ABSTRACT

AirMouse, a Hand Gesture Recognizing System which detects various positions of hand within the window. On recognizing a gesture similar to one that is saved in it's database it handles the function of Music Player accordingly, some of those functions are Play Music, Pause, Play Next, Volume Up, Volume Down, Display Playlist, Display Lyrics and Text on the screen.

AirMouse enables the user to overcome the physical barriers and reduces the Physical Communication between the user and the system, thus also reduces the wear and tear of the device.

It requires a basic Camera located in Front of the system, windows (with the most recent release) as Operating System and a Powerful Processor for making recognition work and Transform. AirMouse is coded in Python with Libraries OpenCV, NumPy, Math and Pyglet. OpenCV is a free Library of programming features mainly aimed at real-time computer vision developed by Intel (in 1995).

In Future, it can be used to control other utilities and applications and even to control Peripheral devices.

Table of Contents

Bonafide Certificate **i**

Acknowledgement **ii**

Abstract **iii**

Table of Content **iv**

List of Tables **vii**

List of Figures **viii**

List of Symbols, Abbreviations and Nomenclature **ix**

Chapter 1: Introduction **1**

1.1: Problem Statement 1

1.2: Purpose 1

1.3: Scope 2

1.4: Overview 2

Chapter 2: System Overview **3**

2.1: Goals and Objective 3

Chapter 3: Design Considerations **5**

3.1: Designs Assumptions and Dependencies 5

3.2: Design Constraints 5

3.2.1: Hardware Constraints 5

3.2.2: Software Constraints 6

3.2.3: Performance Constraints 6

3.3: Design Goals and Guidelines 6

Chapter 4: Data Design **7**

4.1: Data Description 7

Chapter 5: System Architecture **9**

5.1: Architectural Design 9

5.2: Description of Components 10

5.2.1: Hand Gesture Recognition 10

5.2.1.1: Processing Narrative 10

5.2.1.2: Gesture Recognition Interface Description 10
5.2.1.3: Processing Detail 10
5.2.2: Real Time Initiator 11
5.2.2.1: Detailed Design 12
5.2.3: Music Player 12
Chapter 6: Real – Time Initiator (Module I) **13**
6.1: Capture frames and convert to grayscale 13
6.2: Blur Image 13
6.3: Thresholding 14
6.4: Draw Contours 15
6.5: Find Convex Hull and Convexity Defects 15
6.6: Plotting Centroid 16
6.7: Finding Hand Positions 16
Chapter 7: **Music Player (Module II)** **17**
7.1: Load, Append and Read into Queue 17
7.2: Position Recognition 17
Chapter 8: **Libraries and Tools** **22**
8.1: Python 22
8.1.1: History of Python 22
8.1.2: Python Features 23
8.2: OpenCV 24
8.2.1: History of OpenCV 24
8.2.2: Methods Used 25
8.3: NumPy 28
8.3.1: Limitations 28
8.3.2: History of NumPy 28
8.3.3: Methods Used 29
8.4: Pyglet 30
8.4.1: Features of Pyglet 30
8.4.2: AVbin 30
8.4.3: Methods Used 31

8.5: Math 32
8.5.1: Methods Used 32
8.6: PyCharm 33
8.6.1: Why use PyCharm? 33
8.6.2: Why might I not want to use PyCharm? 33
8.6.3: What are some unfair criticisms of PyCharm? 34
Conclusions and Future Scope **36**
References **37**

List of Table

Table Number	Table Name	Page Number
Table 8.1	OpenCV Methods	25
Table 8.2	NumPy Methods	29
Table 8.3	Pyglet Methods	31
Table 8.4	Math Methods	32

List of Figures

Figure Number	Figure Name	Page Number
Figure 4.1	Entity Relationship Diagram of System Entities	7
Figure 4.2	Use case Diagram for RTI	8
Figure 5.1	Use case Diagram of the system	9
Figure 5.2	Data Flow diagram for Gesture Recognition (Level 0)	11
Figure 5.3	Detailed Data Flow Diagram for RTI (Level 1	12
Figure 6.1	Grayscale of Captured Frame	13
Figure 6.2	Blurred Image of Grayscaled image	14
Figure 6.3	Threshold Image of Captured Image	14
Figure 6.4	Contours on Captured Image	15
Figure 6.5	Convexity Defects and Convex Hulls on Captured Image	15
Figure 6.6	Centroid of the Frame	16
Figure 7.1	Use Case Diagram for Music Player	17
Figure 7.2	Started with Hand Gesture	18
Figure 7.3	Play Music Gesture	18
Figure 7.4	Volume Up Gesture	18
Figure 7.5	Next Track Gesture	19
Figure 7.6	Volume Down Gesture	19
Figure 7.7	Pause Music Gesture	19
Figure 7.8	Playlist Gesture	20
Figure 7.9	Playlist of Tracks	20
Figure 7.10	Lyrics Gesture	21
Figure 7.11	Lyrics of the playing Music	21

List of Symbols, Abbreviations and Nomenclature

Symbol, Abbreviations and Nomenclature	Descriptions
HCI	Human Computer Interaction
RTI	Real Time Initiators
OpenCV	Open Source Computer Vision
IDE	Integrated Development Environment
PC	Personal Computer
GB	Giga Byte
HD	Hard Disk
ROI	The Portion Of The Image To Be Used

Chapter 1

INTRODUCTION

AirMouse is a Hand Gesture Recognizing Application written in Python with OpenCV for Windows Operating System. It recognizes hand gestures and perform functions on a Music Player Virtually Created in Python.

1.1 Problem Statement

Human Computer Interaction (HCI) is getting more and more important with the improvement of the technology. In the history, different perspectives inspired people to develop some innovative systems. Static keys and buttons, track path devices, touch and multi-touch screens have developed respectively. The next innovation should let people use computers, game consoles or mobile devices without touching in order to control those devices easily and effortlessly. Thus Hand Gesture Recognition are the next improvement of interaction with computers and mobile devices. Controlling Computer and its applications is not an easy task to operate when the user is a little far away from the system. Listening Music along with working on the system or doing something else, This Recognition System will help to operate the Application on an Operating System.

- To design a software/interface/utility which detects hand gestures patterns instead of physical touch.
- The camera is positioned such that it recognizes the movement of hand and performs some operations.

1.2 Purpose

This Project enables a user to overcome the physical barriers between the user and the System and thus reduces the mechanical wear and tear of the system. Also it increases durability of that system, making process of maintenance easier for a user.

1.3 Scope

- The goal is to manage computers and control its basic functions with gestures rather than pointing and clicking a mouse or touching a display directly.
- User friendly application which minimizes the requirement of physical connection.
- Powerful Media application that minimizes the timing consumption.
- This System is a Real Time System, which works on the real-time data and processed simultaneously.

1.4 Overview

"**AirMouse**" is a Hand Gesture Recognizer which detects position of a hand when placed near the active webcam or a simple camera.

It can then perform various functions accordingly, after relating that gesture to its database. Some functions that are performed on a Music Player in our project are Play, Pause, Play Next, Volume Up, Volume Down, Showing Lyrics of Playing Songs, and Playlist etc.

This System is written in Python with OpenCV Library, using Pycharm as an IDE.

Chapter 2
SYSTEM OVERVIEW

The main goal of the project is to track and recognize basic hand gestures and process them to control a Music Player in Windows Operating System. The functionality of AirMouse is to provide an interaction between the Music Player and its user without need to touch the device. Thus Laptop/PC users can control the devices when they are doing their daily activities. There are some activities that one has to use his hands while doing it, like cooking meal. The AirMouse offers a solution to people to control their Laptop/PC even when they are doing that kind of activities. The system will also work properly in some specific conditions that make people use their Laptop/PC difficult such as using the device with wet hands or with gloves.

2.1 Goals and Objectives

- **Reliability**

  Our system is highly reliable as its requirements are simple that is webcam which is present in every laptop and computer. It will contain a dataset of various hand features with different types and colors in order to decrease the error tolerance of the system.

- **Low-Cost**

  Implementation of the stereovision technology by using low cost cameras provides an inexpensive solution with respect to the current technologies which are already available for all version of devices for free.

- **Maintainability**

  Any error occurred while the system is in use will be tolerated and system recovers itself and continues properly.

- **Portability**

  This software is portable as it is just to be put on the devices which have python and OpenCV installed on them which are free and platform independent which makes them portable.

- **Performance**

  Since the system will work on real time, performance is one of the most important topics of the system. Performance of the system will be high enough that user can use the system without noticeable delays or performance problems.

- **Usability**

  Usability for the system is quite easy as the person just needs to perform various hand gestures and move hand in various directions which makes the particular software very user-friendly.

- **Time**

  Software will work on real time. Response of the system after the user makes an action will be fast enough that does not cause a problem. Although delay is about 0.4 seconds to remove any confusion.

- **Security**

  As this software is based on python thus the internal security manager provides it a level of security making it highly efficient also when it will work on UNIX it will become more secure from external attacks.

- **User – Friendly**

  This system is user friendly, thus easy to learn and easy to operate.

- **Availability**

  This project is available to all as this project is based on OpenCV and python which are open-source that is anybody can use and operate it easily without facing any issues.

Chapter 3

Design Considerations

AirMouse comes with many assumptions, dependencies and especially constraints.

3.1 Designs Assumptions and Dependencies

This Project comes with many assumptions and dependencies,

- The system requires a camera or a webcam located in front, that might not be present in few systems.
- This Python based System is limited to Windows operating System.
- It is needed that the performance and time limits of the software will be applied considering that the software would be working on a laptop/PC.
- It is assumed that this project will operate well on new generation systems without any problems if the systems have powerful processors and a camera located in front.

3.2 Design Constraints

AirMouse has many hardware and software constraints.

3.2.1 Hardware Constraints

The system needs a powerful processor in order to make a recognition work and transform. A powerful processor like Intel Atom will be sufficient.

The hardware System should have atleast 2 GB RAM and minimum of 5 GB HD space. System memory consumptions never exceed 20% of the total memory, approximately 200 MB.

A Webcam or a camera with atleast 30 frames per second and minimum 640x480 pixels located in front of laptop/PC is the most essential part of this System.

3.2.2 Software Constraints

When the hardware part of the system is completed, software should work on Windows x32 or higher (with most recent release).

Webcam Drivers which are device specific should be installed in the system.

This System is coded in Python along with libraries OpenCV, Math, Numpy and Pyglet. Therefore, all libraries should be imported and installed in the system.

3.2.3 Performance Constraints

When the system is activated, system will be in a state that interaction between end-users will be the most frequent event; thus, the system should process 6-7 frames/sec for a proper and error-free interaction.

3.3 Design Goals and Guidelines

The highest priority of the software is to design the Gesture Recognition module. After the module is designed, Hand Recognition module will be inherited and Music Player module will be manipulated. Thus, design of Gesture Recognition module is the first main step of the software. Module descriptions are explained in detailed at the fifth chapter. One of the most important goals of the software is to keep the memory use in reasonable degree.

Since the aim of the software is the software can be implemented into Laptops and PCs, memory use is one of the most critical points that the software will faced. The upper limit of the memory use will be 20% of the memory.

The other important goal is the speed of the software. The process of the software, which is recognition of the gesture, processing it and providing the action, will be done in at most 0.2 second in order that the software will work in reasonable time.

Chapter 4
Data Design

4.1 Data Description

AirMouse system consists of two different components that are working together. Real-Time initiator and Music Player Interface are the mentioned components.

At first the Real-Time initiator part tracks a hand gesture and sends it to the interface. Interface translates the gesture to the appropriate command and if it is valid, the corresponding transition or the corresponding action will be done. The ER diagram of the perfect cooperation of these two parts, Real-Time initiator and Music Player Interface can be seen on Figure 1.

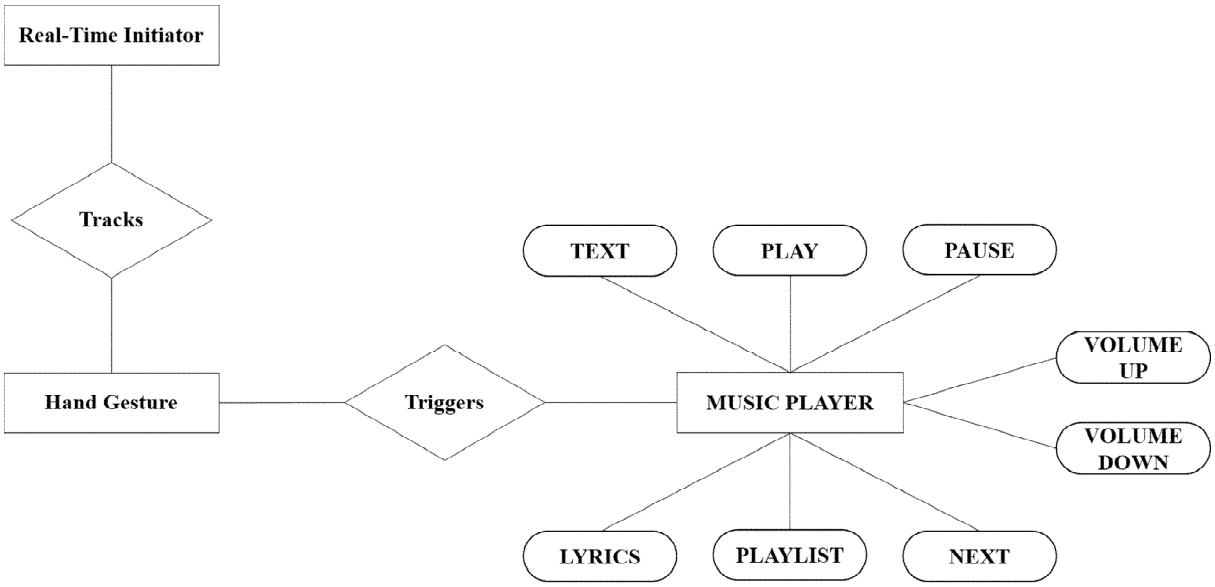

Figure 4.1 Entity Relationship Diagram of System Entities

AirMouse System consists of a main data structure named Hand Gesture, this data structure is created, modified or used by the two modules of the system, Real-Time Initiator and Music Player. Real-Time Initiator part of AirMouse System tracks a hand or gesture and creates a Gesture object belonging to it. RTI sends that gesture object to the Music Player and it is reflected on the screen by this module and will be changed according to the module.

Whenever the gesture is detected the position of the hand is calculated and according to which the matching function is called. For every function there is different position and gesture pattern is defined.

In the above ER Diagram (Figure 4.1), entities RTI and Music Player performs different roles in completion of the whole process. The output generated by RTI is the Hand Gesture created by the user and manipulated by the system which is then invokes functions of the Music Player. Functions such as Play Music, Pause, Next, Volume Up, Volume Down, Show Lyrics, Playlist and Text play as attributes of the Music Player (Entity).

For RTI, Video is captured and simultaneously processed into noise free Threshold image which is then used to detect Hand Gestures which then invokes Music Player functions. For clear and understandable video capturing the user should make proper hand movements so that they are easily detected. In order to be more understandable, Data Flow Diagram for this detection can be seen in Figure 4.2.

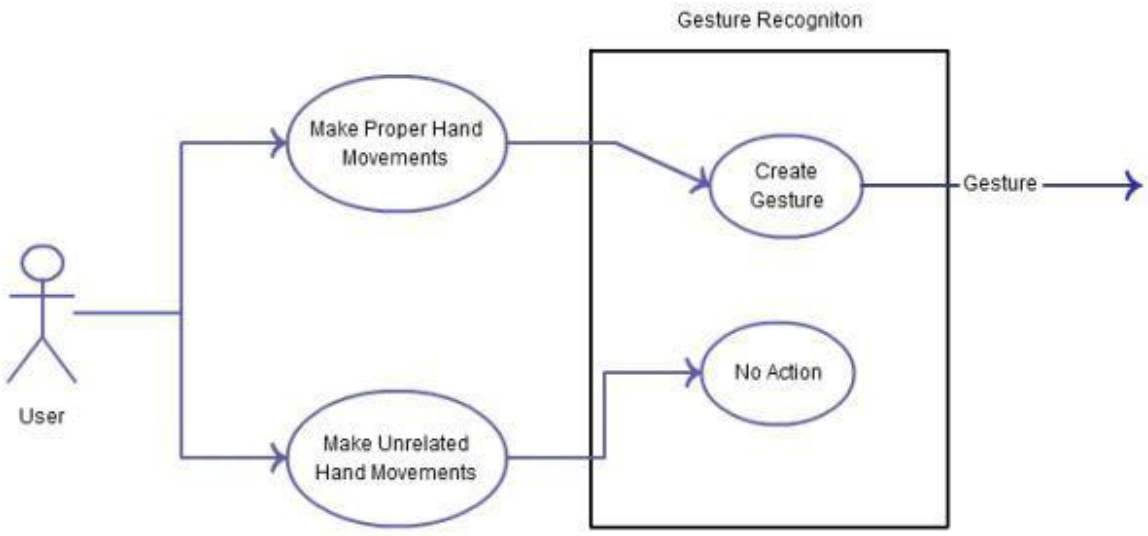

Figure 4.2 Use case Diagram for RTI

Chapter 5

System Architecture

Description of system architecture is given below.

5.1. Architectural Design

Our system is consists of two main modules, namely, RTI and Music Player. RTI is responsible for recognizing of hand movements and recognizing of position of the hand. Once, valid gesture comes, RTI interprets related action and triggers Music Player objects to do the action of the suggested gesture. In basic, Music System is the subsystem which enables the user of the Device to give the freedom of control without touching the device and RTI is the subsystem that is responsible for capturing, analyzing and detecting the desired action for the incoming hand movement and position.

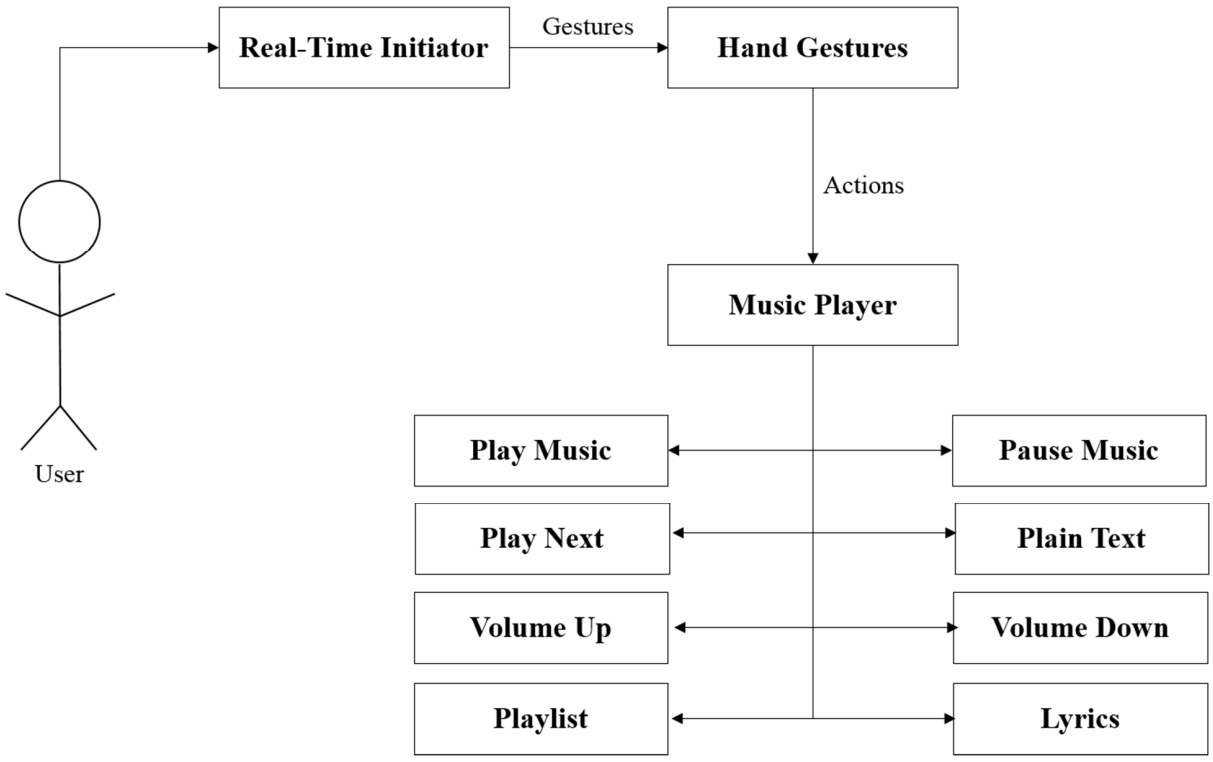

Figure 5.1 Use case Diagram of the system

5.2 Description of Components

In the upper section it's briefly said that AirMouse has two subsystems, namely Real-Time Initiator module and Music Player module. Real-Time Initiator is the combination of two components, Hand Recognition and Gesture Recognition. The responsibility of Hand Recognition is to capture consecutive frames and try to recognize the Hand of the user. Gesture Recognition gets the hand movements and position of hand from camera and tries to interpret the gesture. If a gesture is found in pre-defined gesture data set, Music Player is notified. Music Player looks for the gesture and takes in action the desired action.

5.2.1 Hand Gesture Recognition

Gesture Recognition is the component that basically recognizes incoming hand movement and position of hand from camera.

5.2.1.1. Processing Narrative

Two responsibilities of the Hand Gesture Recognition component is to recognize gestures and create gesture objects and send it to Music Player module for further processing with position of hand with respect to centroid of the frame.

5.2.1.2. Hand Gesture Recognition Interface Description

Frames that comes from the camera and trigger from hand recognition are the input to this component and output is the gesture object that is created after the recognition of the hand movement. Output of Hand Gesture Recognition component is an input to the Music Player Module.

5.2.1.3 Processing Details

The frames Recognized by the Hand Gesture component transform the input and creates noise free frames which then transferred to Music Player as Output and proper functions are invoked.

5.2.2 Real Time Initiator

RTI transforms the frame captured into threshold and calculates the position of the hand with respect to the centroid.

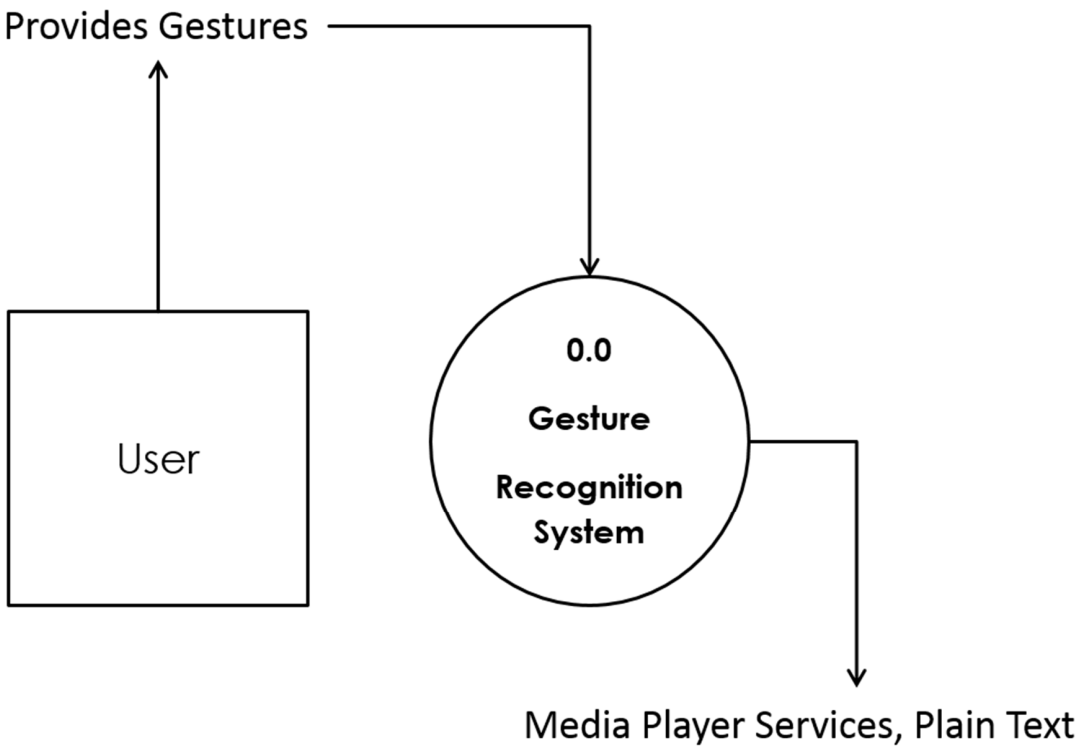

Figure 5.2 Data Flow diagram for Gesture Recognition (Level 0)

In the above diagram, when the user provides hand gestures it is sent to the System for Function calling. The gestures provided by the user are matched with the defined gestures in the system it then is sent to music player Module.

After the recognition of the frame it transform the image in several steps, as explained in detailed design.

5.2.2.1 Detailed Design

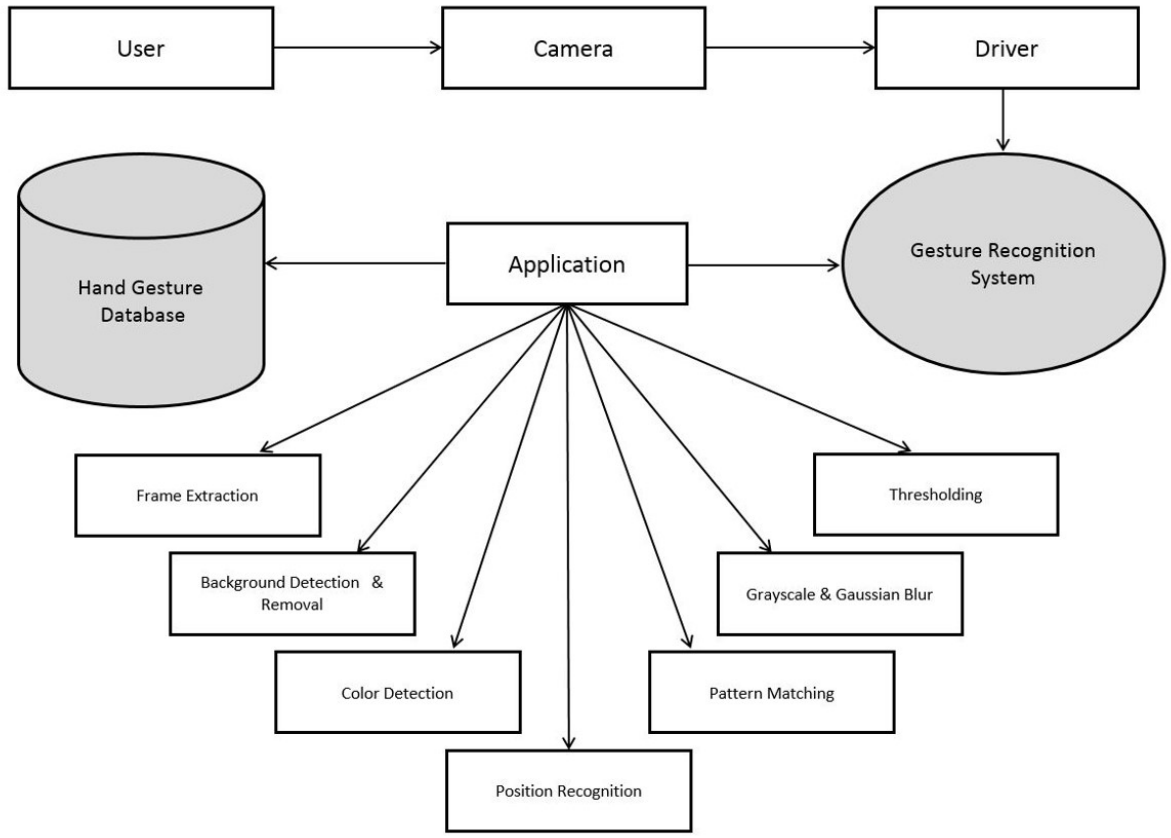

Figure 5.3 Detailed Data Flow Diagram for RTI (Level 1)

As in the above diagram, The frame captured are convert into grayscale then it is blurred using Gaussian Blur and finally converted into Threshold for complete removal of noise from the frame.

5.2.3 Music Player

Music Player takes Noise free hand gesture as input and then it calculates position of the hand and according to which it performs functions like play, pause and next etc.

Chapter 6
Real Time Initiator (Module 1)

This module Captures the image frame and transform into a Noise free product which is then taken as input to Music player for Gesture Recognition and evaluation of the functions of a Music Player.

6.1 Capture frames and convert to grayscale

Frames are captured using VideoCapture () function of OpenCV. The captured image is then converted from RGB to grayscale and then to binary in order to find the ROI i.e. the portion of the image which are further used for image processing. By doing this the task becomes binary to choose: "yes the pixel is of interest" or "no the pixel is not of interest".

Figure 6.1 Grayscale of Captured Frame

6.2 Blur image

The image is then blurred using Gaussian Blur for smoothing and to reduce noise and details from the image. There is no need for the details of the image but the shape of the object is required to track.

By blurring, the image is created smooth transition from one color to another and reduce the edge content. Thresholding is used for image segmentation and to create binary images from grayscale images.

Figure 6.2 Blurred Image of Grayscaled image

6.3 Thresholding

In very basic terms, thresholding is like a Low Pass Filter by allowing only particular color ranges to be highlighted as white while the other colors are suppressed by showing them as black.

Otsu's Binarization method is used in this system for Thresholding. In this method, OpenCV automatically calculates/approximates the threshold value of a bimodal image from its image histogram. But for optimal results, a clear background in front of the webcam is required which sometimes may not be possible.

Figure 6.3 Threshold Image of Captured Image

6.4 Draw contours

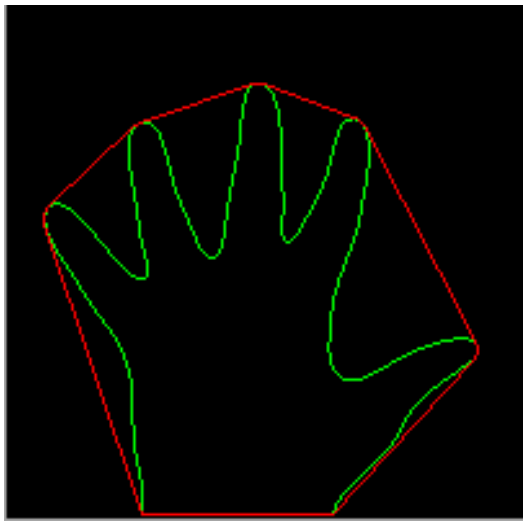

Figure 6.4 Contours on Captured Image

6.5 Find Convex Hull and Convexity Defects

The Convex Hull Points and the Convexity Defect Points are then searched.

The convex points are generally, the tip of the fingers. But there are other convex point too. So, Convexity Defects are also searched, which is the deepest point of deviation on the contour. By this the number of fingers extended can be found and then different functions according to the number of fingers extended can be performed.

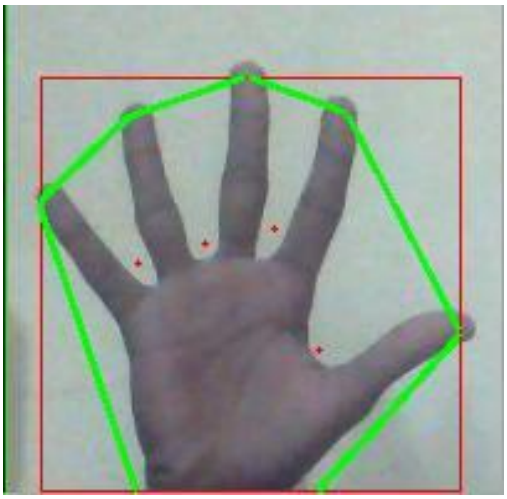

Figure 6.5 Convexity Defects and Convex Hulls on Captured Image

6.6 Plotting Centroid

Centroid of the frame is calculated, so that relative position of the hand is recognized and from that hand positions relative function is invoked.

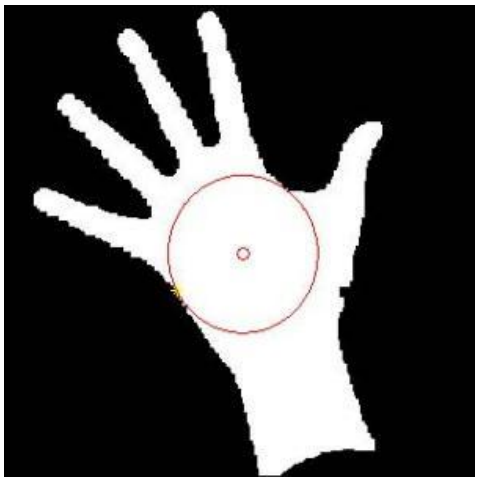

Figure 6.6 Centroid of the Frame

6.7 Finding Hand positions

When the hand is

- Far Left from the Centroid.
- Far Right from the Centroid.
- Far above the Centroid.
- Far below the Centroid.
- Near or on the Centroid.

Chapter 7

Music Player

This module perform functions on the basis of gestures performed by the user and the relative position of the hand from the centroid.

7.1 Load, Append and Read into Queue

After the frame is recognized by the RTI, the Music Tracks are loaded into Memory and Objects of Media Player are created using Media.Load (), Media.Player () functions of Pyglet Library.
All Objects of media player are stored into the queue and list of all the players is created using object.queue(song name) and players.append(object) of Pyglet.

7.2 Position Recognition

According to the position of the hand from the centroid following functions are invoked

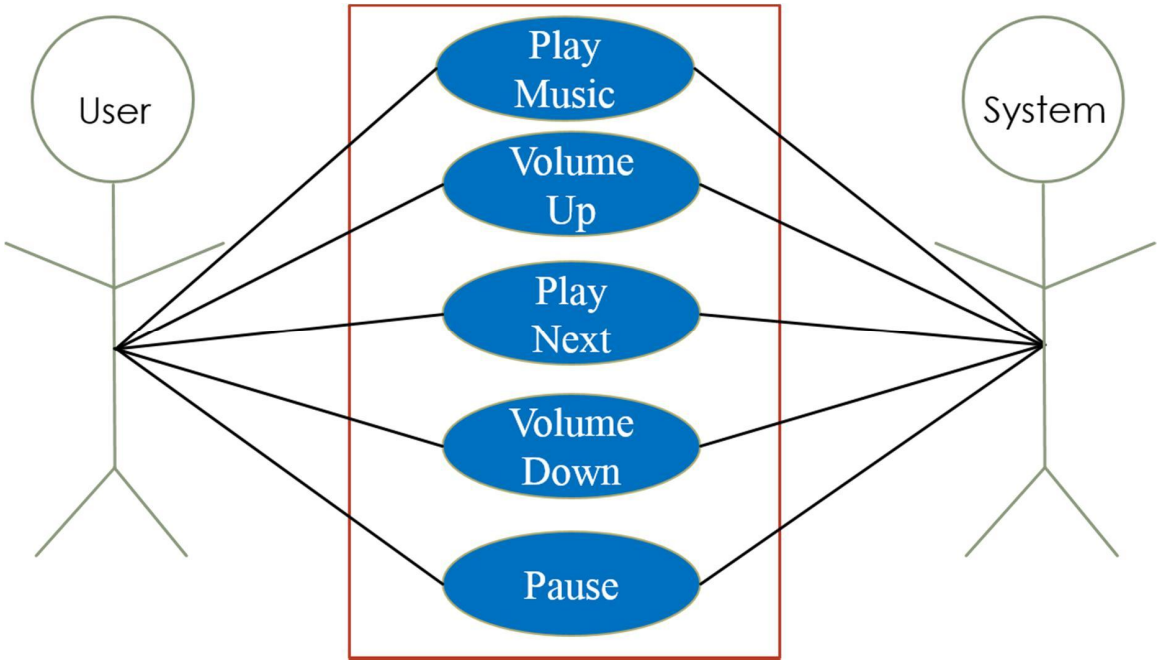

Figure 7.1 Use Case Diagram for Music Player

- Near or on the Centroid (with all fingers closed) – Started with AirMouse.

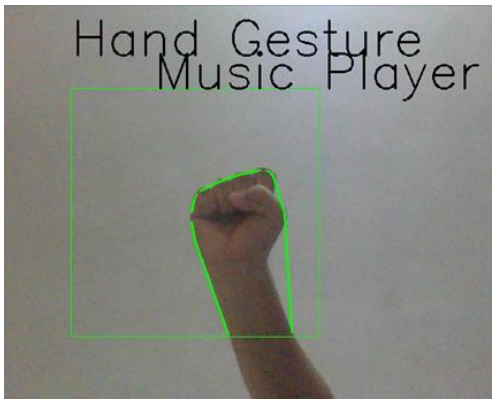

Figure 7.2 Started with Hand Gesture

- Near or on the Centroid (with all fingers Open) – Play Music.

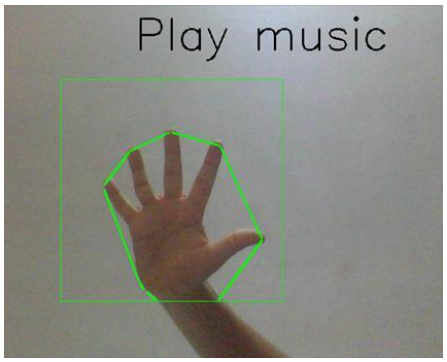

Figure 7.3 Play Music Gesture

- Far Above from the Centroid – Increase Volume.

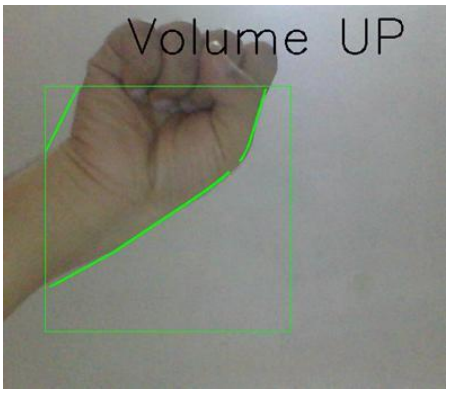

Figure 7.4 Volume Up Gesture

- Far Right from the Centroid – Play Next Track.

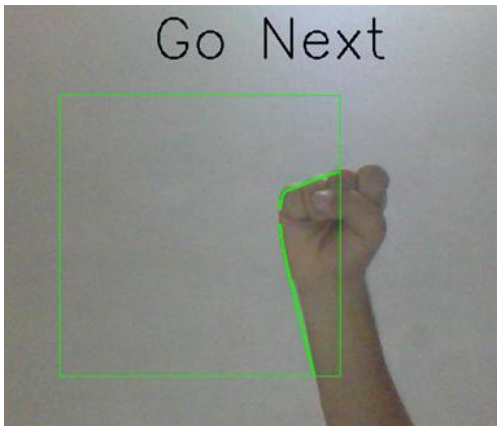

Figure 7.5 Next Track Gesture

- Far Below from the Centroid – Decrease Volume.

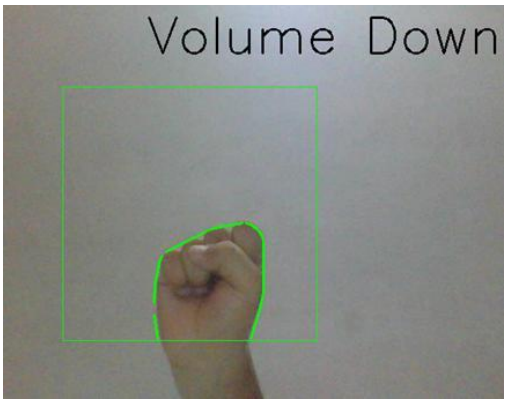

Figure 7.6 Volume Down Gesture

- Far Left from the Centroid – Pause Music.

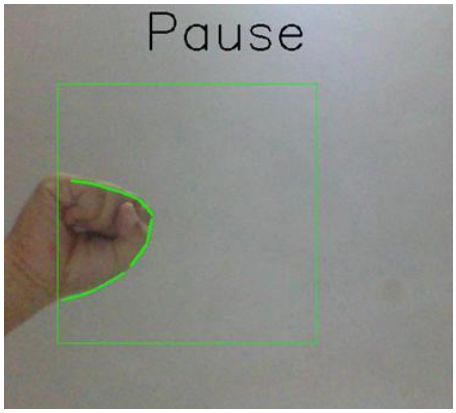

Figure 7.7 Pause Music Gesture

- Near or on the Centroid (with four fingers Open) – Opens Playlist.

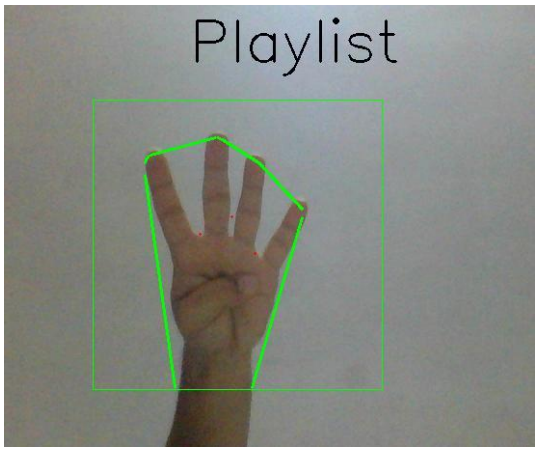

Figure 7.8 Playlist Gesture

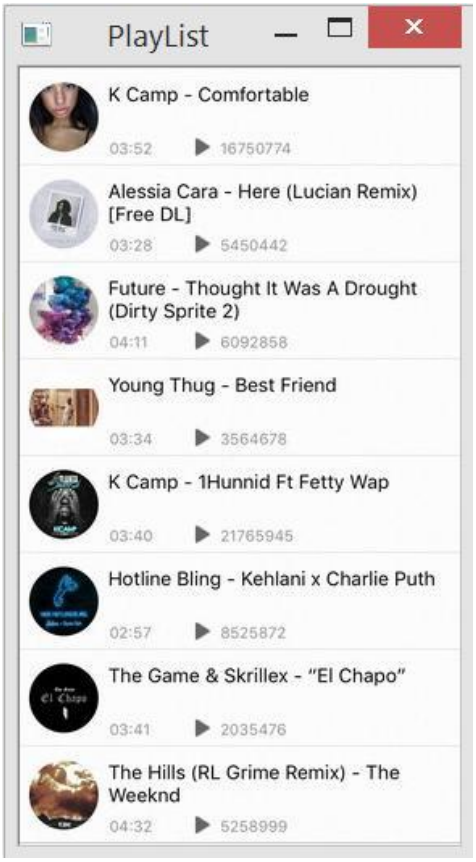

Figure 7.9 Playlist of Tracks

- Near or on the Centroid (with three fingers Open) – Opens Lyrics.

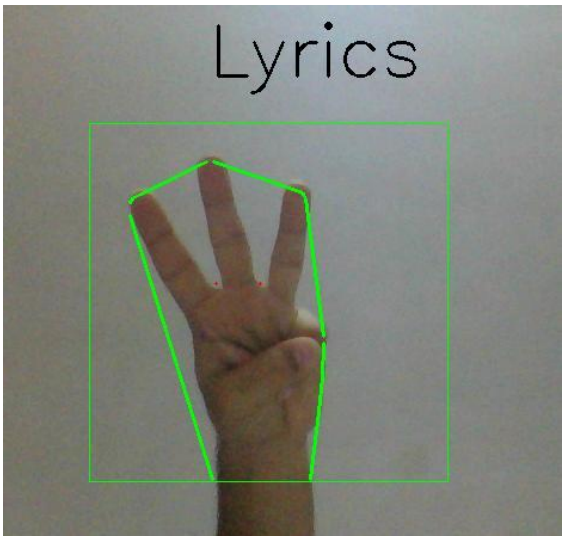

Figure 7.10 Lyrics Gesture

Figure 7.11 Lyrics of the playing Music

Chapter 8
Libraries and Tools

8.1 PYTHON

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

- **Python is interpreted:** Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- **Python is Interactive:** You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
- **Python is Object-Oriented:** Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
- **Python is a Beginner's Language:** Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

8.1.1 History of Python

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands. Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, Smalltalk, UNIX shell, and other scripting languages. Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL). Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

8.1.2 Python Features

Python's features include:

- **Easy-to-learn:** Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
- **Easy-to-read:** Python code is more clearly defined and visible to the eyes.
- **Easy-to-maintain:** Python's source code is fairly easy-to-maintain.
- **A broad standard library:** Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
- **Interactive Mode:** Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
- **Portable:** Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
- **Extendable:** You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
- **Databases:** Python provides interfaces to all major commercial databases.
- **GUI Programming:** Python supports GUI applications that can be created and ported to many system calls, libraries, and windows systems, such as Windows MFC, Macintosh, and the X Window system of UNIX.
- **Scalable:** Python provides a better structure and support for large programs than shell scripting.
- Apart from the above-mentioned features, Python has a big list of good features, few are listed below:
- IT supports functional and structured programming methods as well as OOP.
- It can be used as a scripting language or can be compiled to byte-code for building large applications.
- It provides very high-level dynamic data types and supports dynamic type checking.
- IT supports automatic garbage collection.
- It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

8.2 OPENCV

OpenCV (Open Source Computer Vision) is a library of programming functions mainly aimed at real-time computer vision. Originally developed by Intel's research center in Nizhny Novgorod (Russia), it was later supported by Willow Garage and is now maintained by Itseez. The library is cross-platform and free for use under the open-source BSD license.

8.2.1 History

Officially launched in 1999, the OpenCV project was initially an Intel Research initiative to advance CPU-intensive applications, part of a series of projects including real-time ray tracing and 3D display walls. The main contributors to the project included a number of optimization experts in Intel Russia, as well as Intel's Performance Library Team. In the early days of OpenCV, the goals of the project were described as:

- Advance vision research by providing not only open but also optimized code for basic vision infrastructure. No more reinventing the wheel.
- Disseminate vision knowledge by providing a common infrastructure that developers could build on, so that code would be more readily readable and transferable.
- Advance vision-based commercial applications by making portable, performance-optimized code available for free—with a license that did not require to be open or free themselves.

The first alpha version of OpenCV was released to the public at the IEEE Conference on Computer Vision and Pattern Recognition in 2000, and five betas were released between 2001 and 2005. The first 1.0 version was released in 2006. In mid-2008, OpenCV obtained corporate support from Willow Garage, and is now again under active development. A version 1.1 "pre-release" was released in October 2008.

The second major release of the OpenCV was on October 2009. OpenCV 2 includes major changes to the C++ interface, aiming at easier, more type-safe patterns, new functions, and better implementations for existing ones in terms of performance (especially on multi-core systems). Official releases now occur every six months and development is now done by an independent Russian team supported by commercial corporations.

In August 2012, support for OpenCV was taken over by a non-profit foundation OpenCV.org, which maintains a developer and user site.

8.2.2 METHODS USED

Functions	Description
cv2.videocapture()	This function is for video capturing from video files, image sequences or cameras where cv2 is OpenCV object.
cv2.flip()	Flips a 2D array around vertical, horizontal, or both axes
cv2.cvtColor()	For grey scale conversion, we use the function cv2.cvtColor(input image, flag) where flag determines the type of conversion
cv2.GaussianBlur()	Blur the images with various low pass filters. In this, Gaussian kernel is used. It is done with the function, cv2.GaussianBlur (). We should specify the width and height of kernel which should be positive and odd
cv2.threshold()	The function used is cv2.threshold. First argument is the source image, which should be a grayscale image. Second argument is the threshold value which is used to classify the pixel values. Third argument is the maxVal which represents the value to be given if pixel value is more than (sometimes less than) the threshold value
Cv2.findContours()	Contours can be explained simply as a curve joining all the continuous points having same colour or intensity. There are three arguments in cv2.findContours() function, first one is source image, second is contour retrieval mode, third is contour approximation method. And it outputs the contours and hierarchy
Cv2.ContourArea()	Contour area is given by the function cv2.contourArea()
Cv2.rectangle()	Draws a simple, thick, or filled up-right rectangle

Cv2.imshow()	Use the function **cv2.imread()** to read an image
Cv2.boundingrect()	It is a straight rectangle, it doesn't consider the rotation of the object. So area of the bounding rectangle won't be minimum. It is found by the function cv2.boundingRect ().
Cv2.convexhull()	Convex Hull will look similar to contour approximation, but it is not (Both may provide same results in some cases). Here, cv2.convexHull () function checks a curve for convexity defects and corrects it. Generally speaking, convex curves are the curves which are always bulged out, or at-least flat. And if it is bulged inside, it is called convexity defects
Cv2.drawContours()	To draw the contours, cv2.drawContours function is used. It can also be used to draw any shape provided you have its boundary points. Its first argument is source image, second argument is the contours which should be passed as a Python list, third argument is index of contours (useful when drawing individual contour.
Cv2.moments()	Image moments help you to calculate some features like centre of mass of the object, area of the object etc. The function cv2.moments() gives a dictionary of all moment values calculated
Cv2.convexityDefects()	Any deviation of the object from this hull can be considered as convexity defect. This function helps to determine the defects.
Cv2.circle()	Draws a circle it has parameters image where circle is drawn, centre of circle, radius of circle, colour of circle
Cv2.line()	Draws a line segment connecting two points. Its parameters are image, first point, second point, thickness, colour
Cv2.putText()	Draws a text string. Parameters are image, text string, font and font-face.
Cv2.imread()	Use the function **cv2.imread ()** to read an image. The image should be in the working directory or a full path of image should be given.

	Second argument is a flag which specifies the way image should be read. cv2.IMREAD_COLOR: Loads a colour image. Any transparency of image will be neglected. It is the default flag. cv2.IMREAD_GRAYSCALE: Loads image in grayscale mode. cv2.IMREAD_UNCHANGED : Loads image as such including alpha channel.
Cv2.destroywindow()	Destroys a window. Parameter is name of window to be destroyed.
Cv2.waitKey()	Waits for a pressed key. Parameters are delay in milliseconds.
Cv2.destroyAllWindows()	Destroys a window.

Table 8.1 OpenCV methods

8.3 NUMPY

NumPy is the fundamental package for scientific computing with Python. It contains among other things: a powerful N-dimensional array object sophisticated (broadcasting) functions tools for integrating C/C++ and Fortran code useful linear algebra, Fourier transform, and random number capabilities. Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined. This allows NumPy to seamlessly and speedily integrate with a wide variety of databases.

8.3.1 Limitations

NumPy's arrays must be views on contiguous memory buffers. A replacement package called Blaze attempts to overcome this limitation.

Algorithms that are not expressible as a vectorized operation will typically run slowly because they must be implemented in "pure Python", while vectorization may increase memory complexity of some operations from constant to linear, because temporary arrays must be created that are as large as the inputs. Runtime compilation of numerical code has been implemented by several groups to avoid these problems; open source solutions that interoperate with NumPy include Scipy.weave, numexpr and Numba.

8.3.2 History

The Python programming language was not initially designed for numerical computing, but attracted the attention of the scientific/engineering community early on, so that a special interest group called matrix-sig was founded in 1995 with the aim of defining an array computing package. Among its members was Python designer/maintainer Guido van Rossum, who implemented extensions to Python's syntax (in particular the indexing syntax) to make array computing easier. An implementation of a matrix package was completed by Jim Fulton, then generalized by Jim Hugunin to become Numeric, also variously called Numerical Python extensions or NumPy. Hugunin, a graduate student at MIT, joined CNRI to work on JPython in 1997 leading Paul Dubois of LLNL to take over as maintainer. Other early contributors include David Ascher, Konrad Hinsen and Travis Oliphant.

A new package called Numarray was written as a more flexible replacement for Numeric

8.3.4 Methods Used

Methods	Description
Numpy.hstack(tup)	Stack arrays in sequence horizontally (column wise).Take a sequence of arrays and stack them horizontally to make a single array. Rebuild arrays divided by hsplit. Tup parameter is sequence of ndarrays that install arrays must have shape except along second axis.
Numpy.zeroes()	Return a new array of given shape and type, filled with zeros Parameters: **shape** : int or sequence of ints Shape of the new array, e.g., (2, 3) or 2. **dtype** : data-type, optional The desired data-type for the array, e.g., numpy.int8. Default is numpy.float64. **order** : {'C', 'F'}, optional Whether to store multidimensional data in C- or Fortran-contiguous (row- or column-wise) order in memory. Return: ndarray Array of zeros with the given shape, dtype, and orde

Table 8.2 NumPy Methods

8.4 PYGLET

Pyglet is a library for the Python programming language that provides an object-oriented application programming interface for the creation of games and other multimedia applications. Pyglet runs on Microsoft Windows, Mac OS X, and Linux; it is released under BSD Licence.

It supports windowed and full-screen operation, and multiple monitors. Images, video, and sound files in a range of formats can be done natively, with more additional capabilities supplied by the optional AVbin plugin, which uses the Libav package to provide support for audio formats including MP3, Ogg/Vorbis, and Windows Media Audio, and video formats such as DivX, MPEG-2, H.264, WMV, and XviD.

8.4.1 Features of Pyglet

No external dependencies or installation requirements. For most application and game requirements, pyglet needs nothing else besides Python, simplifying distribution and installation.

Take advantage of multiple windows and multi-monitor desktops. *Pyglet* allows you to use as many windows as you need, and is fully aware of multi-monitor setups for use with full screen games.

Load images, sound, music and video in almost any format. *Pyglet* can optionally use AVbin to play back audio formats such as MP3, OGG/Vorbis and WMA, and video formats such as DivX, MPEG-2, H.264, WMV and Xvid.

Pyglet is provided under the BSD open-source license, allowing you to use it for both commercial and other open-source projects with very little restriction.

8.4.2 AVBIN

AVbin is a binary release of a cross-platform, thin wrapper around Libav's video and audio decoding library, providing long-term binary compatibility for applications and languages that need it.

AVbin was originally created for the Pyglet project as its media decoding/decompression library, though we hope others find AVbin useful as well

8.4.3 Methods Used

Methods	Description
Pyglet.media.load()	Audio and video files are loaded in the same way, using the pyglet.media.load() function, providing a filename
Pyglet.media.player()	We can implement many functions common to a media player using the Player class. Use of this class is also necessary for video playback. There are no parameters to its construction:
Player.queue(Parameter)	A player will play any source that is "queued" on it. Any number of sources can be queued on a single player, but once queued, a source can never be dequeued (until it is removed automatically once complete). The main use of this queuing mechanism is to facilitate "gapless" transitions between playbacks of media files.
Player.append()	Create a list of players to play a particular playlist.
Player.volume	This state tells the audio level, expressed as a float from 0 (mute) to 1 (normal volume). This can be set at any time.
Player.play	This state begin or resume playback of the current source
Player.pause	This pause playback of the current source.
Player.playing	This state is true if the player is currently playing, False if there are no sources queued or the player is paused. This is read-only (but see the *pause* and *play* methods).

Table 8.3 Pyglet Methods

8.5 MATH

This module is always available. It provides access to the mathematical functions defined by the C standard.

These functions cannot be used with complex numbers; use the functions of the same name from the cmath module if you require support for complex numbers. The distinction between functions which support complex numbers and those which don't is made since most users do not want to learn quite as much mathematics as required to understand complex numbers. Receiving an exception instead of a complex result allows earlier detection of the unexpected complex number used as a parameter, so that the programmer can determine how and why it was generated in the first place.

The following functions are provided by this module. Except when explicitly noted otherwise, all return values are floats.

8.5.1 Methods Used

Methods	Description
Math.sqrt()	Returns the square root of a function
Math.acos()	Return the arc cosine of *x*, in radians

Table 8.4 Math Library Methods

8.6 PYCHARM

8.6.1 Why use PyCharm?

Unlike a regular text editor, PyCharm *knows Python* and is built from the ground up to deal with code, not text. Other editors tend to deal with only simple text matching (e.g. for syntax highlighting) and leave the hard work to separate plugins. With PyCharm, though, the various features share the same language-aware core, so the navigation, completion, validation, code generation, etc. are all aware of the language rules and the full code structure. Another difference between PyCharm and other editors is that PyCharm is developed by a team of people who are still actively working on it full-time. Plugin ecosystems, on the other hand, are often full of no-longer-maintained side projects that don't necessarily mesh well together.

The most common PyCharm features are code navigation (go to file/class/symbol by name, jump to implementation, find usages, etc.), autocomplete, problem detection/auto resolution, refactoring, integrated debugging, integrated unit tests, and clickable stack traces. There quite a bit of depth to these features as well as other secondary features that you discover when using PyCharm for longer.

8.6.2 Why might I not want to use PyCharm?

Every editor is its own beautiful snowflake, so there are plenty of little reasons why you might prefer one editor over another, but there are also some notable deficiencies in PyCharm that are worth mentioning up-front:

The JavaScript support is less powerful than the Python support. The IDE doesn't understand require statements, so it tends to resolve usages by just using naive name matching, and for common names it tends to overestimate the possible references. Also, JavaScript indexing is slow (it takes a few minutes to index webapp when starting PyCharm for the first time), and in some cases working with JS files is slow.

PyCharm isn't an "officially supported" dev setup, so typical steps may not work smoothly with PyCharm, and changes to the dev environment may cause even worse problems in the future. For example, Alan had to refactor the test infrastructure to get unit tests to run inside PyCharm, and they still don't completely work. If you want to take the simple, stable approach, stick with the

command line for everything. Using PyCharm may require a fair amount of tinkering and will add complexity to your dev setup, but will hopefully make you more productive as well.
PyCharm costs money and is not completely open source (although a significant amount of it is open source).
Even if you decide not to use PyCharm, it may provide some inspiration about what to expect from an editor, and you may be able to find (or develop) some equivalent plugins that work for your editor.

8.6.3 What are some unfair criticisms of PyCharm?

Well, these criticisms are at most half-fair.
"IDEs force you to use the mouse." Almost everything in PyCharm is accessible from just the keyboard. Most things you do are "actions", and you can perform any action by name through the keyboard or give a keyboard shortcut to any action (if it doesn't already have one). The main exception is that modifying preferences is usually best done with the mouse.
"IDEs are for beginners who are afraid of the command line." While it's true that PyCharm is probably more beginner-friendly because it gives immediate feedback about things like syntax mistakes, the features being actively developed make it clear that it's mostly targeted at professional software developers. Maybe some people use PyCharm to avoid the command line, but the better approach is to be aware of how to do anything on the command line and to use PyCharm when it makes you more productive. PyCharm even has a built-in terminal so you can use the command line inside its windowing system if you want.
"IDEs are slow and bloated." At least with Python, PyCharm is very rarely slow, and I've never run into memory problems in my 8GB MacBook Pro. Like I said above, though, it can be slow with JavaScript. Also, some operations can be slow the first time while PyCharm builds indexes, then are fast.
"PyCharm forces me to give up the great text editing provided by vim/emacs." PyCharm has a mode that gives emacs key bindings. There's also a plugin called IdeaVim that makes PyCharm text editing feel like vim. However, it's unclear how good these modes are compared to actual vim/emacs.
"PyCharm isn't very extensible or configurable." PyCharm (and the IntelliJ platform in general) is really a combination of plugins built on a core IDE system, and there is an ecosystem of third-

party plugins as well, and you can write your own plugins. It also has many, many settings, including custom keyboard shortcuts, live templates, color schemes, and code style options. Still, it is true that you occasionally run into cases where the implementation details of a feature don't quite do what you want and there's no way to fix it (e.g. auto-generated imports don't match our code style).

CONCLUSIONS AND FUTURE SCOPE

The AIRMOUSE Project is a small effort to reduce the physical communication between the User and the Device.

As the growing use of computers and other electronic devices would mean the growing demand on rapid and quick technical support, this system is carefully designed to fit with the rapid technical support.

This System controls the Music Player and its Functions through Hand Gestures. But in future it can be taken forward to make control over other utilities and applications and improve to reduce Physical Communication between the User and the Device.

AirMouse is designed to accommodate future upgrading and development without building a new system to fit with the growing needs and demands of the system.

REFERENCES

[1] OpenCV usage documentation, http://docs.opencv.org

[2] Convexity Defects and Convex Hulls, http://docs.opencv.org/2.4/modules/imgproc/doc/structural_analysis_and_shape_descriptors.html

[3] Pyglet usage Documentation, https://pypi.python.org/pypi/pyglet

[4] NumPy usage Documentation, https://www.numpy.org/

[5] Math-Python usage Documentation, https://docs.python.org/2/library/math.html

[6] Rhitvij P. and Preksha Pareek (2016), "Event Triggering Using Hand Gesture Using OpenCV", IJECA, Vol-5, Issue-2, PP 15673-15676.

[7] Amardip G. and Binitha Chirakattu (2015), "Virtual Mouse using Hand Gesture and Color Detection", IJCA, Vol-128, No.11, pp 6-10.

[8] Dnayanada R Jadhav and LMRJ Lobo (2015), "Navigation of PowerPoint Using hand Gestures", IJSR, Vol-4, Issue 1 PP 833-837.

[9] Suresh D S and Bhavana I (2014), "Virtual Mouse Implementation using Color Pointer Detection", IJRSSET, Vol-1, Issue-5, PP 23-32.

[10] Sajjad Ur Rahman, Zeenat Afroze and M. Tareq (2014), "Hand Gesture Recognition Techniques For Human Computer Interaction Using OpenCV", IJSRP, Vol-4, Issue-12.

[11] Manoj Makhija and Kapil Goyal (2014), "Control Windows Photo Viewer by Hand Gesture Recognition using MATLAB", Vol-1, Issue-8, PP 198-204.

[12] M. Krekovic, PCeric, T. Dominko, M Ilijias, K Ivancic, V Skolan and J. Sarlija (2012), "A Method for Real time detection of human fall from Video", MIPRO, May 21-25, Opatija, Croatia, PP 1709-1712.

[13] Vladimir I. Pavlovic, Rajeev Sharma and Thomas S. Huang (1997), "Visual Interpretation of Hand Gestures for Human-Computer Interaction: A Review". IEEE, Vol-19, No. 7, PP 677-695.