Smart Media Player with Eye Blinking Control

Thajun Najaah M.A.

2020

Declaration

The thesis is my original work and has not been submitted previously for a degree at this or any other university/institute.

To the best of my knowledge it does not contain any material published or written by another person, except as acknowledged in the text.

Student Name: M.A.Thajun Najaah

Registration Number: 2017/MIT/080

Index Number: 17550803

_____

Signature: Date: 2020.11.18

This is to certify that this thesis is based on the work of

Mr./Ms. _____

under my supervision. The thesis has been prepared according to the format stipulated and is of acceptable standard.

Certified by:

Supervisor Name: Dr. M.G.N.A.S.Fernando

_____

Signature: Date:

Smart Media Player with Eye Blinking Control

A dissertation submitted for the Degree of Master of Information Technology

Thajun Najaah M.A

University of Colombo School of Computing
2020

Abstract

When we are playing a video on our Laptop or Mobile or PC and if someone calls us, we have to get away from the device or simply look away from it and sometimes we may sleep in the middle while watching a video and we have to drag back to see the video where we have left. As a solution to overcome this issue we can build a smart media player which can be able to play or pause the media stream while the human face is detecting in front of the screen and while the persons eyes are opening. It detects the Human face and Eyes and for the time it detecting an open eye/eyes video will be play else video will be pause automatically. The system runs continuously while monitoring whether a human eye open or not and the player starts to running as soon as it is detecting the eyes again and which keeps running as long as it is detecting the eyes without the interruption. This can be accomplished by using the web camera which is on the top of the computer. The player pauses as soon as the human eyes not completely seen. If there is no human face detected then the player will be stopped immediately. We are using Image Processing techniques and Open CV haar-cascade algorithms and haar- features for performing face and eye detection in real time. In addition to controlling media player by eye blinking the player offers all the basic features of a normal media player and it is generating a statistics report with performance analysis of the Smart Media Player. As per the performance analysis system has achieved ~6 frames per second and ~6 faces per second.

Key Words: Media Player, Face Detection, Eye Detection, Web Camera, Image Processing, Haar-cascade, Algorithms, Haar-features, Accuracy, Open CV

Acknowledgements

I owe a great many thanks to the people who have helped and supported me during the Implementation of Smart Media Player.

I sincerely convey my gratitude to my supervisor, Dr. M.G.N.A.S. Fernando for the everlasting inspirations, valuable guidance, constant help and encouragements at each steps of my work. I am really blessed to have a supervisor who cared so much about my work, and who replied my questions and queries so promptly.

I would also thank to my Faculty University of Colombo School of computing and all academic and non-academic staffs without whom this project would be a distant reality. I also express my heartful thanks to my family and well-wishers for their support to accomplish this task.

Table of Contents

Contents

List of Figure

Chapter 1

Introduction

1.1 Background & Motivation

Communications between intelligent devices and Human Computer Interactions (HCI) are being a dynamic field of study from the last decades. Each and every person depending on their computers to perform their tasks. To get the efficient use of these devices we need computer applications to communicate more and more.

Media player is also that kind of an application which is used to play multimedia files. Media players contain standard control icons such as play, pause, next, previous, stop, fast and back forward controls, volume, time controls and etc. Media players can design to function with different features and for different goals such as some media players can be control with human body gestures or emotions or voice commands.

Nowadays people are very busy with their daily activities. However, they managed to watch videos as they find time or even while they are working. Even though most of the housewives watch dramas and movies as a daily routine while they are busy with their work at home.
Also, people spend time to watch video in the meantime of vehicles move. When we play a video on our laptops and if somebody calls us, we simply look away from it, or get away from the device, due to this we will lose some part of video. Then we need to drag back to see the video from the point we have left. As a solution for this issue we can develop a media player that can detects human eye blinking and while it is detecting face and open eyes, the video will be play else video will get pause automatically. The player starts playing again as soon as it detects the open eye and the face appears in front of the camera. This can be done using the laptop web camera. As long as the camera detects an open eye the player keeps on playing video.

The other similar researches talking only about the players which are controllable based on the face detection only. But the problem is, sometimes the user can appear in front of the camera and put the eyes down or aside when they miss their concentration.

1.2 Motivation

Video players does not pause itself even though the user suddenly get into another work. So, they shall bit confuse in which frame they have ended watching. For instance, if a mother suddenly turns to look after the child, she will miss the frame from where she left. Therefore,

there is a possibility to struggle on re-locating. Likewise, if a student studies or a housewife cook by looking at a video, he/she should pause, write and vice versa. This makes people tired of doing it again and again. Moreover, it causes energy wastage for both human and electronic devices. Sometimes, a person may sleep in the middle while watching a video. This also can cause to waste of Mobile data and Loss of charge. Because, screen does not go sleeping mode until the video pause. People does not often consider on what they were doing when they meet an important work suddenly. In such situations, they would feel convenient if there is an option for auto pause and auto play.

1.3 Objectives

The problem domain leads to find a solution based on the movements of a user. In general, Human eye focus on what they see over. Monitoring and capturing the motion of human eye while a video be opened, can help to detect what they wish to do; pause or play. Thereby, the solution for the problem domain is to develop a smart video player with eye detection by image processing to pause or play automatically.

This innovative option can further be extended till making the option customized to operate auto or manual. Following objectives will be achieved for complete the development:

- Exploring the features of existing video players and explore to design with basic features of a player
- Exploring about image processing and understanding how to detect objects over webcam & finding a suitable way or technology to implement face detection and capturing eye motion
- Designing a system architecture for capturing eye motion and operations to be done in the media player with the captured motion
- Exploring about image processing and Understanding how to detect eye movement over webcam and detecting how the algorithm varies while angle and distance change.

The solution for the problem domain would be throw concern on implementing motion of human eye based on image processing. System can detect if there is human eye in front of web front camera. It detects the eyes when it is open. Moreover, the system contain options like playing video from browsing directory, Play, pause more like designed in a simple video player. This system will focus only on the innovative idea which gives solution for the problem. But the compulsory options of the player also developed. Limitations of the scope would be in general cases like using spectacles will not bother support or response correctly or will not be considered as the system designed for. This would always be an exception case of operator's eye to be detected by the system. And the system may not correctly response for the people who have disabilities in their eye. The user may operate it from a long distance or watch from

far away from the screen which is not capture human eye for doing the operation. So, the system will have limitation for those who operate from a long distance.

In this chapter, background information, identified problem and the solution for that problem were discussed. Chapter 2 explains the review on similar systems with appropriateness. Chapter 3 explains the design of the solution in the methodological approach. Whereas, the chapter 4 draws an evaluation on the final outcomes and the achievement of my research and elaborates about the positive, negative and future implementation. Chapter 5 gives a conclusion about the result from the project.

Chapter 2

Review of Similar Systems and Technologies

2.1 Introduction

In this chapter we are going to discuss about the current issues in existing video players. Other Approaches taken to overcome the problem, the differences between those solutions and Specialties of our proposed solution are also discussed here.

2.2 Other approaches

Piotar Dalka [1] introduces the Multimodal Human Computer Interface for severely disabled and paralyzed people which can track the lip movement and doing lip gesture recognition using Artificial Neural Network for lip gesture recognition and Fuzzy clustering used for lip shape segmentation. They used web cameras to capture the faces. Marcelo Archajo Jos [2] presents an innovative lip control system which is specially focused on the people who got affected by tetraplegia. They design a system by using lower lip potentials as control commands for an input device. This is considering as an interesting approach in the development of assistive technologies. K. Meenakshi [5] developed a software interface to use mouth gestures to move cursor on the screen and to perform mouse click operations. Disha H, Shubhangi T, Snehal P [7] have proposed a solution for handicaps or those who unable to operate computer systems. The proposed solutions they came up with is consists of a five-step algorithm to calculate the direction of eyes to control computer system through a single web camera. Margrit Betke [3] introduces "Camera Mouse" system to give computer access for the physically disabled people. In here the system tracks the persons movements captured through a video camera and translate them as the mouse pointers movement on the screen. Yo-Jen Tu [4] implements a system to control by head poses and hand gestures of human. All of these systems are designed for physically disabled peoples they need the user in front of the system or user interaction and they don't have the facility to automatically control the system if the user gets away from the system or user changed his focus to another work.

Mukesh V et al [9] implemented a system architecture for controlling a media player based on the face detection using webcam. Also, they are doing the same research, but they have used to detect face and motion and control the media player. S.V. Viraktamath et al [10] they are dealing with detection and face tracking using Open CV and analyzing the captured image and stored the image in database. This is used to check the location of the human face and size, position. Knowledge base approach, Integration Based Approach with multiple functionalities and Static Approach are used to perform face detection. Deepika Nadar, Suma Acharya, Sarvesh Parab and Akhil karkera [6] have created a look base media player which could be operated based on face tracking with the help of web camera. Hand gestures plays a crucial

role in increasing and reducing the volume. They also have used Haar Cascades algorithm like most of the other researchers used. In this system they only considering the Face and eyes detection without considering the eye blinking states. (Eyes Open / Eyes Closed). In our system in addition to face and eye detection we are considering the eye blinking and allows to play the video if the eyes open only.

Dr. Abhijit Banubakode et al [20] done a research and implements a Media player which can be control by human emotions. They introduce a new technology to identify human emotions called, "Emotion Sensory World of Blue Eyes Technology". Moreover, they used Eye detection, face detection, lip detection, image processing techniques to identify the facial parts where can be observe emotions such as happy, sad, hate, angry, thinking etc. Model-Based Eye Detection and Animation [13] implements a system to detect the human eye motion from a video stream. In eye motion estimation they collect the eye location information in each consecutive frames of the video and synthesizing these eye motion estimations into a virtual character and makes the virtual face moves the eyes in the same way than the human face. Edge detection algorithms used for eye detection. Region Segmentation and various image preprocessing techniques applied to the extracted eye features to extract the iris center. This study helps us to get an idea about a system capable of face tracking, eye tracking, eye blinking detection, and eye features extraction.

Magee et al [12] proposed a system to control the computer application by tracking eye motion direction. They used an USB camera for capturing the human face. They used multiscale template correlation to capture face. By using the camera, they detect the left and right sides of the faces and they exploit the symmetry between the both eyes of the user. Zafer Savas et al [14] presents a system to detect the human eyes from videos in real time. In here CAMSHIFT algorithm used to track the outline of an object. This only considers the objects with irregular shaped and can change the shapes and sizes during the process. Several steps of image pre-processing applied to the face area to get efficient results during the searching process. Then by applying the geometrical properties of the face both Left and Right eyes get determined locate each eye separately. The detected left right eyes and location info used as a training eye sets to the eye area detection algorithms. At the end of the process by associating the geometrical locations of human with the target area, they extracted eye gaze information. These systems only considering the Eye area detection not the eye blinking states.

Siddharth Swarup Rautaray, Anupam Agrawal [8] implemented a vision-based input device for using computer vision and gesture recognition for controlling VLC media player. They used K Nearest Neighbors algorithm for gesture recognition. Uma Annamalai et al [21] implemented a system for Controlling Multimedia Applications Using Hand Gesture Recognition. This system acts as an intermediary between human machines Interaction. They used various hand gestures to control media player operations such as up, down, left, right etc. It has an application running on the backend to capturing and storing various hand gesture inputs. And they are

using various image pre-processing techniques to remove the noise in images. Mahdi Abbasi, Mohamed R.Khosravi [11] they are processing with big datasets of eye videos which requires fast and robust algorithms for eye tracking and to predict the eye positions of human in consecutive frames of the videos. In their research they are proposing a genetic algorithm to increase the eye detection rate at the sampling step of particle filtering. Xuebai Zhang et al [15] developed a system called, "VLEYE" for assisting video lectures. It is based on eye tracking. This System has 3 major modules, those are Eye Movement Recorder, Video Analyzer, and Dynamic Area of Interest Module. The main task of the system is to allow data gathering in Dynamic Areas of Interest and integrate it with the eye movement data. Ince and Yang [16] done a research to detect exact eye blobs by applying differential geometry within sub-pixel level of the captured web camera images of eye features and which helps them to found out a low-cost eye tracking system with blinking and drowsiness detection. This approach used in our system for eye blink detection. Hossain Mahbub et al [32] proposed a system to get the accurate eye central tracker using web camera to detect the eye ball direction. It is mainly focusing on low. Power consuming devices. They used the algorithm to calculate the mean of gradient. Vector. But it failed to detect eye states such as open/closed.

Furkan Ince et al [33] introduced an algorithm to detect the eye blobs. It is focusing on fast and detecting the sub pixel level in a precise way. This is existing in OpenCV library. It is providing efficient performance for lower cost eye tracking applications and blink detections. This algorithm used to track the eye ball location info with the central coordinates.Ruslana Makovetsky and Gayane [22] implemented a media player with face detection and tracking with web camera. In this system, the player will start / keep playing if the human face is totally visible to the camera if he turns it will pause and if he gets away from the screen it will stop. In this system even if the user not looking the screen or he is sleeping while face is in front of the screen video will keep playing continuously without pausing. Our proposed solution can avoid the drawbacks by considering the eyes and eye blink states and it allows the player to keep play if there is at least one open human eye detected. Otherwise it will be paused automatically.

Gautami Shingan et al [31] implemented a media player which is controlled by colors. This system used web camera to capture the color cards and image processing techniques and OpenCV libraries to detect and recognize the various colors to control the basic player operations. They used Java FX for player implementation. They used black card for Close, red for Pause, Blue for Play and Play Next Song and Green for Previous Song. They applied image capturing, image sharping, color detection, color recognition algorithms in order to get the task accomplished.

By a deep analysis of existing media players most of them are focusing only face detections or hand gesture recognition or emotional detection and color detection, voice-based approaches. Considering the faces cannot be a proper solution because most of us using media players while

on the bed before sleeping or sleeping in front of laptop screen or focusing on other works or eyes will be focusing other places even though if we are in front of the laptop. Due to those issues we came across with a solution to use both face and open eye detection to play control the media player. For our system implementation we are using the image capturing algorithms from Gautami Shingan et al [31].

Dr.Abhijit et al [10] implemented a media player which operates by human emotions. In their application they used face detection algorithm to extract facial features implemented by Viola and Johnes which has a high detection rate and most suitable for real time applications. We used the same algorithms for face detection which is inbuild with OpenCV haar-cascade classifiers. Detecting Human Face can be obtaining from several ways. S.V. Viraktamath et al [10] used knowledge based and static approaches with integration approach in their system. Which has incorporated with several functionalities to achieve the task and time consuming which is not suitable for a real time player. Sanyam Garg [26] suggested a method to use Voila Johnes algorithm for only eye regions of the consecutive frames using pre trained haar cascade classifiers for detecting eye states. He found a solution as using a common classifier to detect the eye regions whether it is open /closed. And then using the specific and respective classifiers for left and right eyes which can be detect only the open eyes. We are using this method for open eye detection in our smart media player.

Chapter 3

Methodology and Implementation

3.1 Introduction

According to previous literature review clearly indicates that we can convert the human body features to create a better Human Computer Interaction to communicate with the devices in an efficient manner. This chapter included the modules and components and how each module interacted with each other and the system implementation details of Smart Media Player.

3.2 System Analysis and Design

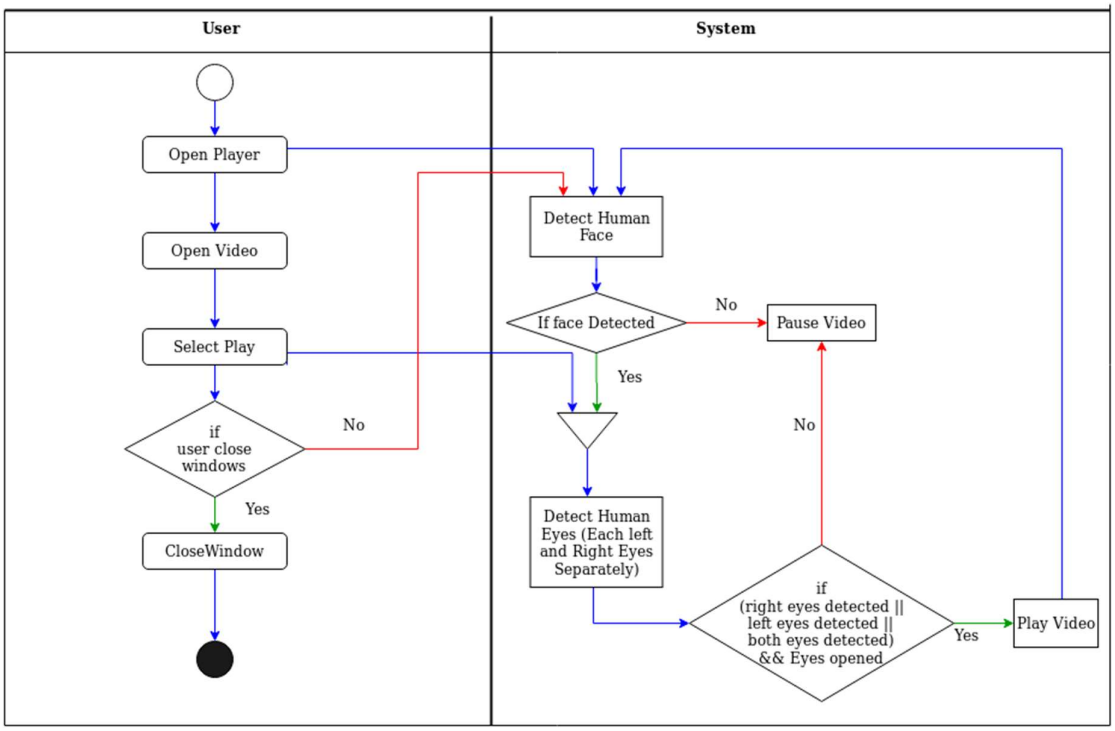

Figure 3.1: Activity Diagram for Smart Media Player

This player contains all the basic functionalities and features of the existing media player and more over it implements with the logic of if at least one human face is detected and one of left or right eyes or both eyes were detected and if the eyes are open, player start to play the movie or if it is already started then keeps it playing until the open eye detected. If eyes were closed then the video has to pause. Otherwise, if there is no human face detected then the video get stops. The above Figure 3.1 shows the activity diagram of the Smart Media Player controlled by face and eye states detection.

Eye state recognition is collaborated with the face detection from the input video frame captured by web camera on real time. This system implementation is divided into following categories. Those are,

Set the field of view of the web camera: in our system implementation we are using the laptop web camera for capturing the images. Webcams are fixed focus imaging systems where we don't have any optical component to transformation (aperture, lenses) etc. For web cameras field of view is normally stated relative to sensor-size, sensor shape and the focal length of the lens. A wide sensor can have a larger field of view where as a square sensor with the same area will have a smaller field of view [29]. The standard focal distance for fixed focus web cam manufacturers is nearly 1.5 meters distance from the web camera. This value guarantees that a large enough distance range falls under suitable focus. [28] Field of view implies the region of a real-world scene that is visible for the imaging medium. It can be determined by few ways. It can be measured horizontally, vertically, or diagonally. Following Figure 3.2 shows the relationship between focal length and the Field of View.

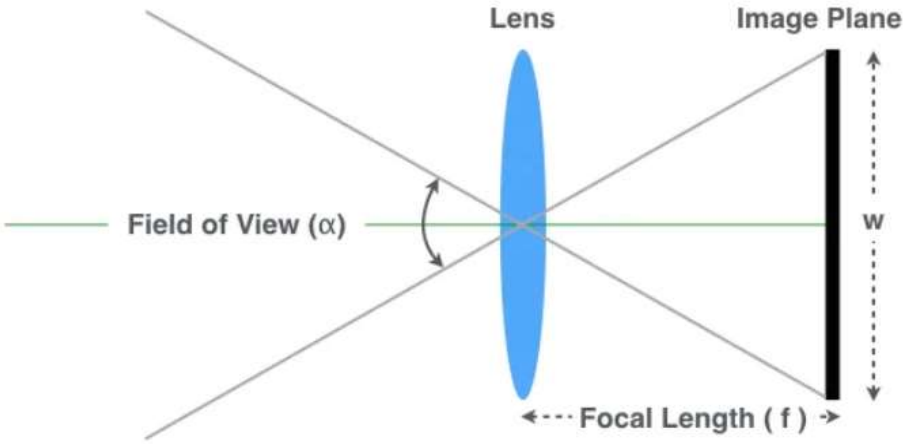

Figure 3. 2 - Relationship between focal length and the Field of View with the size of the sensor

The relationship between the field of view (**α**) and image plane dimension(**w**) are related with the focal length(**f**) becoming like this, **f = w/2 cot (α/2).** Most of the time horizontal field of view for the web cameras is between 50 - 70 degrees. Plugging the range of field of view (**α**) the focal length becomes the value between the value related with the width of image plan (**w**) and the rage becoming like this, **0.7w <= f <= w**. But it gives the crude approximation and not 100% accurate value for the focal length. But it gives the rule of thumb and it is useful even we are using precise calibration. For example, if the web camera resolution is **1280×720**, the

focal length will between1100 – 1300. OpenCV provides the way to set the Horizontal, Vertical resolutions to set the focus manually while stop the auto focus of the web camera [30].

Following lines showing the code segment for setting camera resolution to 1280×720 for getting a better view range throw web camera.

```
VideoCapture videoDevice = new VideoCapture(0) # Generate camera object
videoDevice.set (CV_CAP_PROP_SETTINGS, 1) # to set autofocus off
videoDevice.set (3, 1280) # set the Horizontal resolution
videoDevice.set (4, 720) # set the Vertical resolution
```

Capturing the video from web camera: We are using OpenCV [18] java libraries for the purpose. We have created a thread running on backend of the player which continuously capturing image using web camera until the player get closed. Web camera captures around 5 -10 images per second.

Pre-processing: Before passing each consequent frame to face detection algorithm image pre-processing steps will be applied to each frame to make the algorithm more precise. According to below Figure 3.3 First, Original image(a) is transformed from RGB colour format to gray scale(b) and then, image resizing occurred by applying pixels sampling to increase the algorithm work faster(c), According to Ruslana Makovetsky et al [23] factor of 5 applied to the original image to scale down.

At the final step of pre-processing histogram enhancement applied to increase the precision of algorithm by normalizing the brightness and increasing the contrast. As a result of this step, the black and white structures of the frame become more significant (d). Following Figure 3.3 shows each steps of pre-processing.

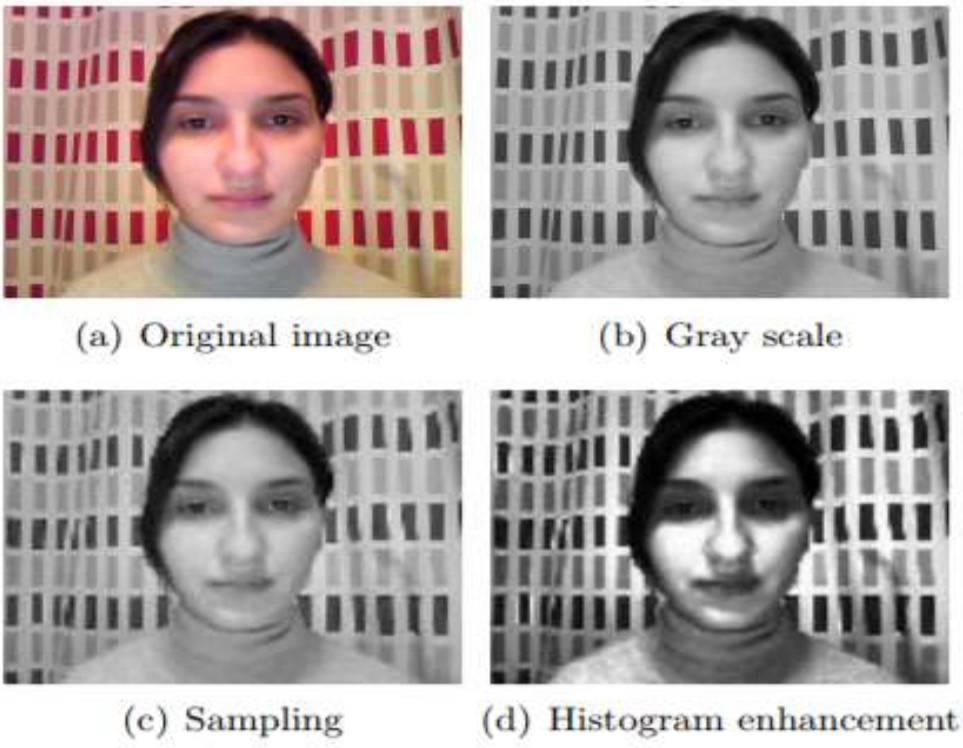

Figure 3.3 – Image Pre-Processing Steps

Face Detection: After pre-processing steps, preprocessed image forwarded to face detection process, where various positions of the face are detected. Viola and Johnes algorithm used to detect faces. It is the high detection rate and more competitive algorithm to detect objects in real time. It is using haar cascades for face detection. Harr cascade classifiers for frontal face and profile face are available in OpenCV [18] library. Basically, this classifiers pre trained with positive and negative images of human faces. Then we can simply use this feature for extracting faces from images.

Figure 3.4 demonstrates the four different types of rectangle features used by the haar classifiers. In that they are using the pixel intensities instead of using direct pixels. According to that each single feature value is calculated by subtracting the sum of pixels of white rectangle from sum of pixels of black rectangle [25]. In Figure 3.4, A, B illustrates two rectangle features, C shows three rectangle and D shows the four-rectangle feature.

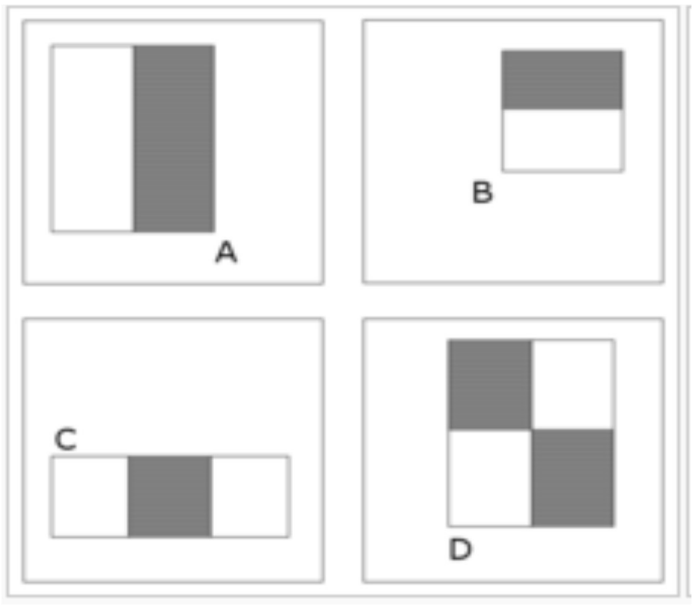

Figure 3.4 - Haar extraction rectangle features

However, before using the classifier the image is mirrored and then the haar-cascade frontal face classifier is applied. If the face is found the coordinates of box containing the image are flipped back to correspond to actual image [25].

Following Figure 3.5 shows how does the haar features working. It focusing on the eye region in first image and that is often darker than the nose region and cheeks. [25]

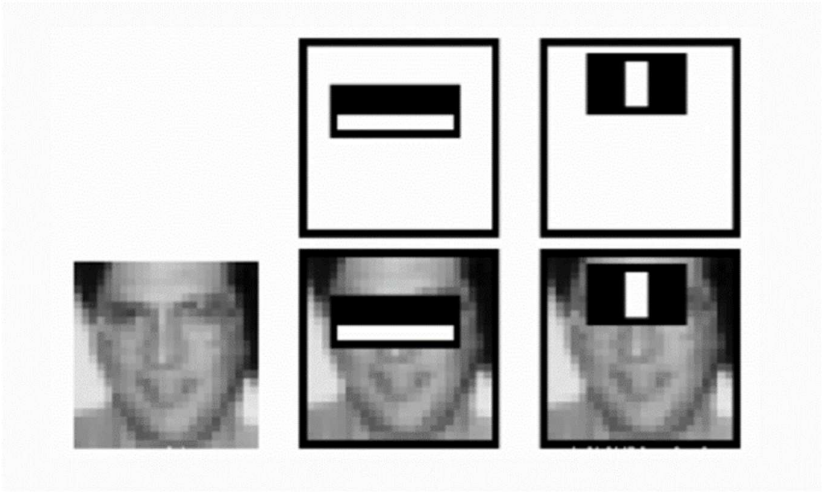

Figure 3.5 - Examples of harr extraction on human face

In out code implementation we are using **detectMultiscale** module of OpenCV. Which creates a rectangle region around the detected face and the coordinates of the region used to detect face. Let we assume it as x, y, w, h. Most important code parameters of this module are,

- **scaleFactor**: This indicates the size to be reduced from image size at each image scale. It has values of x,y (x,y are the arbitrary values to be set). A lower value for scale factor leads to downscaling.
- **minNeighbors:** It is used to initialized the number of "neighbors" candidate rectangles. If this value is high than the quality of detected image will increase but no of detection will decrease.
- **minSize**: This param used to set the minimum object size. If our object size is small in the captured image than we need to set a lower value for this. By default (30,30) used [25].

In our smart media player, we are using frontal face classifiers to detect front faces only. Because we are allowing to play the video files if the user looks the screen only.

Eye Detection: It can be done via extracting the eye region from the detected face. So, it is always following the face detection and it cannot perform it individually. It is performed with the use of haar cascades pre learned classifiers in OpenCV. It is very difficult detect the blinking eye whether it is open or closed and which algorithm to be used for that. By an extensive literature review we found that we have to use separate cascade detectors for the left and right eyes to achieve this goal. Those are,

- haarcascade_eye_tree_eyeglasses.xml used to detect both open and close states.
- haarcascade_lefteye_2splits.xml used to detect left eye while its open only
- haarcascade_righteye_2splits.xml used to detect right eye while it is open only

In smart media player first, we are applying the common eye classifier on the detected face in the region where maybe containing eyes. It will detect if eyes existing unaware of the eye states. If the result for common eye classifier success then the separate detectors of left and right eyes will be executed to find out whether the eyes are open. Left eye classifier applied to the right side of the detected face and Right eye classifier will be applied to the Left side of the detected face. Eye detection is happening on the actual image size while face detection is happening on

the reduced image size [26]. If we get an eye detection result for both common and at least one specific detector then the eyes are open and the player will start to play or keep playing. If we can detect an eye with the common classifier and not detect with specific classifiers that will consider the eye state as closed and player will be paused. If none of them find eye it will result no eyes found and player will pause. The following Figure 3.6 shows how does the flow of haar-cascade algorithm works for human eye detection.

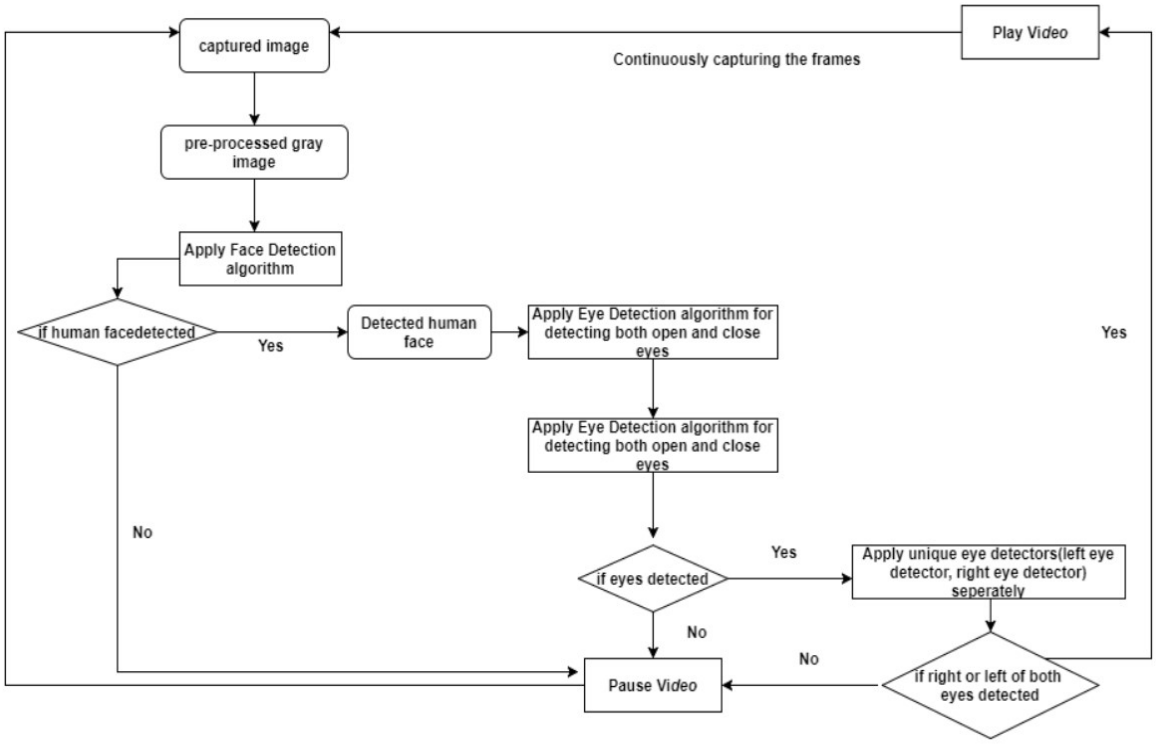

Figure 3.6- Haar-Cascade Algorithm for Eye Detection

Media Player: JAVAFX Framework for media player implementation. The player using following logic. If at least one face and one open eye detected player plays the movie. If eye state closed and faces are detected player will be paused. If there are no faces detected player stops.

More than that player has implemented with all the basic components and functionalities like open file, playlist, close, volume, time bar, previous, next, fast, slow, normal speed, move

forward, move backward operations. Moreover, to basic player operations we are implementing the time tracker as well to get videos total duration and time left.

JavaFX Designer called Scene Builder used to build the complex GUI for this smart video player. The Code implementations for the Media player is included in **Appendix B**. Following Figure 3.7 shows the player GUI while running a video [1]

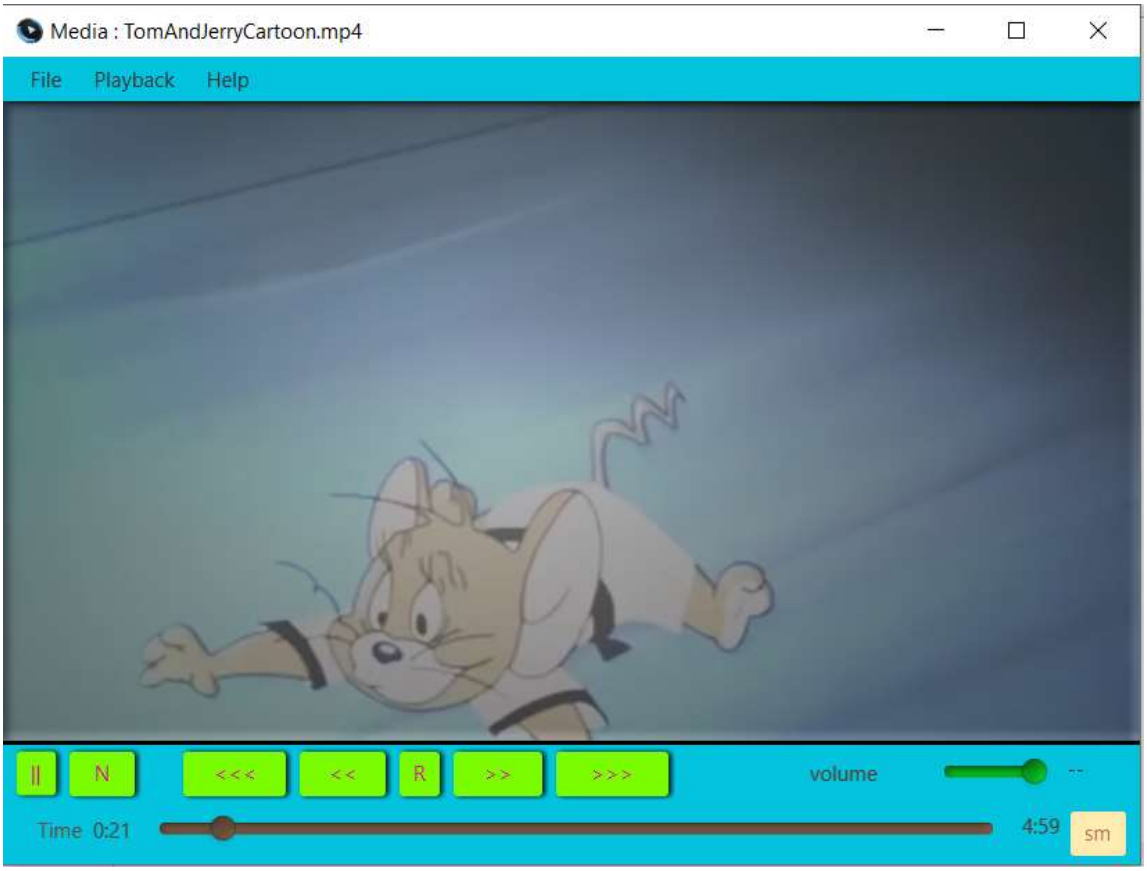

Figure 3.7 - GUI of Smart Media Player

Chapter 4

System Evaluation

4.1 Introduction

This Chapter describes statistics of the algorithms which are used for the face detection and eye detection and the frame rate and the summarization of the advantages and disadvantages of our Smart Media Player while comparing with the existing media players and the perceptional end user evaluation for quality analysis of the player and functionality testing and accuracy of the work carried out.

4.1. Experiments and results

System performance analysis done by running the thread for continuously around 60 minutes (60*60 = 3600 seconds). Following Figure 4.1 illustrates the final outcome of number of frames detected on that duration and frame rate per second and no of different human faces detected and total numbers of human face detected during the period and face detection rate per second and number of opened left eyes and the percentage of left eyes only against the total number of faces and the number of opened right eyes and the percentage of right eyes only against the total numbers of human faces and the total number of single eyes detected against the total number of human faces and also total number of both opened left and right eyes detected and the percentage of both eyes against total numbers of faces.

```
------------Statistics of Frames---------------
Application Running Duration in milli seconds: 3601665
Application Running Duration in seconds: 3601
Total Frames: 21818
Frame rate per seconds:  6.059frames per second

------------Statistics of Faces---------------
Number of Different Human Faces Detected: 1
Number of faces Detected from whole frames: 21479
Face detection rate per second: 5.965

------------Statistics of Single Eyes---------------
LeftEyes only detected: 1976
LeftEyes only detected per faces: 9.2%
RightEyes only detected: 2888
RightEyes only detected per faces: 13.446%
Single eyes detected per faces: 22.645%

------------Statistics of Both Eyes-----------------
Both eyes detected: 12711
Both Eyes detected per faces: 59.179%
```

Figure 4.1- Statistical summary of the algorithm's evaluation.

As per the final outcome,

- Image capturing rate is ~6 (6.059) frames per second. Following equation used to calculate this value.

  Image Capturing Rate = Total number of frames detected / Time

  Image Capturing Rate = 21818 / 3601 = 6.059 (~6 frames per second)

- Face Detection rate is ~6 (5.965) faces per second. Following equation used to calculate this value

  Face Detection Rate = Total number of faces detected / Time

  Face Detection Rate = 21479 / 3601 = 5.965 (~6 faces per second)

Face Detection Rate with Total Frames is 98.45%. Following equation used to calculate this value

Faces Detected per frame = (Total number of faces detected / Total number of frames detected) * 100

Faces Detection Rate per Frame = (21479 / 21818) * 100 = 98.45%

- Single Eyes Per face is 22.645%. Following equation used to calculate this value.

  Singl Eyes per Face = ((Total number of left eyes only detected + Total number of right eyes only detected) / Total number of faces detected)
  * 100

  Single Eyes per Face = ((1976 +2888) / 21479) * 100 = 22.645%

- Both Eyes Per face is 59.179%. Following equation used to calculate this value.

  Both Eyes per Face = (Total number of both left and right eyes detected at a time/ Total number of faces detected) * 100

  Both Eyes per Face = (12711 / 21479) * 100 = 59.179%

- Open Eyes per face is 81.82%, following equation used to calculate this value.

Open Eye per Face = Total Number of Open Eyes Detected / Time

Open Eye per Face = ((Total number of single eyes only detected + Total number of both left and right eyes detected) / Total number of faces detected) * 100

Open Eye per Face = ((1976 + 2888 +12711) / 21479) * 100 = 81.82%

4.2 Advantages of Smart Media Player

- User friendly media player
- Avoiding the missing any part of video due to other works or sleep
- Improved and Explicit Human Computer Interaction
- Avoid dragging back towards where they have missed.
- Helps to save users' time
- Helps to save electricity
- Having all the basic functional requirements as existing media players

4.3 Disadvantages of Smart Media Player

- Face detection needs full face of the user.
- Java FX not supporting for all video formats like MKV, AVI

4.4 Perceptional System Analysis of Smart Media Player

Perceptional system analysis of smart media player carried out through user survey and the system evaluated based on how it is responses to different users. There are 33 users participated in this survey. Users given with the video player directly and asked to mark their observation for rating the system.

The parameters for observations are as follows;

- Moving head away from the vision of web front camera and observe whether it pauses or not.
- Entering again to the vision of the web front camera to check whether the video plays again.
- Closing eyes in front of the vision and observe whether it is pauses or not.

- Opening eyes and observe whether it is playing the video again.
- Testing the system functionality while person move back and forward (variability in scale)
- Shaking the users head (in-plane rotations)
- Possible obstructions (person performing actions such as drinking a coffee, partially cover his face by hands, etc.)

Appendix C shows the sample user survey carried out for perceptional quality analysis.

Following Figure 4.2 shows the standard deviation of user's response for usability of the system.

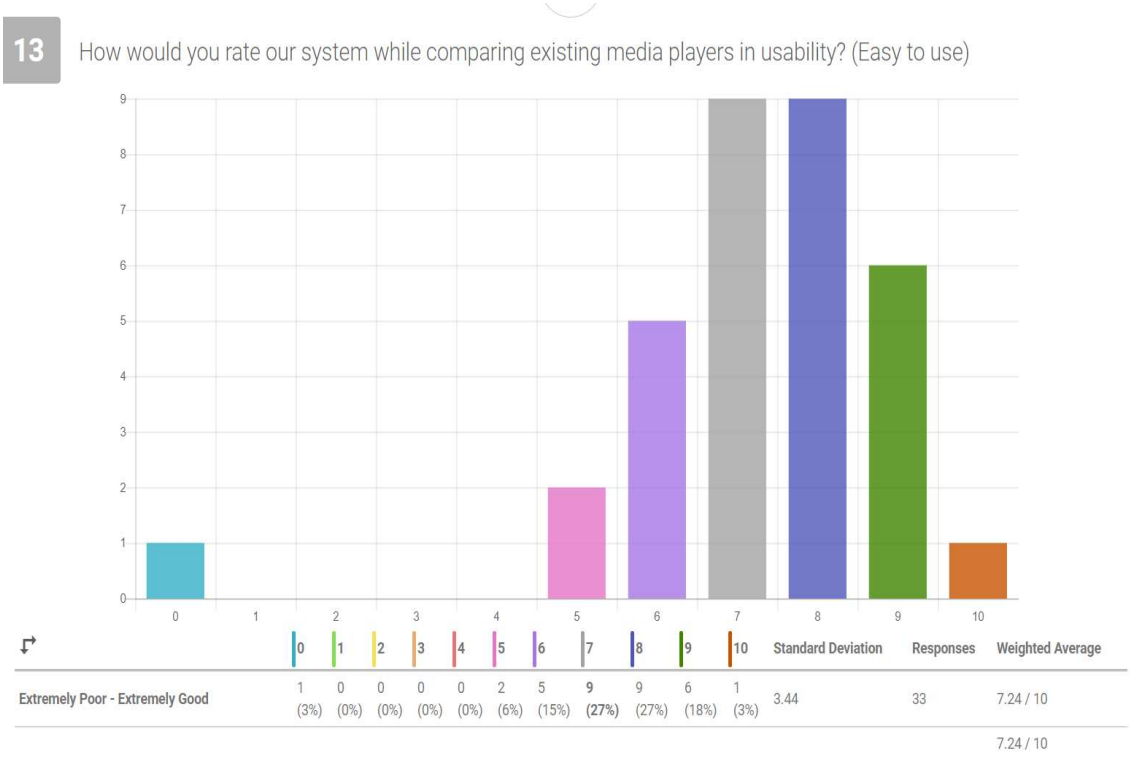

Figure 4.2-User feedback for the usability of smart media player

Following Figure 4.3 shows the users feedback regarding the main functionalities (Eye detection-based player control) of media player

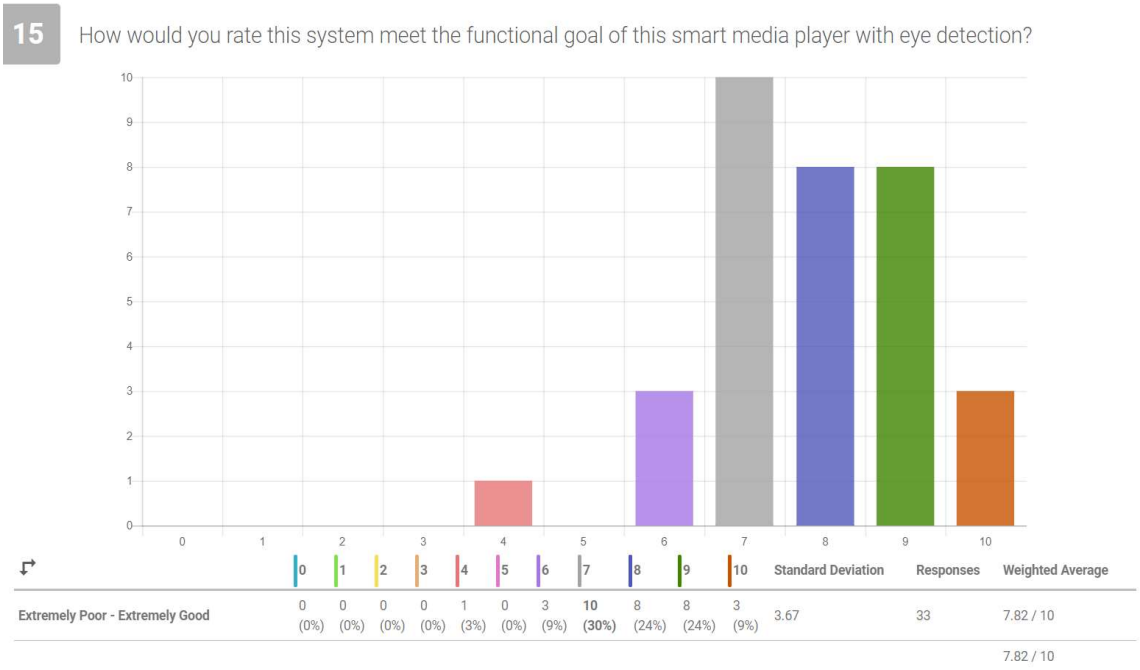

Figure 4.3-User feedback for the functional achievement of the system

These are the snapshots of media player. Following Figure 4.4 shows the screen shots of media player and how does the face detection and eye detection works.

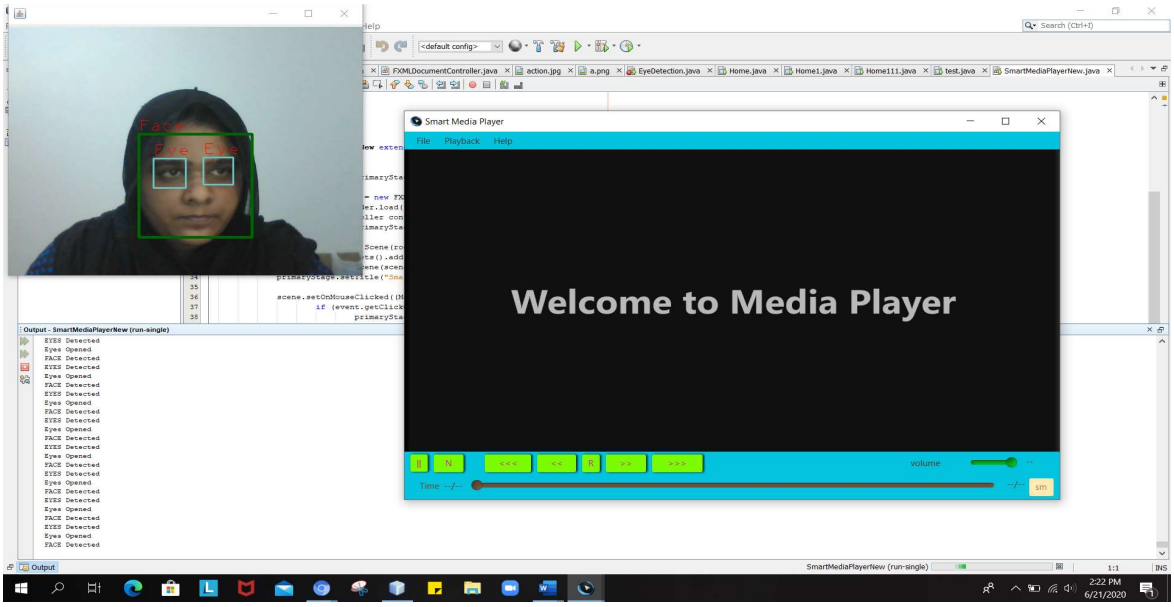

Figure 4.4-Screen shot of media player while starting application

Following Figure 4.5 shows the screen capture while opening media file.

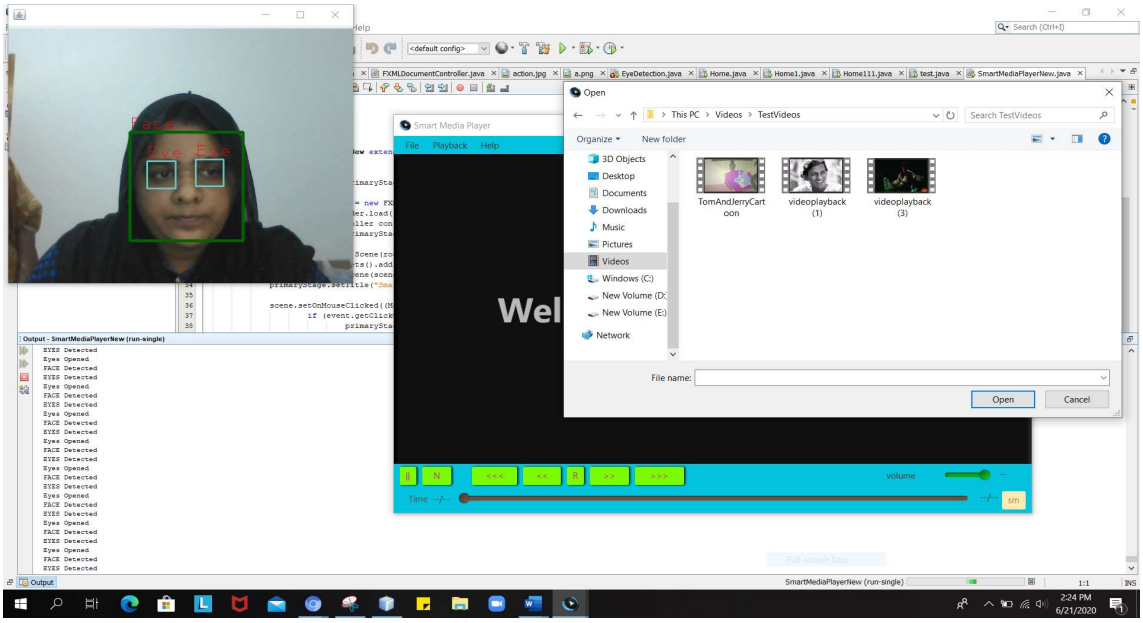

Figure 4.5-Screen shot of media player while choosing video file

Following Figure 4.6 shows the player playing while face and open eyes detection.

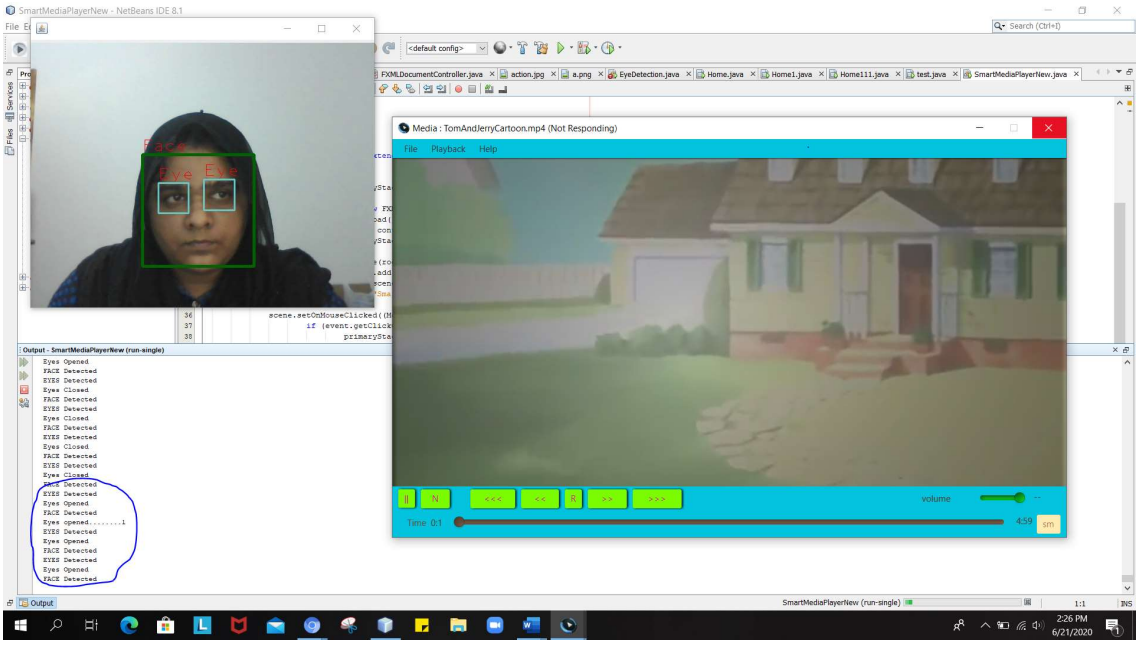

Figure 4.6-Screen shot of media player while detecting face and eyes

Following Figure 4.7 shows the results of media player while left eye closed and right eyes opened.

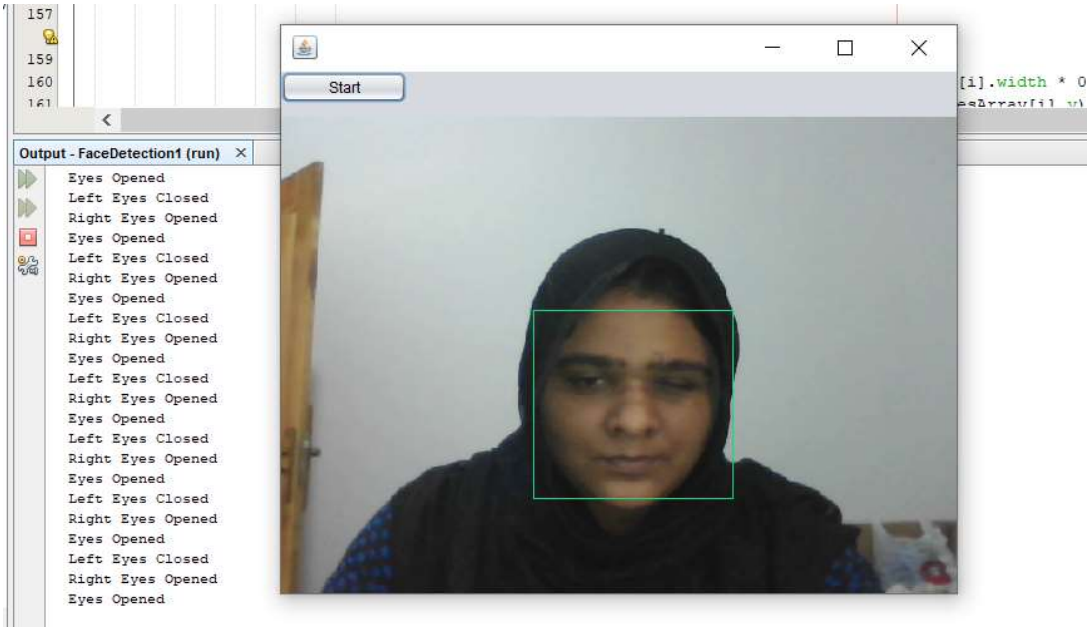

Figure 4.7-Screen shot of the application while left eye closed

Following Figure 4.8 shows the screen shots of application while closing both eyes

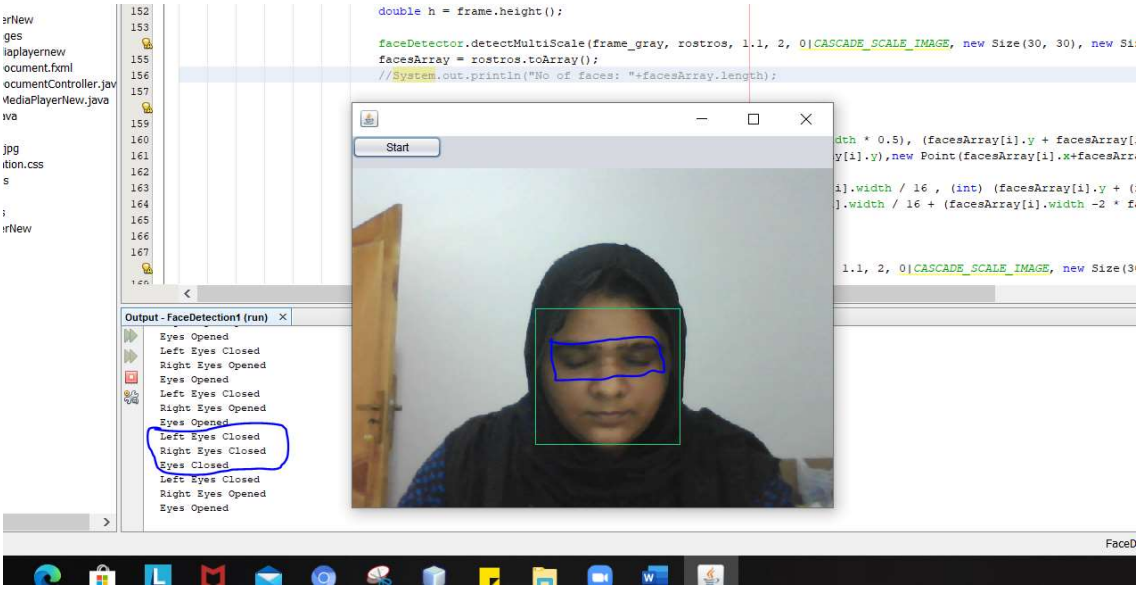

Figure 4.8-screen shots of media player while both eyes closed

Following Figure 4.9 shows the screen shots of media player while covering the half of the face.

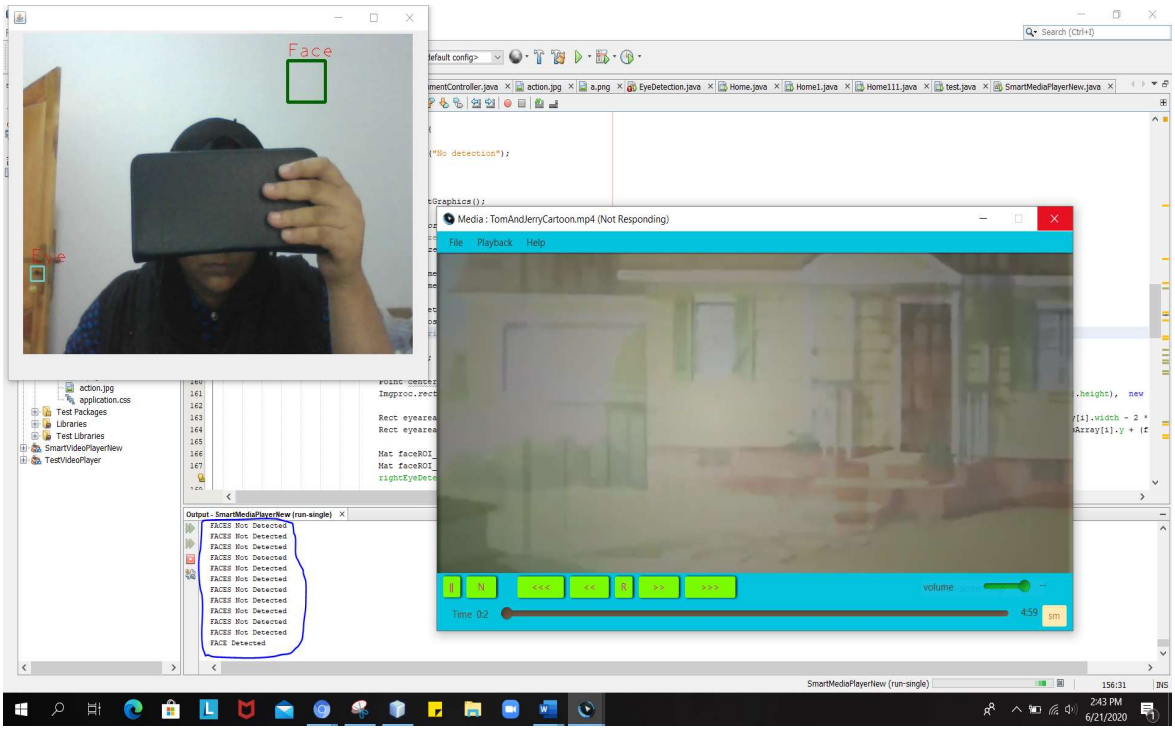

Figure 4.9-Screen shots of the media player while both eyes closed

Chapter 5

Conclusion

According to the literature surveys and analyzing the current media players, we have concluded that in our proposed solution we have introduced a media player application that simplify the image capturing and face detection from the captured image and checking whether the eyes are opened or closed. This process using image capturing and face detection, eye detection algorithms running as a thread in the backend. The player gets started to play automatically or keep playing as long as only while the eyes are open and it will go to pause while the eyes are closed or eyes not detected or human faces not detected. It will be a most beneficial product for the people whoever watching films while they are working and etc. As per the statistical analysis of the system, it can capture ~6 (6.059) frames per second and face detection rate is ~6 (5.965) faces per second. The percentage of open eyes detection is 81.82% with comparing the total human faces detected. But in our system, we are facing some environmental noise while detecting the human face and the similar objects like human face also considered as a human and it is affecting the performance of this system. So, we need a noise free background to get the efficient performance of the player. Even we make it user friendly some of the other functionalities like voice control, playing next, playing previous, fast forward, backward, slow operations are needs to be done manually. Thus, in future we have to find out solution for overcome these issues.

References

[1]. PiotarDalka, “Lip movement and gesture recognition for a multi model human computer interface”, IEEE, 2009

[2]. Marcelo Archajo Jos´ and Roseli de Deus Lopes, “Human computer interface controlled by lip”,IEEE Journal of Biomedical and Health Informatics, Vol.19, No.1,January 2015

[3]. MargritBetke, James Gips,” The Camera Mouse: Visual Tracking of Body Features to Provide Computer Access for People with Severe Disabilities”, IEEE Transactions on Neural Systems and Rehabilitation Engineering, Vol.10, No.1, March 2002.

[4]. Yo-Jen Tu1, Chung-Chieh Kao1, Huei-Yung Lin1 and Chin-Chen Chang, “Face and Gesture Based Human Computer Interaction”,International Journal of Signal Processing, Image Processing and Pattern Recognition Vol.8, No.9, pp.219-228,2015.

[5]. K.Meenakshi, Dr.A.Geetha and A.RajaPrabu, “Mouth Gestures as Human-Computer Interface”, International Journal for Science and Advance Research in Technology Vol.4. No.4, pp. 503-509, April 2018.

[6]. Deepika Nadar, Suma Acharya, Sarvesh Parab, Akhil karkera, “Look based Media Player”, International Journal for Research in Applied Science and Engineering Technology (IJRASET), Volume 7 Issue IV, Apr 2019

[7]. Disha H. Nagpure, Shubhangi T. Patil, Snehal P. Bujadkar, “Eye Motion Detection Using Single Webcam for Person with Disabilities”, International Journal of Engineering Research and Technology (IJERT), Vol. 3 Issue 3, March – 2014

[8]. Siddharth Swarup Rautaray, Anupam Agrawal, “A Vision based Hand Gesture Interface for Controlling VLC Media Player”, International Journal of Computer Applications (0975 – 8887), Volume 10– No.7, November 2010

[9]. Mukesh Vishwakarma, Akshay Navratne, Sneha Ghorpade, Saket Thombre, Trupti Kumbhare, “Media Player with Face Detection and Hand Gesture”, International Research Journal of Engineering and Technology (IRJET), Volume: 04 Issue: 03 | Mar -2017

[10]. Paul Viola and Michael J. Jones, “Robust Real-Time Face Detection”, Published in 2004 International Journal of Computer Vision 57(2), 137154, and 2004.

[11]. Mahdi Abbasi, Mohamed R.Khosravi, ”A Robust and Accurate Particle Filter-Based Pupil Detection Method for Big Datasets of Eye Video”, Published in December 2019

[12]. J.JMagee, M.R.Scott, B.Waber,M.Betke, “Eyekeys: a real-time vision interface based on gaze detection from a low-grade video camera,” IEEE Workshop on Real-Time Vision for Human- Computer Interaction (RTV4HCI), 2004

[13]. Dr. Aman Bhardwaj,” Model-Based Eye Detection and Animation”, International Journal of All Research Education and Scientific Methods (IJARESM), ISSN: 2455-6211, Volume 3, Issue 6, June- 2015

[14]. ZAFER SAVAS,” REAL-TIME DETECTION AND TRACKING OF HUMAN EYES IN VIDEO SEQUENCES”, August 2005

[15]. Xuebai Zhang, Shyan-Ming Yuan, Ming-Dao Chen, Xiaolong Liu," A Complete System for Analysis of Video Lecture Based on Eye Tracking", IEEE, Volume: 06, ISSN: 2169-3536, 16 August 2018

[16]. I.F.Ince,T.C.Yang, "A new low-cost eye tracking and blink detection approach: extracting eye features with blob extraction", 5th International Conference on Emerging Intelligent Computing Technology and Applications, 2009

[17] JavaCV.http://code.google.com/p/javacv/.

[18] OpenCV.http://opencv.willowgarage.com/wiki/.

[19] Paul Viola and Michael J. Jones, "Robust Real-Time Face Detection", Published in 2004 International Journal of Computer Vision 57(2), 137154, and 2004.

[20] S.V. Viraktamath, Mukund Katti, Aditya Khatawkar Pavan Kulkarni, "Face Detection and Tracking using Open CV", The SIJ Transactions on Computer Networks & Communication Engineering (CNCE), Vol. 1, No. 3, July-August 2013

[21] Dnyanada Jadhav1, Prof. L.M.R.J. Lobo2 ,"Hand Gesture Recognition System To Control Slide Show Navigation", International Journal of Application or Innovation in Engineering & Management (IJAIEM) Volume 3, Issue 1, January 2014

[22] N.Krishna Chaitanya, 2, R.Janardhan Rao "Controlling of windows media player using hand recognition system", The International Journal of Engineering and Science (IJES) Volume 3 Issue 12 Pages

[23] Ruslana Makovetsky and Gayane Petrosyan , "Fundamentals of Computer Vision Face Detection and Tracking with Web Camera".

[24] Accessible at https://blog.idrsolutions.com/2014/11/difference-java-javafx/ (March 20, 2016)

[25]https://becominghuman.ai/face-detection-using-opencv-with-haar-cascade-classifiers-941dbb25177

[26] https://sanyamgarg.blogspot.com/2016/03/a-blink-detection-technique-using.html

[27] https://www.pyimagesearch.com/2017/04/24/eye-blink-detection-opencv-python-dlib/

[28] https://www.quora.com/How-can-I-set-the-focus-of-a-webcam-or-any-other-camera-using-OpenCV

[29] https://photo.stackexchange.com/questions/40694/field-of-view-of-a-webcam

[30] Satya Mallick, https://www.learnopencv.com/approximate-focal-length-for-webcams-and-cell-phone-cameras/

[31] Gautami Shingan, Ajay Paratmandali, Yadnyawalkya, "Implementation of Media Player using Image processing and OpenCV JavaFx library", Published in August 2016, International Journal of Engineering Science and Computing, Volume 6 Issue No.8, 3 pages.

[32] Hossain M.E, Didar I, Imtiaz A, "Webcam-Based Accurate Eye-Central Localization", Published in 2013 Second International Conference on Robot, Vision and Signal Processing, INSPEC Accession Number: 14383177

[33] Furkan Ince I, Tae-Cheon Y, "A New Low-Cost Eye Tracking and Blink Detection Approach: Extracting Eye Features with Blob Extraction", Published in September 2009, Issn: 0302-9743

Appendix A

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package smartmediaplayernew;

import java.awt.FlowLayout;
import java.awt.Image;
import java.awt.image.BufferedImage;
import java.io.ByteArrayInputStream;
import java.io.File;
import java.io.InputStream;
import javax.imageio.ImageIO;
import javax.swing.ImageIcon;
import javax.swing.JFrame;
import javax.swing.JLabel;
import org.opencv.core.Core;
import org.opencv.core.Mat;
import org.opencv.core.MatOfByte;
import org.opencv.core.MatOfRect;
import org.opencv.core.Point;
import org.opencv.core.Rect;
import org.opencv.core.Scalar;
import org.opencv.core.Size;
import org.opencv.imgcodecs.Imgcodecs;
import org.opencv.imgproc.Imgproc;
import org.opencv.objdetect.CascadeClassifier;
import static org.opencv.objdetect.Objdetect.CASCADE_SCALE_IMAGE;
import org.opencv.videoio.VideoCapture;

/**
 *
 * @author LENOVO
 */
public class Web extends Thread {

    static JFrame frame;
    static JLabel lbl;
    static ImageIcon icon;
```

```
Rect[] facesArray;
Rect[] eyesArray;

Rect[] left_eyesArray;
Rect[] right_eyesArray;

MatOfRect left_eyes = new MatOfRect();
MatOfRect right_eyes = new MatOfRect();

public void run()
    {
       try{

           CascadeClassifier cascadeFaceClassifier = new CascadeClassifier("C:/Program Files (x86)/opencv3.1.0/build/etc/haarcascades/haarcascade_frontalface_alt2.xml");
           CascadeClassifier cascadeEyeClassifier = new CascadeClassifier("C:/Program Files (x86)/opencv3.1.0/build/etc/haarcascades/haarcascade_eye.xml");
           CascadeClassifier openEyeClassifier = new CascadeClassifier("C:/Program Files (x86)/opencv3.1.0/build/etc/haarcascades/haarcascade_eye_tree_eyeglasses.xml");
           CascadeClassifier rightEyeDetector = new CascadeClassifier("C:/Program Files (x86)/opencv3.1.0/build/etc/haarcascades/haarcascade_righteye_2splits.xml");
           CascadeClassifier leftEyeDetector = new CascadeClassifier("C:/Program Files (x86)/opencv3.1.0/build/etc/haarcascades/haarcascade_lefteye_2splits.xml");


           VideoCapture videoDevice = new VideoCapture(0);
           Imgcodecs imageCod = new Imgcodecs();
```

```
  if(!videoDevice.isOpened())
  {
      System.out.println("Error");
  }
  else
  {
      Mat frameCapture = new Mat();
      Mat frame_gray = new Mat();
      while(true)
      {
          videoDevice.read(frameCapture);

          Imgproc.cvtColor(frameCapture, frame_gray, Imgproc.COLOR_BGR2GRAY);
          Imgproc.resize(frame_gray, frame_gray, new Size(), 0.5, 0.5, Imgproc.INTER_LINEAR);
          Imgproc.equalizeHist(frame_gray, frame_gray);

          double w = frameCapture.width();
          double h = frameCapture.height();

          MatOfRect faces = new MatOfRect();
          cascadeFaceClassifier.detectMultiScale(frame_gray, faces, 1.1, 2, 0|CASCADE_SCALE_IMAGE, new Size(30, 30), new Size(w, h) );
          facesArray = faces.toArray();

          imageCod.imwrite("action.jpg", frame_gray); //to store the frame capture

          for (Rect rect : faces.toArray()) {
              Imgproc.putText(frameCapture, "Face", new Point(rect.x,rect.y-5), 1, 2, new Scalar(0,0,255));
              Imgproc.rectangle(frameCapture, new Point(rect.x, rect.y), new Point(rect.x + rect.width, rect.y + rect.height),new Scalar(0, 100, 0),3);
          }
```

```
if (faces.toArray().length > 0){
    System.out.println("FACE Detected");
    for (int i = 0; i < facesArray.length; i++) {
       MatOfRect eyes = new MatOfRect();
        MatOfRect oEyes = new MatOfRect();
        cascadeEyeClassifier.detectMultiScale(frameCapture, eyes);
        for (Rect rect : eyes.toArray()) {
            Imgproc.putText(frameCapture, "Eye", new Point(rect.x,rect.y-5), 1, 2, new Scalar(0,0,255));
            Imgproc.rectangle(frameCapture, new Point(rect.x, rect.y), new Point(rect.x + rect.width, rect.y + rect.height),new Scalar(200, 200, 100),2);
        }
        if (eyes.toArray().length > 0){
            PushImage(ConvertMat2Image(frameCapture));
            System.out.println("EYES Detected");
        }else{
            System.out.println("EYES Not Detected");
        }

        Rect eyearea_right = new Rect( facesArray[i].x , facesArray[i].y ,facesArray[i].width /2, facesArray[i].height * 2/3  );
        Rect eyearea_left = new Rect(facesArray[i].x , facesArray[i].y ,facesArray[i].width /2, facesArray[i].height * 2/3);

        Rect eyearea_right = new Rect( facesArray[i].x + facesArray[i].width / 16 , (int) (facesArray[i].y +
                                       (facesArray[i].height / 4.5)) ,
                                       (facesArray[i].width - 2 * facesArray[i].width / 16) / 2, (int) (facesArray[i].height / 3.0));
        Rect eyearea_left = new Rect( facesArray[i].x + facesArray[i].width / 16 + (facesArray[i].width -2 * facesArray[i].width / 16)/2 ,
                                      (int) (facesArray[i].y + (facesArray[i].height / 4.5)), (facesArray[i].width - 2 * facesArray[i].width / 16) / 2,
                                      (int) (facesArray[i].height / 3.0));

        Mat faceROI_right = frame_gray.submat(eyearea_right);
        Mat faceROI_left = frame_gray.submat(eyearea_left);

        rightEyeDetector.detectMultiScale(faceROI_right, right_eyes, 1.1, 2, 0|CASCADE_SCALE_IMAGE, new Size(30, 30), new Size(w, h));
        leftEyeDetector.detectMultiScale(faceROI_left, left_eyes, 1.1, 2, 0|CASCADE_SCALE_IMAGE, new Size(30, 30), new Size(w, h));

        left_eyesArray = left_eyes.toArray();
        right_eyesArray = right_eyes.toArray();

        if ((left_eyesArray.length > 0) || (right_eyesArray.length > 0)){
            System.out.println("Eyes Opened");
        }   else{
            System.out.println("Eyes Closed");
        }
    }
}else{
    System.out.println("FACES Not Detected");
}
```

```
                }
            }
            Thread.sleep(1000);
            File file = new File("action.jpg");
            file.delete();
            videoDevice.release();
        }catch(Exception e){}
    }

    private static BufferedImage ConvertMat2Image(Mat image) {
        MatOfByte byteMatImage = new MatOfByte();
        Imgcodecs.imencode(".jpg", image, byteMatImage);
        byte[] byteArray = byteMatImage.toArray();
        BufferedImage goruntu = null;
        try {
            InputStream in = new ByteArrayInputStream(byteArray);
            goruntu = ImageIO.read(in);
        } catch (Exception e) {
            e.printStackTrace();
            return null;
        }
        return goruntu;
    }

    public static void SetFrame() {
        frame = new JFrame();
        frame.setLayout(new FlowLayout());
        frame.setSize(600, 500);
        frame.setVisible(true);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }

    public static void PushImage(Image img2) {
        if (frame == null)
            SetFrame();
        if (lbl != null)
            frame.remove(lbl);
        icon = new ImageIcon(img2);
        lbl = new JLabel();
        lbl.setIcon(icon);
        frame.add(lbl);
        frame.revalidate();
    }
}
```

Appendix B

To initialize the media player

```
me = new Media(VUrl);
mp = new MediaPlayer(me);
mv.setMediaPlayer(mp);
mp.setAutoPlay(false);
```

To choose media file from path

```
public void handle(ActionEvent e)
{
try{
        mp.pause();
        FileChooser fileChooser = new FileChooser();
        File file = fileChooser.showOpenDialog(stage);
        String path =  file.getAbsolutePath();
        stage.setTitle("Media : " + file.getName() );
        path = path.replace("\\", "/");
        if(file!=null)
        {
                try
                {
                        me = new Media(new File(path).toURI().toString());
                        mp.stop();
                        mp = new MediaPlayer(me);
                        mp.setAutoPlay(true);
                        mv.setMediaPlayer(mp);
                }
                catch(Exception e1)
                {
                        e1.printStackTrace();
                }
        }

        mp.currentTimeProperty().addListener(new InvalidationListener()
        {
                public void invalidated(Observable ov)
                {
                        updateValues();
                }
        });
        }catch(Exception ex)
{}
}
```

To set volume slider

```
//------------volumeslider-----------------------

vol.valueProperty().addListener(new InvalidationListener() {
            public void invalidated(Observable ov) {
                if (vol.isValueChanging()) {
                    mp.setVolume(vol.getValue() / 100.0);
                }
                vl.setText("%" + vol.getValue());
            }
        });
```

To Set time slider

```
//-----------timeslider----------------------

time.valueProperty().addListener(new InvalidationListener()
        {
            public void invalidated(Observable ov)
            {
                if(time.isPressed())
                {
                    mp.seek(mp.getMedia().getDuration().multiply(time.getValue()/100));
                }
            }
        });
```

For Play video

```
public void play(ActionEvent event){

        Status status = mp.getStatus();

        if(status==Status.PLAYING)
        {
            mp.pause();
        }

        if(status==Status.PAUSED)
        {
            mp.play();
        }

}
```

For basic player controls

```
public void fast(ActionEvent event){
        r = r + 1.5;
        mp.setRate(r);
}

public void slow(ActionEvent event){
        s=s*0.8;
        mp.setRate(s);
}

public void reload(ActionEvent event){
        mp.seek(mp.getStartTime());
        mp.play();
}

public void normal(ActionEvent event){
        mp.setRate(1);
}

public void backward(ActionEvent event){
        mp.seek(mp.getCurrentTime().divide(2.5));
}

public void forward(ActionEvent event){
        mp.seek(mp.getCurrentTime().multiply(2.5));
}

public void help(ActionEvent event){

        Alert alert = new Alert(AlertType.INFORMATION);
        alert.setTitle("Help");
        alert.setHeaderText("Welcome to Smart Media Player Help");
        alert.setContentText("Documentation:\n||    : Play/Pause Button\nN

        alert.showAndWait();
}
```

For playlist

```
public void playlist(ActionEvent event){
        mp.pause();
        DirectoryChooser directoryChooser = new DirectoryChooser();
        File file = directoryChooser.showDialog(null);
        if(file!=null){
                plist = file.getPath();
                plist = plist.replace("\\", "/");
                plist = plist + "/" ;
                listp = file.list();
        }
}
```

For face and eye detection from captured image

```
protected void cam(){
    try{
        Mat frameCapture = Imgcodecs.imread("action.jpg");
        cascadeFaceClassifier.detectMultiScale(frameCapture, faces);
        if (faces.toArray().length > 0){
            facesArray = faces.toArray();
            MatOfRect eyes = new MatOfRect();
            cascadeEyeClassifier.detectMultiScale(frameCapture, eyes);
            if (eyes.toArray().length > 0){
                for (int i = 0; i < facesArray.length; i++) {
                    Rect eyearea_right = new Rect( facesArray[i].x , facesArray[i].y ,facesArray[i].width /2, facesArray[i].height * 2/3  );
                    Rect eyearea_left = new Rect(facesArray[i].x , facesArray[i].y ,facesArray[i].width /2, facesArray[i].height * 2/3);
                    Mat faceROI_right = frameCapture.submat(eyearea_right);
                    Mat faceROI_left = frameCapture.submat(eyearea_left);
                    double w = frameCapture.width();
                    double h = frameCapture.height();
                    rightEyeDetector.detectMultiScale(faceROI_right, right_eyes, 1.1, 2, 0|CASCADE_SCALE_IMAGE, new Size(30, 30), new Size(w, h));
                    leftEyeDetector.detectMultiScale(faceROI_left, left_eyes, 1.1, 2, 0|CASCADE_SCALE_IMAGE, new Size(30, 30), new Size(w, h));
                    left_eyesArray = left_eyes.toArray();
                    right_eyesArray = right_eyes.toArray();
                    if ((left_eyesArray.length > 0) || (right_eyesArray.length > 0)){
                    } else{
                        Status status = mp.getStatus();
                        if(status==Status.PLAYING)
                        {
                            mp.pause();
                            cam2();
                        }
                    }
                }
            }else{
                Status status = mp.getStatus();
                if(status==Status.PLAYING)
                {
                    mp.pause();
                    cam2();
                }
            }
        }else{
            Status status = mp.getStatus();
            if(status==Status.PLAYING)
            {
                mp.pause();
                cam2();
```

For next song

```
protected void nextsong()
{
        numlist ++;
        if(numlist == listp.length)
        {
            numlist = 0 ;
        }
        stage.setTitle("Media : " + listp[numlist] );
        try
        {
            me = new Media(new File(plist + listp[numlist]).toURI().toString());
            mp.stop();
            mp = new MediaPlayer(me);
            mp.setAutoPlay(true);
            mv.setMediaPlayer(mp);
        }
        catch(Exception e1)
        {
            e1.printStackTrace();
        }

        mp.currentTimeProperty().addListener(new InvalidationListener()
        {
            public void invalidated(Observable ov)
            {
                updateValues();
            }
        });
}
```

For previous song

```
protected void previoussong()
{
        numlist --;
        if(numlist == -1)
        {
            numlist = 0 ;
        }
        stage.setTitle("Media : " + listp[numlist] );
        try
        {
            me = new Media(new File(plist + listp[numlist]).toURI().toString());
            mp.stop();
            mp = new MediaPlayer(me);
            mp.setAutoPlay(true);
            mv.setMediaPlayer(mp);
        }
        catch(Exception e1)
        {
            e1.printStackTrace();
        }

        mp.currentTimeProperty().addListener(new InvalidationListener()
        {
            public void invalidated(Observable ov)
            {
                updateValues();
            }
        });
}
```

Appendix C

Smart Media Player - User Survey

Smart media player can be able to play or pause the media stream while the human face is detecting on front of the screen and while the persons eyes are opening. It detects the Human face and Eyes and for the time it detecting a eyes video will be played else video will paused automatically. The system runs continuously while monitoring whether a human eye open or not and the player starts to running as soon as it is detecting the eyes again and which keeps running as long as it is detecting the eyes without the interruption. The player pauses as soon as the human eyes not completely seen. If there is no human face detected then the player will be stopped immediately.

1* Are you thinking that this kind of media player is essential ?

- [ ] Yes
- [ ] No

2* Moving head away from the vision of web front camera and observe whether it pauses or not?

- [ ] Yes, Paused
- [ ] Not Paused
- [ ] Other
- [ ] Other (Please Specify)

3* Entering again to the vision of the web front camera to check whether the video plays again.

- [ ] Yes, Playing
- [ ] Not playing
- [ ] Other
- [ ] Other (Please Specify)

4* Closing eyes in front of the vision and observe whether it is pauses or not.

- [ ] Yes, Paused
- [ ] Not Paused
- [ ] Other
- [ ] Other (Please Specify)

5* Opening eyes and observe whether it is playing the video again

- [ ] Yes, playing
- [ ] Not Playing
- [ ] Other
- [ ] Other (Please Specify)

6* Did you observe any changes in media player while moving back and forward?(Please specify the changes here)

7* Is that player playing while you shaking your head upward?

- [ ] Yes
- [ ] No

8* Is that player playing while you shaking your head below?

- [ ] Yes
- [ ] No

9* Is that player playing while you shaking your head to right sight?

- [ ] Yes
- [ ] No

10* Is that player playing while you shaking your left side ?

- [ ] Yes
- [ ] No

11* Is that video playing while you partially cover your face?

- [ ] Yes
- [ ] No
- [ ] Other

12* Is that video playing while you cover the eyes?

☐ Yes ☐ No ☐ Other

☐ Other (Please Specify)

13* How would you rate our system while comparing existing media players in usability? (Easy to use)

0 1 2 3 4 5 6 7 8 9 10

Extremely Poor Extremely Good

14 What are the changes your expecting in this media player? (If you have any suggestion please mention below)

15* How would you rate this system meet the functional goal of this smart media player with eye detection?

0 1 2 3 4 5 6 7 8 9 10

Extremely Poor Extremely Good