Optical Flow," in International Conference of the Biometrics Special Interest Group (BIOSIG), Darmstadt, pp. 1-5, dol. 10.1109/BIOSIG.2015.7314604, 2015.
[14] C. Dan, V. Maria and F. Fernando, "REAL-TIME FACE TRACKING METHODS," Universitat Aut`onoma de Barcelona, Bellaterra, 2009.
[15] K.-B. Ge, B. Fang and J. Wen, "ADABOOST ALGORITHM BASED ON MB-LBP FEATURES WITH SKIN COLOR SEGMENTATION FOR FACE DETECTION," in International Conference on Wavelet Analysis and Pattern Recognition, Guilin, pp. 40-43, 2011.
[16] "Cascade Classification OpenCV 2.4.13.2 documentation", Docs.opencv.org, 2017. [Online]. Available: http://docs.opencv.org/2.4/modules/objdetect/doc/cascade_classificatio n.html. [Accessed: 09- Sep- 2015].
[17] Doxygen, "OpenCv Open Source Computer Vision," OpenCv , 18 December 2015. [Online]. Available: http://docs.opencv.org/3.1.0/d7/d8b/tutorial_py_lucas_kanade.html#gsc.tab=0. [Accessed 8 December 2016].
[18] T. Nisarg and K. Salil, Mastering OpenCV Android Application Programming, Packt Publishing, 2015
[19] S. Notes, "SDK Tools Release Notes Android Studio", Developer. android.com, 2017. [Online]. Available: https://developer.android.com/studio/releases/sdk-tools.html. [Accessed: 09- Jan- 2016].

