Air-Swipe Gesture Recognition Using OpenCV in Android Devices

Twinkle Sharma, Sachin Kumar,Naveen Yadav,Kritika Sharma,Piyush Bhardwaj
Department of C.S.E and I.T.
BPIT
Delhi, India

Abstract—***There is a need to enhance communication between human and computers which is being greatly defined in this changing era of technology with Human Computer Interaction (HCI) which is helping in determining new communication models and accordingly new ways of interacting with machines. Current smartphone inputs are limited to physical buttons, touchscreen input, cameras or built-in sensors. The rapid development of Smartphone's in the last decade was mainly due to interaction and visual innovations. For Example the given approaches of input either require a dedicated surface or Line-of-Sight for interaction. But in today's scenario of increasing computability of smartphone or other gadgets and their decreasing sizes have raised a need for such touch free operations over these gadgets. In such an intendment we introduce Air-Swipe Gesture Recognition System which can be useful to enable user to make In-Air gestures in front of the camera and to do different operations. This System can give a user-friendly and a live-experience of interaction and visualization, enhancing the usability and making the android device more interactive. It does not require any hardware changes instead only uses the native camera of the device and a machine learning software such as Open Source Computer Vision (OpenCV) algorithms to detect the changes in environment and respond accordingly in varying conditions. We tested this classification and found out the result that considering the frames to be divided into quadrants along x-axis and y-axis and found that the value of the frame matrix changes. Our approach has the capability of recognizing gestures with precision of almost 96%.***

Keywords—Air-Swipe Gesture Recognition, Android, OpenCV, HCI, mobile interaction, computer vision.

I. INTRODUCTION

The vision of universal access to information has become the state-of-the-art in the mobile devices. Accessibility of the data is at our fingertips wherever we are. The way we consume and produce information has literally been changed by these devices. However, the best way of interaction ("HCI", n.d.) with mobile content is a long way from its solution. Though popular and intuitive, direct touch interaction also comes with drawbacks. Due to continue condense sizes of mobile devices, touch technology is becoming progressively limited, leading to smaller on-screen targets and fingers causing barrier to displayed content (Song et al., 2014). For Example while reading any text on a mobile device or when performing complicated manipulations that require many touchscreen controls.

These mobile devices can be upgraded and can be implemented to a more natural interface of interaction in the coming future. More changes can be made to transform the We can make more changes to transform these computer vision devices to become an input instruction rather than just a function of taking photo and an alternative to the keyboard in the near future.

Earlier OpenCV was particularly restricted to computer interaction and visualization. Now computer vision has extended its growth to various other devices ("Manual OpenCV4", n.d.). We augment this implementation to Android Devices.

Air-Swipe Gesture Recognition:
Interpretation of human gestures using mathematical algorithms is defined as Gesture recognition. Gesture includes the movements of body parts that imply a specific meaning or message that can be used for communication between the sender and the receiver. Gesture Recognition can act as a bridge between humans and computers by understanding human body language by machines. Air-Swipe gesture will be a great impact on human communication. Air-swipe gesture involves dynamic hand movements and static gesture too. Static gesture is just a hand posture in a steady position over the front camera of the device. In dynamic gestures, wavy hand movements are involved in different directions. Types of gestures recognition, then subtypes of the vision based gesture recognition systems (Joshi et al., 2015) which are appearance based technique and 3D model based techniques.

Open Source Computer Vision (OpenCV):
OpenCV (Open Source Computer Vision Library) is an open s ource machine learning software library based on computer vision("About OpenCV", n.d.). It was built to have a common framework for computer vision applications and to increase the use of machine approach in the commercial market. It helps in making it easy for developers to modify and utilize the code. It has over 2500 optimized algorithms including classic and state-of-art machine learning and computer vision algorithms. These algorithms can be used for various purposes like detection and assimilation of faces, classifying human actions in videos, identifying objects, tracking movement of objects, tracking camera movements, extracting 3D models of objects, producing 3D point clouds from stereo camera, find similar images from the database, stitching images together to make a high resolution image of an whole scene, follow eye movements, removing red eyed effect from images, conceive scenery and show markers to overlay it with developed reality, etc.("Android platform", n.d.) OpenCV has above 47 thousand

of user community and has a huge number of downloads. The library is extensively used in research works, governmental bodies and companies. OpenCV has various uses like it can be used to stitch street-view images together, monitors mining equipments in China, helps robots to navigate and pick up objects at Willow Garage, detects intrusions in surveillance video in Israel, detects swimming pool drowning accidents in Europe, checks runways for scrap in Turkey, inspects labels on products in factories around the world and rapid face detection in Japan, runs interactive art in Spain and New York. OpenCV is written natively in C++ and has a template interface that works seamlessly with STL containers.

Using vision-based interpretation (Rautaray et al., 2012) devices the use of keyboard may be eliminated in future Android Devices. In the present research scenario different challenges have been considered when defining an environment for acquiring the air-swipe gesture recognition techniques for working in the virtual environment

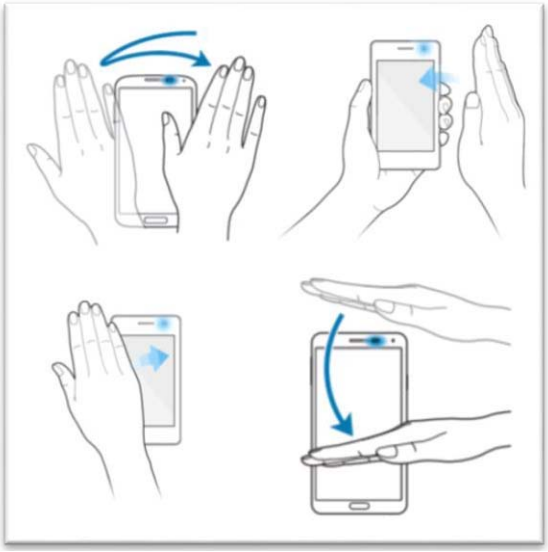

Fig. 1 Types of Air-Swipe Gestures showing the waving hand movements

One of the challenges encountered includes the noisy environment where detecting and recognising human Air-swipe gestures become difficult and thus the performance get affected by the problems or issues caused due to these challenges. The designing of this application motivates cost effectiveness and utilises cost effective in-built tools for example capturing of gestures as input can be done by the front camera of the device (Rautaray et al., 2012).

II. Related work

Mobile interaction is currently based on the touchscreen input but this restricts the expressiveness of the input to a certain level. Recently some models are designed based on computer vision to recognize the gestures on the limited resource devices like Computers and Laptops (Joshi, 2015; Patil, 2016). Here we try to transform the input space from touchscreen to the areas in front of Android devices like by waving hands to interact with the device which can be beneficial for deaf and dumb persons to connect and share thoughts with each other as shown in Fig. 1 that shows the various Air-Swipe hand gestures by waving hands.

Computer Vision can be one way to make use of the robust processing power of your smartphone – the way images are perceived by human eyes, computer vision can provide the same ability for a device to acquire process, understand and analyze these images. The modern smart phones or gadgets like Android Devices can use the powerful CPU to interpret the images captured through the camera. Examples of use cases are Air-Swipe detection and recognition or simple post-processing of photographs. OpenCV library is the best approach to use computer vision on Android Devices. OpenCV uses the front camera of the android devices to capture the image and further frame pre-processing gesture extraction and assimilation is done by device (Joshi et al., 2015). For Assimilation machine learning algorithm such as Matrix formation is used. It assimilates the wave gestures with an accuracy of 96%. This application is helpful only for Air-Swipe gestures recognition. The acquired images from the front camera are formed into frames and these captured frames are sent for frame pre-processing stages like to pick up the most suitable size for the preview frames, as too small frames will result in a bad outcome when we do the processing and will lead to errors, and too large frames will slow down the complete procedure to an insupportable level. We would need to pick up a minimum acceptable size and go through each of them to find the correct one because there are various set of sizes supporting the preview frames of the camera. Fig. 2 gives the complete system architecture of the working of native camera in android devices with the help of OpenCV to recognize the gestures. OpenCV has its own Java API (Application Program Interface) (Hellman, 2013), and doesn't rely on the Java-based Android Camera API.

System based on OpenCV recognizes waving of object using a device of good RAM with 92% accuracy. This system is flexible to location, scale and orientation changes. An adequate distance between two fingers is required to recognise gesture by the system. But the system gets confused in assimilating the gestures involving thumb because of the short length of thumb (Joshi et al., 2015). Many works are coming up on Dynamic Hand Gesture Recognition (Dixit V. & Agrawal A., 2015) that can be combined with OpenCV library to create a wide range of gestures that can be recognized easily giving a natural interface of interaction.

III. Proposed system

In the dawn of hand gesture troubleshooting was done on various versions of Android devices to detect the in air gesture being performed. An accepted and widely used example for hand gesture recognition is (Shaikh et al., 2016) Air-Swipe

Gesture Recognition. Progression of computer hardware and software has enhanced a lot in the present circumstances; this also affects the computing performance.

Improvements or advanced use of gesture recognition has supplanted the role of standard input methods to a more automated screenplay due to the additional costs of gestures and their complications in detection. Our system opens up the input space in front of the native camera of the Android device ("Camera Preview", n.d.). In doing so, our aim is to connect the gestures performed in front of the device with the regular touchscreen inputs. Before explaining the technical gesture recognition approach, we exhibit numerous convincing interaction scenarios, enabled by air-swipe gestures in front of the android devices, especially in multi-modal input (Song et al., 2014).

The air-swipe gesture can also perform in a more closely incorporated environment, in which the up and down swipe gesture invokes scrolling of a document being read on the device. Also, the left and right swipe gesture can be used for viewing the succeeding and preceding videos and photos in the Gallery. Similarly this can be used for Music Player, connecting and disconnecting calls and for various other activities on your Android Devices.

Combination of touch and gestures inputs concurrently modified multiple parameters and made them more integrated than that of the touchscreen inputs alone. Similarly, more multifaceted user-interfaces that involve explicit state changes can also be possible by gestures around the device for streamline interaction and allow the user to parameterize tools and functionalities. The algorithm proposed in this paper consists of three main stages- Matrix Allocation, Frame Processing and Gesture Extraction and Assimilation stage as shown in Fig. 3. The matrix allocation stage allocates matrices for the frames to be processed as effectual as possible; the allocation for the needed matrices will be done only once and will be re-used for every new frame. The first stage that is the frame processing constructs the input gesture, obtained from the front camera for further stage processing. Also, the Gesture extraction uses the preprocessed frame from the earlier stage and compares the new frame with the previous frame and extracts feature using Open Source Computer Vision (OpenCV). These frames retain 90% of the image data (Phansalkar et al., 2014).

This is very useful for real-time computation. All the relevant features are extracted from the images when the system traverses the images through all the stages.

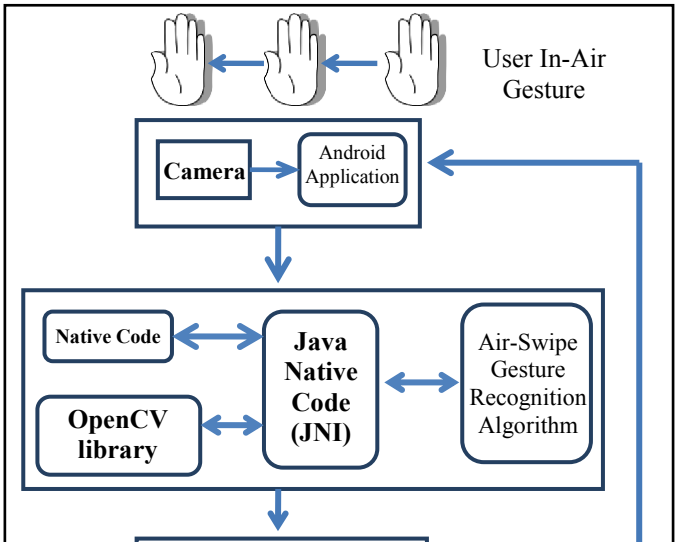

Fig. 2 System Architecture

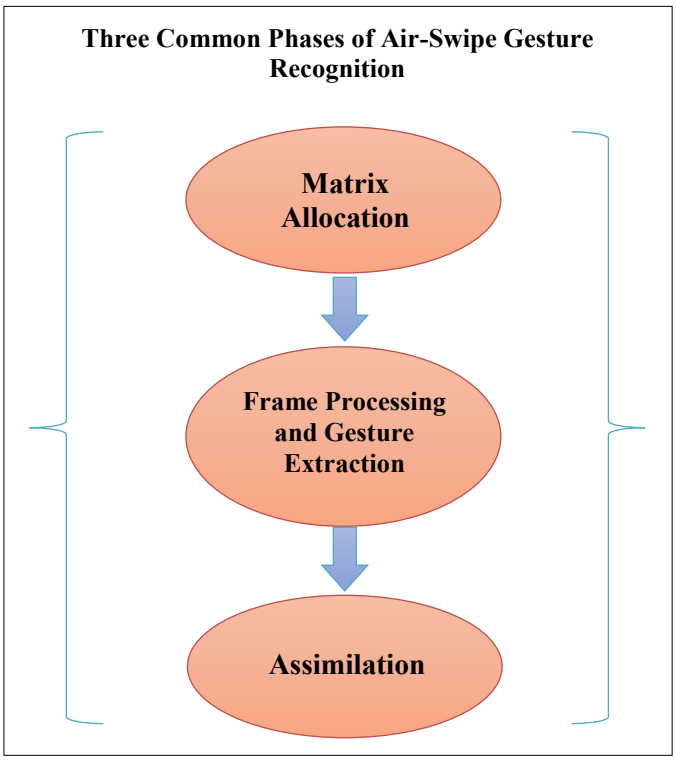

Fig. 3 Common Phase of Air Swipe Gesture Recognition

A. MATRIX ALLOCATION

The peripheral and radial factors are being taken into account for the distortion by OpenCV ("Camera Calibration", n.d.). For the radial factor, the formula used is as follows:

$$x_{corrected} = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)$$
$$x_{corrected} = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)$$

So for any pixel located at (x , y) coordinates in the input image, its location on the corrected output image will be $(x_{corrected} , y_{corrected})$. The presence of the radial deformation is evident in form of the "barrel" or "fish-eye" effect.

Peripheral deformation occurs because the image capturing lenses are not absolutely parallel to the imaging plane. It can be corrected via the formulas:

$$x_{corrected} = x + [2p_1xy + p_2(r^2 + 2x^2)]$$
$$y_{corrected} = y + [p_1(r^2 + 2y^2) + 2p_2xy]$$

So we have five deformation parameters which are presented in OpenCV as one row matrix with 5 columns:

$$Deformation_{coefficients} = (k_1\ k_2\ p_1\ p_2\ k_3)$$

Now for the units translation we use the formula:

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

The use of homograph coordinate system (and $w=Z$) explains the presence of w here. The f_x and f_y (camera focal lengths) and (c_x,c_y) are the unknown parameters referred as optical centers expressed in pixels coordinates. A common focal length for both the axes is used with a given *'a'* aspect ratio (usually 1), then $f_y = (f_x * a)$ and a single focal length f in the upper formula. *Camera matrix* is defined by the matrix containing these four parameters. While the deformation coefficients remains constant independent of the camera resolutions used, these should be scaled along with the current resolution from the calibrated resolution ("Camera Calibration", n.d.).

Calibration is the process of determining these two matrices. Some basic geometrical equations are used through which the calculation of these parameters is done. It depends upon the chosen calibrating objects as to which equation should be used. Presently OpenCV supports the following objects for calibration:

- Classical black-white chessboard
- Symmetrical circle pattern
- Asymmetrical circle pattern

Basically, snapshots of these patterns are taken with the camera so that OpenCV finds them. A new equation is being resulted by each found pattern. An equation system is formed by a premeditated number of pattern snapshots for solving the equation ("Camera Calibration", n.d.). This number is less for the circle ones and higher for the chessboard pattern. For example, theoretically two snapshots are required by the chessboard pattern. So, 10 good snapshots are needed with different positions in input patterns because of noise present in input images.

B. FRAME PROCESSING AND GESTURE EXTRACTION

Now as the matrix is allocated on the native camera using OPENCV, we will rely on the preview frames from the camera since we can retrieve these images relatively fast ("Controlling camera", n.d.). For the preview frames we need to pick an appropriate size, as too small frames will result in cropping of pixels, and too large frames will slow down the system to an insupportable level. Thus, the outcomes will not be favorable while doing the processing phase. So we used a preview size of 320*240. Since smartphone cameras often have a different set of supported sizes for preview frames, we need to pick a minimum acceptable size by going through all of them to find the correct one. OpenCV has its own Java API, and has no need to rely on the Java-based Android Camera API (Hellman, 2013). A class Mat is used to represent an n-dimensional numerical array which is used in gesture recognition algorithm. Real or complex-valued vectors and matrices, grayscale or color images, voxel volumes, vector fields, point clouds, tensors, histograms can be stored by this algorithm.

As the gesture is done on the front camera of the device, the matrix detects the change and detects the gesture depending upon the changes in x-axis and y-axis of the matrix. The device compares the previous frame with the current frame to detect the change in environment. If the change occurs along x-axis of the matrix i.e. along the rows of the matrix, that means the LEFT or RIGHT gesture is done. Similarly, if the change occurs along the y-axis that means the UP and DOWN gesture is done. Fig. 4 shows the complete scenario of x-y axis for gesture recognition phase.

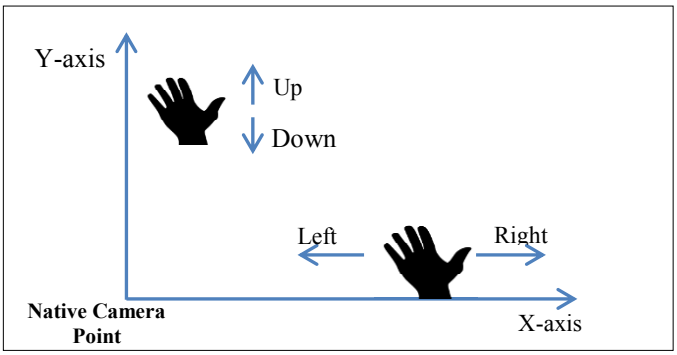

Fig. 4 Axis viewing in gesture recognition phase

C. ASSIMILIATION

Many researchers working presently in gesture recognition demonstrate that swipe gesture system can also be enforced into several types of environments inclusive of virtual environments and various systems. An interactive way developed by the researchers is the slideshow presentation system in the virtual environment. Hand gestures and movement researches helps in establishing models of the human body. This can help to build up a possibility for solving the challenges from mathematical point of view. Usually, pattern identification techniques are able to solve the problem with computation necessities and humbler hardware. In this research effort, we will examine these conditions of humbler hardware and computation by taking it as a plug to a smart

human computer interaction environment of control and virtual object handling. The user actions can be translated into commands which can be further executed in an intelligent (Rautaray et al., 2012) system and can implement the user necessities into practical action.

IV. Experimental Results

A dataset consisting of four gestures (left, right, up, down) performed by 5 different people was created by us. The values were recorded against various backgrounds at 30 frames per second (FPS) at 320×240 with a still front camera of an android device. Each gesture had an effect from four to nine seconds, and involved a minimum of four repeated cycles of the human action. The swipe actions produce drift in the upper and lower segment of the frame respectively (Bayazit et al., 2009), and similar motion around the torso.

Real-time outcomes were achieved with the help of resizing of the user frames before the optical flow computation. Datasets of 30 FPS each were collected by us, and based on the processing time for each action we measured the FPS for each of the actions. Table1 tells that a smaller resolution taken for the processing gains a powerful speed-up with min. loss of accuracy (Bayazit et al., 2009). The percentage amount of the time the correct air-swipe gesture gets the maximum recognition defines us the accuracy here as shown in Fig 5.

Resolution	Accuracy	FPS Mean	FPS Standard Deviation
160X120	0.85	28.67	0.4085
320X240	0.88	9.82	0.457

Table 1: Smaller resolution taken for the processing gains a powerful speed-up leading to increased efficiency with minimum loss of accuracy.

	UP	DOWN	LEFT	RIGHT
UP	1.00	0.00	0.00	0.00
DOWN	0.00	0.95	0.00	0.00
LEFT	0.00	0.00	0.93	0.00
RIGHT	0.00	0.00	0.00	0.96

Table 2: Confusion matrix for the four air-swipe gestures performed by all five people. Rows show the correct action, and columns give the value of actions returned by the classifier.

We tested the above classification and found out the result that:

- Considering the frames to be divided into quadrants along x-axis and y-axis. We found that the value of the frame matrix change from negative to positive along y-axis indicates an UP gesture. Table 2 shows the values of the matrix recorded.
- Similarly, value of the frame matrix changes from positive to negative along y-axis indicates a DOWN gesture.
- Value of the frame matrix change from negative to positive along x-axis indicates a RIGHT gesture.
- Value of the frame matrix change from positive to negative along x-axis indicates a LEFT gesture.

The above proposed algorithm has led us to easy detection of air swipe gestures made by the user in front of the mobile device using just front camera. Upon detection of any specified gesture the application program creates a toast specifying the gesture detected. This approach is able to recognize the given set of four gestures with both recall and precision over 96%.

V. Conclusion

In this research, the proposed approach is to recognize dynamic gestures on the resource limited gadgets like smartphone. The application of OpenCV for matrix allocation, with real-time computation is the key focus of this research effort. The algorithm does not imply with any physical change thus, needs only images from the front camera which are easily available on such gadgets. This helps in making organic gestures on smartphone gadgets available and accessible to an extended user base. We employed a standard matrix allocation algorithm to give an x-y coordinates frames in which the motion feature was used to extract the air-swipe gestures. We achieved 96% accuracy for recognition of dynamic swipe gestures in air. User can control, interact or command their android device using air-swipe gestures instead of physical interaction with the help of this application.

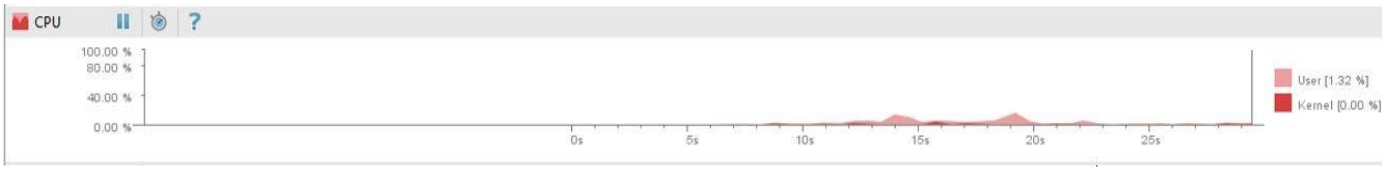

Fig 5 classifies the minimal loss in accuracy for CPU

VI. Future Work

This prompt approach though seems to be user friendly and viable to the conventional input methods but is moderately less sturdy in assimilation stage. A design of an independent

gesture vocabulary could be another aspect of related development that is more ways of Gesture recognition from various other human actions will take place instead of just air-swipe gestures. The presented architecture framework could be autonomous of the application scope and can be very helpful in controlling or commanding various applications like games and also applications like making a gesture to click a picture and sliding between the images in gallery, connecting and disconnecting calls, playing songs and accessing various applications.

Ambient light is also an issue with all vision systems. We manifested that the application can confront a various types of ordinary lighting which is found indoor and outdoor areas. On the other hand, rapid shifts in light like while leaving or entering a closed area or switching off or on light sources are still some issues that need to be resolved. A future enhancement to this can be like switching on the flash light as and when there is an issue of ambient light.

REFERENCES

[1] Song, J.,Soros, G. , Pece, F. , Fanello, S. , Izadi, S. , Keskin , C. &Hilliges , O. (2014). In-air Gestures Around Unmodified Mobile Devices. *Proc.ACM UIST 2014.*

[2] Joshi, T. J. , Kumar, S. , Tarapore, N. Z. &Mohile, V. (2015). Static Hand Gesture Recognition using an Android Device.*International Journal of Computer Applications(0975 – 8887) Volume 120, No. 21.*

[3] Rautaray, S. S. &Agrawal, A. (2012). Real Time Hand Gesture Recognition System for Dynamic Applications. *International Journal of UbiComp (IJU), Vol.3, No.1.*

[4] Bayazit, M. , Beil, A. C. &Mori, G. (2009). Real-time Motion-based Gesture Recognition using the GPU. *Proceedings of the IAPR Conference on Machine Vision Applications (MVA'09).*

[5] Shaikh, S. , Gupta, R. , Shaikh, I.&Borade, J. (2016).Hand Gesture Recognition using OpenCV. *International Journal of Avanced Research in Computer and Communication Engineering Volume 5.*

[6] Phansalkar, N. , Kumthekar, N. , Mulay, M. , Khedkar, S. &Shinde, G.R. (2014). Air Gesture Library for Android using Camera. *International Journal of Engineering Research & Technology (IJERT), ISSN:2278-0181,Vol.3,Issue 2.*

[7] OpenCV: About OpenCV. (n.d.).Retrieved from http://opencv.org/about.html/

[8] OpenCV: Camera Calibration with OpenCV. (n.d.). Retrieved fromhttp://docs.opencv.org/2.4/doc/tutorials/calib3d/camera_calibration/camera_calibration.html/

[9] Hellman, E. (2013). Get Started with OpenCV on Android. Retrieved from http://developer.sonymobile.com/knowledge-base/tutorials/android_tutorial/get-started-with-opencv-on-android/

[10] HCI: Introduction to Human Computer Interaction. (n.d.). Retrieved from http://www.hcibib.org/

[11] OpenCV:Android platform support in OpenCV. (n.d.). Retrieved from http://opencv.org/platforms/android.html/

[12] Controlling the camera. (n.d.). Retrieved from https://developer.android.com/training/camera/cameradirect.html/

[13] OpenCV Tutorial 1 : Camera Preview. (n.d.). Retrieved from https://docs.nvidia.com/gameworks/content/technologies/mobile/opencv_tutorial_camera_preview.htm/

[14] Manual OpenCV4 Android SDK setup. (n.d.). Retrieved from http://docs.opencv.org/2.4/doc/tutorials/introduction/android_binary_package/O4A_SDK.html

[15] Dixit, V. &Agrawal, A. (2015). Real-Time Hand Detection & Tracking for Dynamic Gesture Recognition. *I.J. Intelligent Systtems and Applications*.

[16] Patil, T. B. , Jain, A. , Sawant, S. C. , Bhattacharyya, D. & Kim, H.J. (2016). Virtual Interactive Hand Gestures Recognition System in Real-Time Environments. *International Journal of Database Theory and Application, Vol.9, No.7 (2016),* pp.39-50.