Hand Gesture Recognition for Immersive Gaming

Submitted in partial fulfilment of the requirements
of the degree, of

B. E. Information Technology

By

Divyam Singh
Prathamesh Khandare
Shahbaz Shaikh

Supervisor(s):
Mr. Ganesh Gourshete
Assistant Professor

Department of Information Technology
Ap.Shah Institute of Technology
(Engineering College)
University of Mumbai
2019-2020

CERTIFICATE

This is to certify that the project entitled **"HAND GESTURE RECOGNITION FOR**

IMMERSIVE GAMING" is a bonafide work of **Divyam Singh (17204007), Prathamesh Khnadare (17204005), Shahbaz Shaikh (17204018)**submitted to the University of Mumbai in partial fulfillment of the requirement for the award of the degree of B.E. in Information Technology.

Mr. Ganesh Gourshete

(Project Guide)

Mr.Kiran Deshpande

(Head of Department)

Dr. Uttam Kolekar

(Principal)

Project Report Approval for B.E.

This project report entitled "**HAND GESTURE RECOGNITION FOR IMMERSIVE GAMING" Divyam Singh (17204007), Prathamesh Khnadare (17204005), Shahbaz Shaikh (17204018)**is approved for the degree of **B.E. in Information Technology.**

Examiners

1.---

2.---

Date:

Place:

Declaration

I declare that this written submission represents my ideas in my own words and where others' ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

(Signature)
Divyam Sing(17204007)

(Signature)
Prathamesh Khnadare (17204005)

(Signature)
Shahbaz Shaikh (17204018)

Date:

Abstract

Over the recent years, Computer Vision has started to play a significant role in Human Computer Interaction. With the development of information technology in our society, we can expect that computer systems will be embedded into our environment. These environments will impose needs for new types of human-computer-interaction, with interfaces that are natural and easy to use. The ability to interact with computerized equipment without need for special external equipment is attractive. With efficient use of available resources, it is possible to track motion of human hand and fingers in real time using a simple web camera. The aim of this project is to enhance the level of immersion in computer games by creating an application that will allow users to interact with the game only using their hands. The proposed application will enable the user to play computer games without the need of either a keyboard or a mouse or any other expensive input devices. The users can play games through various gestures done using only their hands. This is the main background of the project. Therefore, the project aims in replacing the traditional mouse and touch pads with human hand (fingers) to interact with games which would allow more immersion of users and is also cheap to implement.

Contents

List of Figures

List of Tables

List of Abbreviations

Sr. No.	Abbreviation	Expanded form
i	ROI	Region of Interest
ii	RGB	Red, Green, Blue
iii	HSV	Hue, Saturation, Value
iv	UI	User Interface
v	DFD	Data Flow Diagram
vi	FPS	Frame Rate per Second
vii	GUI	Graphical User Interface
viii	USB	Universal Serial Bus

Chapter 1

Introduction

Games are primarily played on computer system using various input devices of which keyboard and mouse are the most often used. While using these devices, it is just tapping the keys which make changes in the game accordingly. There also exist other input devices which actually translate a user’s body movement in to actions in the game (Example Kinect). This makes the playing experience better and makes the user connected to the game. Also, these devices may be expensive to purchase for average user hence not everyone can use it.

An alternative to playing games through traditional input devices is using gestures that system can understand. Hence this topic, “Gesture Recognition for Gaming” comprises of using Gesture recognition for playing games. The Aim of this project is to make a low-cost alternative to the existing systems that users can use on their computers without any hassle. The project will cover games that can be played by mouse or keyboard on any computer system.

1.1 Description

Gesture is defined as “a movement of part of the body, especially a hand or the head, to express an idea or meaning”. Gestures allow individuals to communicate a variety of feelings and thoughts, from contempt and hostility to approval and affection, often together with body language in addition to words when they speak. Gestures can be understood by machines to recognize user’s commands.

Gesture recognition is a topic in computer science and language technology with the goal of interpreting human gestures via mathematical algorithms. Gestures can originate from any bodily motion or state but commonly originate from the face or hand. Gesture recognition enables humans to communicate with the machine and interact naturally without any mechanical devices.

Playing games via gesture makes the experience much deeper and rich making it more preferable way than using the old keyboard and mouse.

The project focuses on cheaper alternatives of gesture control for playing computer games. It incorporates the use of webcam as an input device that will track hand gestures of the user and convert them to game controls using various image processing algorithms.

The project will also emphasize on making the experience hassle free for users and will tune various parameters so that the gesture recognition is lag free.

1.2 Problem definition

Presently, majority of games are played using mouse and keyboard controls which don't provide much depth. The gesture recognition hardware presently available in the market are very costly and hence can't be experienced by everyone. Also, those controllers come with special set of drivers and software which require regular update failing which makes things complicated for users managing them. These controllers also require games specially made for them and can't be used to play all the games (For Example Xbox-Kinect) can only be played on Xbox device with specific games like tennis, cricket etc.

1.3 Motivation

The motivation was to create an object tracking application to interact with the computer and develop a virtual human computer interaction device. Many researchers in the human computer interaction and robotics fields have tried to control mouse movement using video devices. However, all of them used different methods to make a clicking event. A click of the mouse button was implemented by defining a screen such that a click occurred when a user's hand passed over the region.

The main purpose is that the technology we are using will provide equal to or greater experience than touch screens or Kinect but its cost is very less. Without using input devices such as keyboard mouse we can access the computer screen. For example, if there is a laptop whose system is working correctly but its keyboard or mouse are not working properly then it reduces the fun of playing games due to faulty input, in such case we can use gesture recognition to play the games.

Main advantages of using visual input in this context are that visual information makes it possible to communicate with computerized equipment at a distance, without need for physical contact with the equipment to be controlled. Compared to speech commands, hand gestures are advantageous in noisy environments, in situations where speech commands would be disturbing, as well as for communicating quantitative information and spatial relationships. The idea is that the user should be able to control equipment in his environment as he is, and without need for specialized external equipment, such as a remote control.

1.4 Proposed solution

The aim of this project is to enhance the level of immersion in computer games by creating an application that will allow users to interact with the game only using their hands. The proposed application will enable the user to play computer games with their fingers without the need of either a keyboard or a mouse. The application will have an option to tune various background parameters and morphological operations to subtract the environment from the Region of Interest (ROI).

After Extracting the ROI, the contours will be detected and the inputs will be generated from the gesture and mapped to computer games.

Thus, the user can use the applications in different environments because of the presence of various parameters that can eliminate the noise.

1.5 Scope of project

The purpose of this project is to develop interface for human-computer-interaction based on visual input captured by computer vision systems, and to integrate such interfaces to produce actions that replace traditional interfaces based on keyboards, mouse, remote controls, data gloves or speech.

Since the computer technology continuous to grow up, the importance of human computer interaction is enormously increasing. Nowadays most of the mobile devices are using a touch screen technology. However, this technology is still not cheap enough to be used in desktop system. Creating a virtual human computer interaction device such as mouse or keyboard using a webcam and computer vision techniques can be an alternative way for the touch screen. In this study, gesture recognition is implemented in gaming using a regular webcam.

In this system, the inputs are the gesture movement of one person that is captured by a webcam and the output is the appropriate mapping of gestures with games.

It may be worth emphasizing that that the aim is not to recognize the kind of expressive gestures that are tightly coupled to our speech, or sign languages aimed at inter-human communication. The goal is to explore hand gestures suitable for various control tasks in human-machine interaction. Multi-modal interfaces including hand gesture recognition, face and gaze tracking and speech recognition will also be considered.

Chapter 2

Review of literature

The Review of Literature focuses on techniques used for hand gesture recognition and the color spaces used while detecting different colors against the backgrounds.

The survey done by Ibraheem, Noor A., et al in [1] provides a good knowledge about various color models used for color detection. It is a review paper for different color models used as well as the mathematical representation of each with their corresponding advantages and disadvantages along with their comparison and their suitable application area. We will use Red-Green-Blue (RGB) to Hue-Saturation-Value (HSV) color conversion algorithm as HSV is better suited for color recognition because OpenCV algorithms using HSV has very similar visual perception to human perception so if you want to recognize areas of distinct color then HSV is better. The conversion between RGB to HSV takes a bit of time which can be for big resolutions or high Frames per second (FPS) applications a problem.

An approach in [2] narrates skin detection in HSV color space. In order to detect skin from an RGB image it is first converted to HSV as it can be perceived closely as human colors. RGB to HSV conversion is done by using values ranging from 6 to 38 for H and mixtures of different filters to detect skin color. Next step is thresholding where non skin pixels were assigned value 0 and skin pixels as 1. Dilate and Erode of kernel size 5x5 are used to filter out the noise from the image to a certain extent. A median filter with kernel size 3x3 was used to soften the image. Now, only the skin region will appear as white pixels and rest all are represented as black pixels. The OpenCV library was used for image processing and the system was developed in C language.

Image filtering algorithms are needed to filter out the noise which is the main focus of [3]. Filtering is required to reduce the noise and improve the visual quality of the image. It gives a detailed explanation about various image filtering techniques. The proposed system will use mean filtering as it removes noise while preserving the edge which is crucial as we need to detect contours of the hand.

The Adaptive boosting [4] for hand detection and haar classifier algorithm to train the classifier was implemented in a system. It used HSV color model for background subtraction & noise removal, convex hull algorithm for drawing contour around palm and fingertip detection. A laptop webcam of resolution 640*480 pixels was used to capture the live stream. OpenCV library and C++ was used to implement this system.

Boundary detection algorithm of an object was proposed by S.Satoshi, K.Abe in [5]. The algorithm finds a detailed boundary that includes object's outer border also known as 1-component. It also consists of hole-border between the hole and the 1 –component surrounding it directly. This can be modified for detecting convex and concave parts (hulls) of the hand to detect the contours.

Chapter 3

System Analysis

3.1 Functional Requirement:

1. **Gesture Recognition:** Software should automatically recognize the gesture through the laptop inbuilt camera or through external webcam.

2. **Basic Commands implementation using Gesture:** Software should provide user with easily perform certain basic commands through gestures.

3. **Game handling:** Software should be capable of controlling the game through gesture i.e. the game should take input from the camera and through gesture, controls should be performed.

4. **Support various platforms:** Software should run on as many platforms as possible.

3.2 Non-Functional Requirement:

1. **Availability:** Games that support input from mouse or keyboard and does not require joystick as the main frame input provider should be available.

2. **Reliability:** Proper controlling and handling should be provided for providing good gaming experience.

3. **Scalability:** No matter which game is being played by the user, the software must not lag or hang.

4. **Maintainability:** Software should be coded in a way that is easy to understand and maintain.

3.3 Specific Requirements (Hardware and software requirements)

3.3.1 Technical Feasibility

➢ Hardware Requirements

- Processor : 1 gigahertz (GHz) or faster 32-bit or 64-bit processor
- RAM : 1 gigabyte (GB) RAM (32-bit) or 2 GB RAM (64-bit)
- Hard Disk : 16 GB available hard disk space (32-bit) or 20 GB (64-bit)
- Display Driver: DirectX 9 graphics device with WDDM 1.0 or higher driver.
- Web Camera : Capable of capturing true colors.
- Glove : Colored Glove

➢ Software Requirements

- Coding: C++, OpenCV
- Operating System: Windows 7/8/8.1/10

3.3.2 Economic Feasibility

➢ Hardware Requirements

Requirements	Details	Cost
Processor	Core 2 Duo E8400	799/-
RAM	2GB	999/-
Hard Disk	80GB	950/-
Display Driver	DirectX 9	Free
Web Camera	Intex IT-305WC	1240/-
Glove	Any colored glove	60/-

Table 1. Hardware Requirements

Total Cost: 4048/-

➢ Software Requirements

Requirements	Details	Cost
OpenCV Framework	2.4.10 or lower	Open Source
C++ Environment	Microsoft Studio	Free Student Version

Table 2. Software Requirements

Comparative system's price:

Xbox one 360 with Kinect sensor: ₹30799/-

Wii with wireless gesture controller: ₹28,499/-

3.4 Use-Case Diagram and description

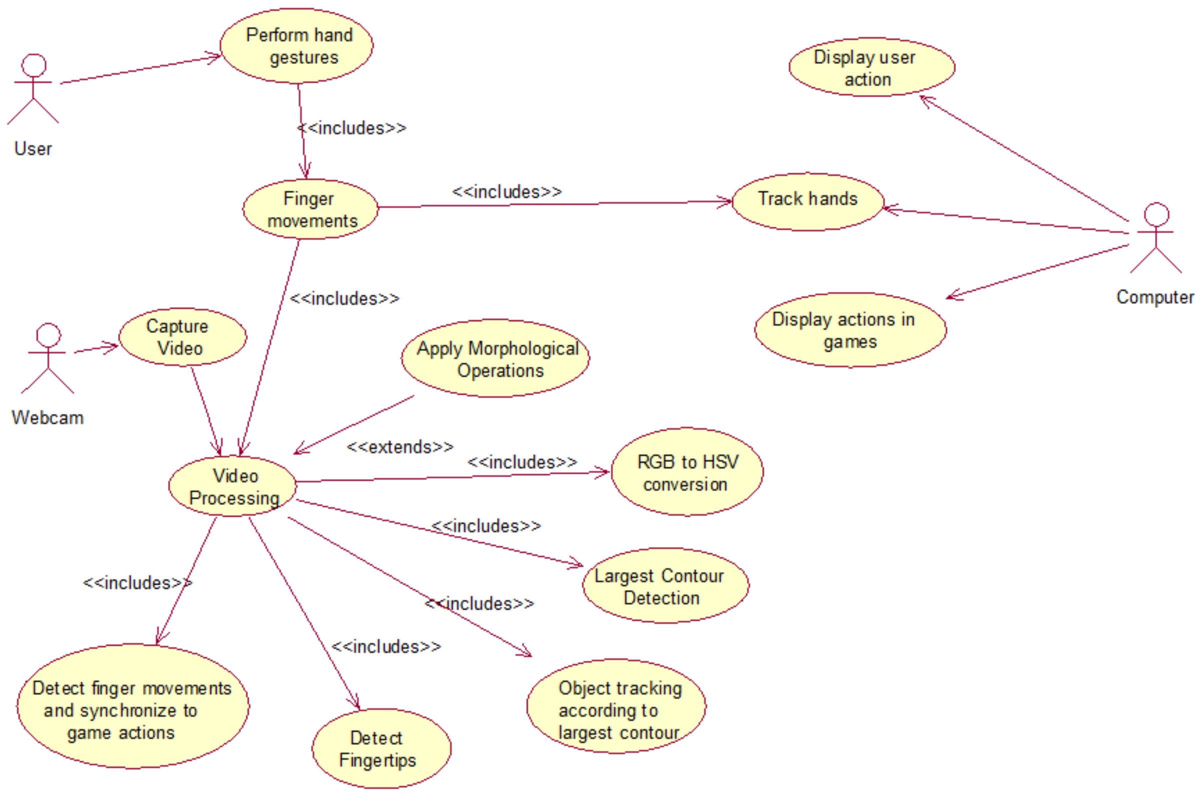


Fig 1. Use Case diagram

Use case diagrams are usually referred to as behavior diagrams used to describe a set of actions (use cases) that some system or systems (subject) should or can perform in collaboration with one or more external users of the system (actors). Each use case should provide some observable and valuable result to the actors or other stakeholders of the system.

Use case diagrams are in fact twofold – they are both behavior diagrams, because they describe behavior of the system, and they are also structure diagrams – as a special case of class diagrams where classifiers are restricted to be either actors or use cases related to each other with associations.

Use case diagrams are used to specify:

- External requirements, required usages of a system under design or analysis (subject) – to capture what the system is supposed to do.
- The functionality offered by a subject – what the system can do.
- Requirements the specified subject poses on its environment – by defining how environment should interact with the subject so that it will be able to perform its services.

Below given is detailed study of the Use-Case Diagram for Gesture Recognition with Immersive Gaming:

Use Case Description of "Perform hand gestures – Finger movements – Track hands"

Use Case	Track Hands
Primary Actor	User
Goal in Context	Allows the User to track finger movements
Preconditions	User performs hand gestures and video processing is active
Trigger	On tracking user, could perform game actions.
Scenario	User observes the User Interface (UI) and performs hand gesture.
Priority	Essential for displaying actions in games.
Secondary Actor	Computer
Exception condition	Unless the video feed in active hands cannot be tracked.

Table 3. Use Case Description 1

Use Case Description of "Capture Video- Video processing"

Use Case	Video processing
Primary Actor	Webcam
Goal in Context	Webcam captures live feed
Preconditions	User must have a webcam
Trigger	On capturing the live feed, it can be send to track hands
Scenario	Webcam captures live feed and can perform various operations
Priority	Every frame captured must be sent for further operations
Secondary Actor	User

Table 4. Use Case Description 2

Use Case Description of “Morphological operations”

Use Case	Morphological operations
Primary Actor	Webcam
Goal in Context	Webcam captures live feed and the images are used for morphological operations
Preconditions	User must have a webcam
Trigger	On performing operations, the pictures could be used to track hands
Scenario	Webcam captures video, video converted to images, images used to perform morphological operations
Priority	Every frame captured must be sent for further operations

Table 5. Use Case Description 3

Use Case Description of “Display user actions – Display actions in Game”

Use Case	Display actions in game
Primary Actor	Computer
Goal in Context	Display Game movements according to hand tracking and gesture recognition
Preconditions	User must have a webcam and video processing is active
Trigger	The appropriate game movement takes place recognized through gestures
Scenario	Webcam captures video, video converted to images, images used to perform morphological operations, which in turn recognizes hand movement and performs game movement
Priority	Every gesture must be recognized
Secondary Actor	User, Webcam

Table 6. Use Case Description 4

Chapter 4

Analysis Modelling

4.1 Activity Diagram:

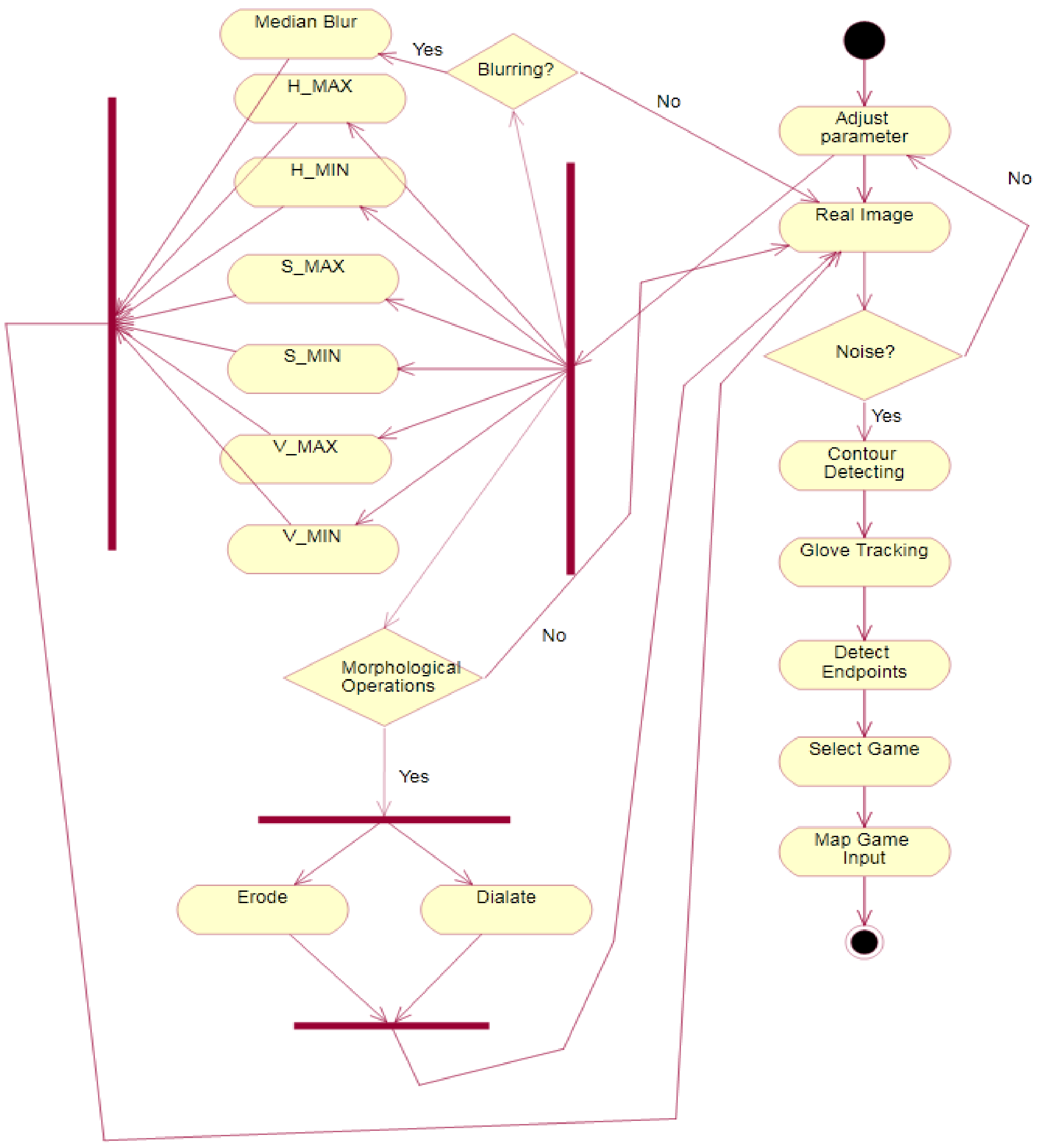


Fig 2. Activity Diagram

4.2 Functional Modelling

In the activity diagram, the system is expressed as various activities in step by step format. First the parameters of picture obtained from the camera is adjusted.

The ways in which the parameters can be adjusted are:

1. The parameters are adjusted Hue-Saturation-value(HSV) color filters. The HSV values can be adjusted by interface available.
2. The second way is using morphological operations on the image i.e. dilation and erosion. Both the HSV and morphological parameters adjusted produces desirable effect.

Now, if the picture contains noise then the parameters are not perfectly adjusted otherwise the next step is detecting the maximum contour in the image. The contour detection is the way of detecting the glove the user wears. Detect end points for finger mapping. Then once the fingers are detected the game controls can be mapped and the game can be played using only hands.

Context level DFD/ Level-0 DFD:

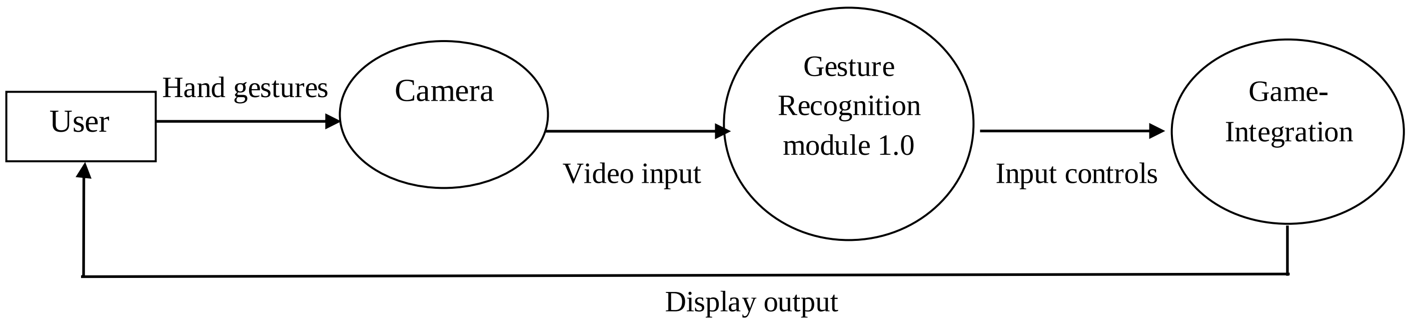


Fig 3. Level-0 DFD

In this **level 0** data flow diagram, the whole system is represented with the help of input, processing and output. The input to the Gesture recognition system is the process of user performing hand gestures in front of camera. The camera provides frame to frame pictures for the gesture recognition module to in turn provide input for Game Integration.

Level 1 DFD:

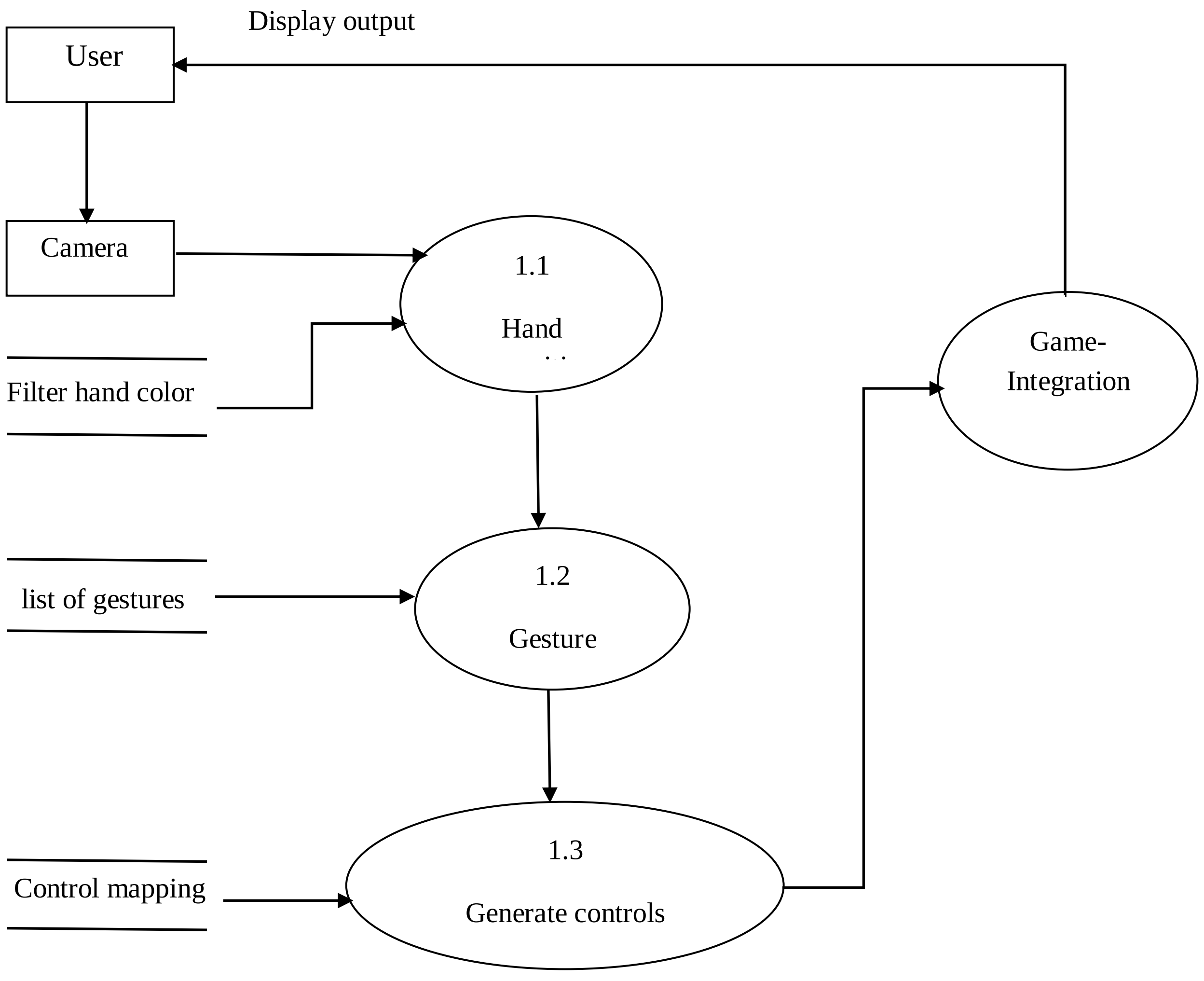
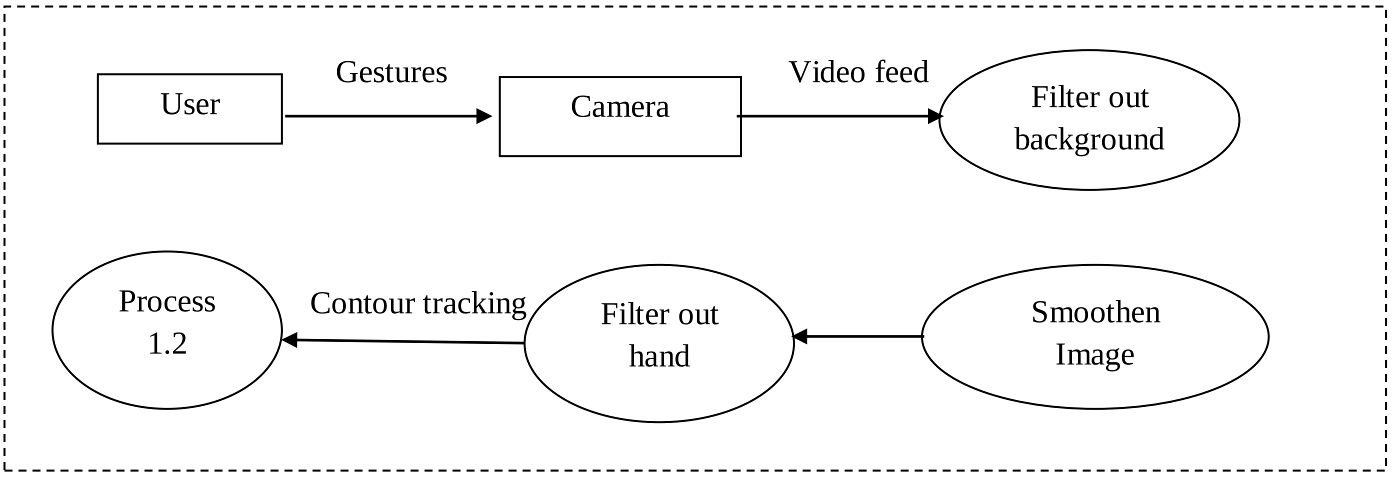


Fig 4. Level-1 DFD

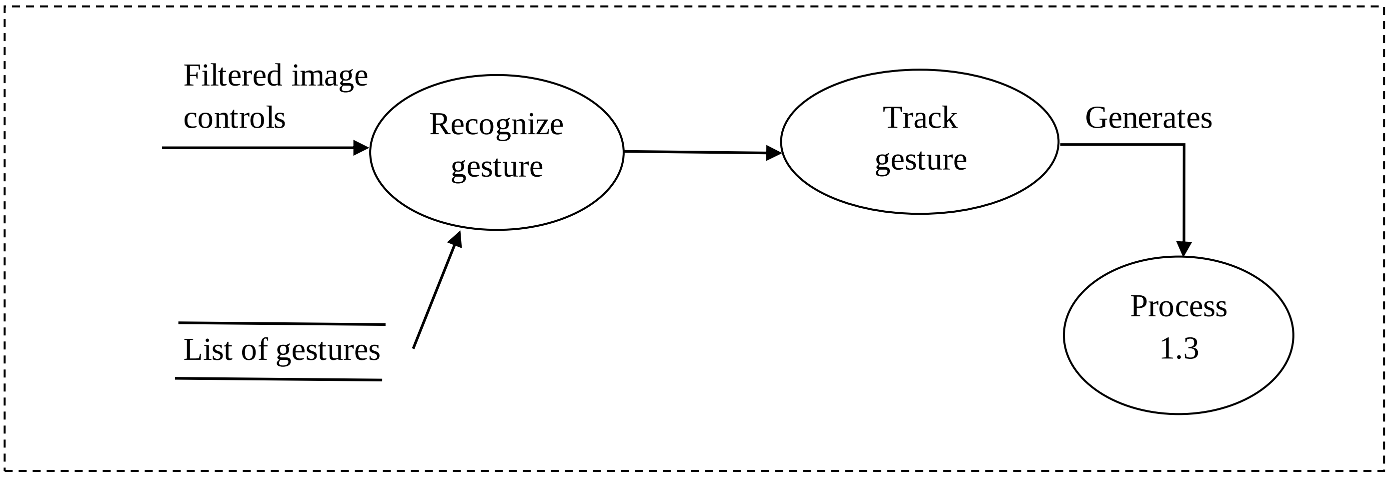
In **level 1** data flow diagram, the gesture recognition module is explained in further details. Initially, the camera provides video feed of user actions. The system after getting feed recognizes the hand in the given frame by distinguishing the hand by color filter. Then once the hand is recognized then the frame is sent to Gesture tracking bubble. The Gesture tracking module checks the required gesture as an input provided by user. The Gesture tracking module contains a database of various gestures stored. The gesture tracking module provides input to the Gesture control module which has a job of control mapping the various gestures recognized by the previous module. Thus, in these processes the gesture is completely recognized and controls are also generated.

Level 2 DFD:

Process 1.1



Process 1.2



Process 1.3

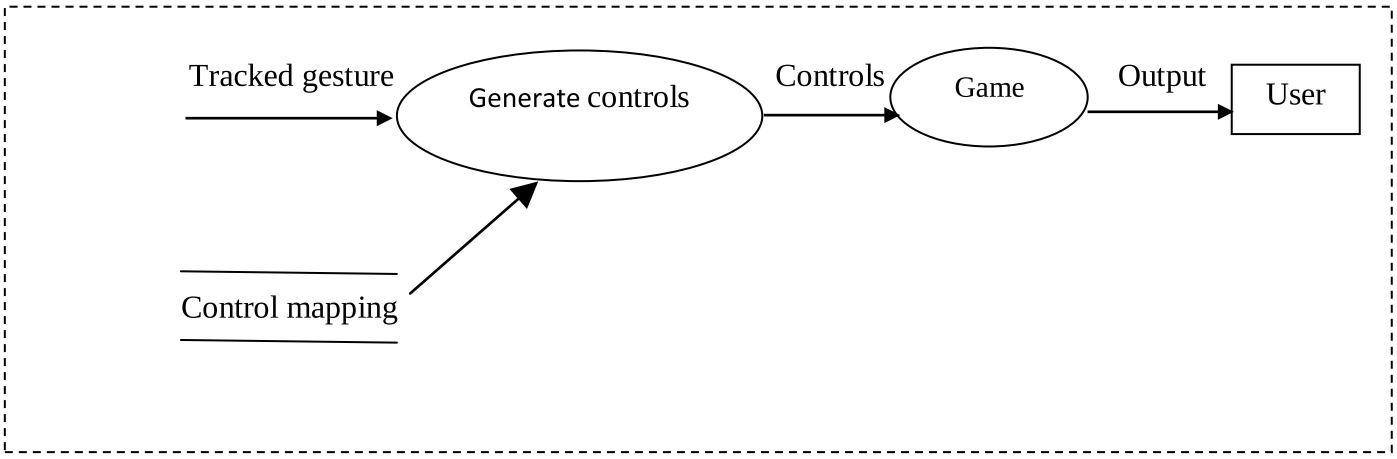


Fig 5. Level-2 DFD

In the **level 2** data flow diagram, the three important processes are explained.

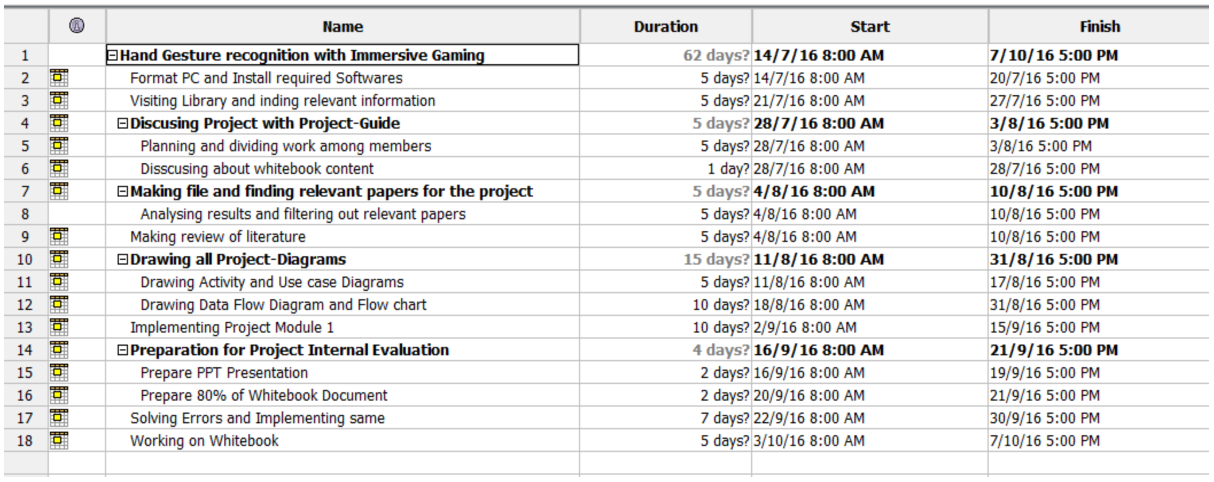
Process 1.1 – Here the video of user performing actions is captured by the camera as a live feed to the system. The video obtained is processed. The processing is done by important processes like filtering out the background and just concentrating on user. This is done through IP algorithms and the image is smoothened. Then the user's hand is recognized through color filter. The hand color is recognized and the contour of that part is tracked.

Process 1.2 - Here the filtered image is taken as input and various gestures are recognized through a database of gestures stored in the system. The tracking of the gestures is done here. The output is the tracked gesture.

Process 1.3 – The tracked gesture is obtained as an input and the control mapping of every gesture is done here. The controls are stored in the database. Once the controls are recognized now the game can be integrated and the output is given to user.

4.3 TimeLine Chart

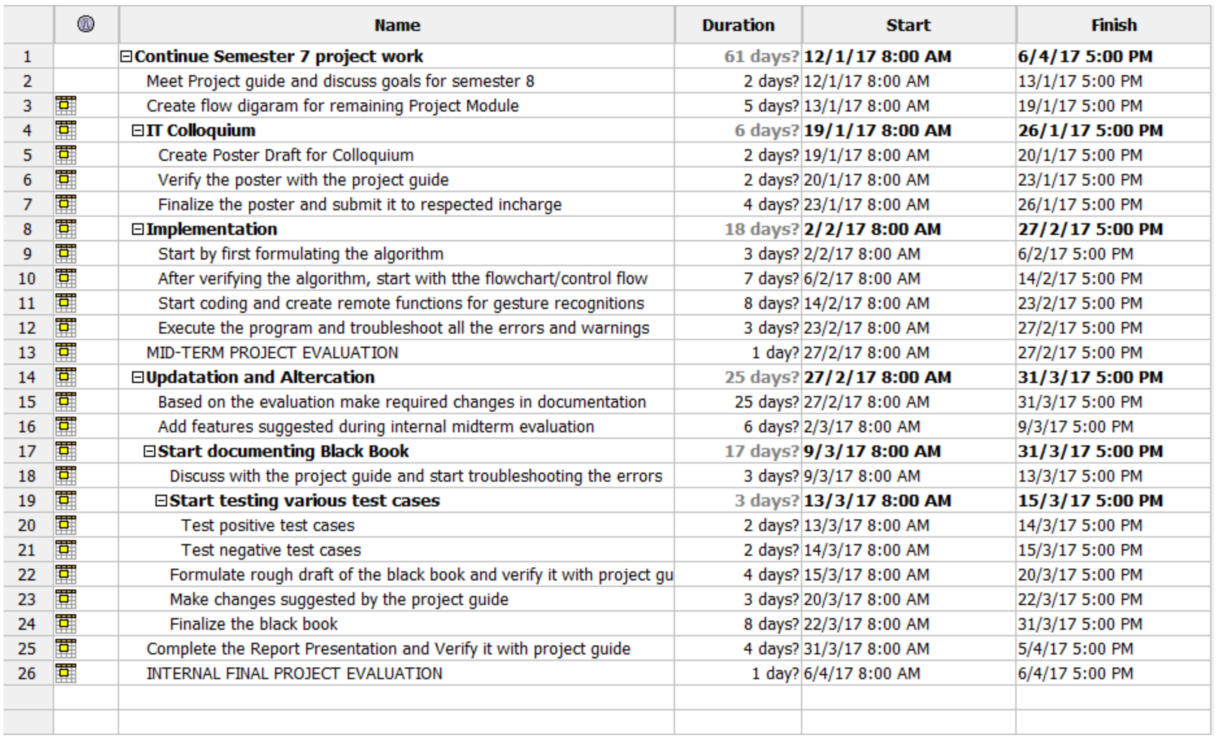
Semester 7 Timeline Chart:



		Name	Duration	Start	Finish
1		⊟**Hand Gesture recognition with Immersive Gaming**	62 days?	**14/7/16 8:00 AM**	**7/10/16 5:00 PM**
2	▦	Format PC and Install required Softwares	5 days?	14/7/16 8:00 AM	20/7/16 5:00 PM
3	▦	Visiting Library and inding relevant information	5 days?	21/7/16 8:00 AM	27/7/16 5:00 PM
4	▦	⊟**Discusing Project with Project-Guide**	5 days?	**28/7/16 8:00 AM**	**3/8/16 5:00 PM**
5	▦	Planning and dividing work among members	5 days?	28/7/16 8:00 AM	3/8/16 5:00 PM
6	▦	Disscusing about whitebook content	1 day?	28/7/16 8:00 AM	28/7/16 5:00 PM
7	▦	⊟**Making file and finding relevant papers for the project**	5 days?	**4/8/16 8:00 AM**	**10/8/16 5:00 PM**
8		Analysing results and filtering out relevant papers	5 days?	4/8/16 8:00 AM	10/8/16 5:00 PM
9	▦	Making review of literature	5 days?	4/8/16 8:00 AM	10/8/16 5:00 PM
10	▦	⊟**Drawing all Project-Diagrams**	15 days?	**11/8/16 8:00 AM**	**31/8/16 5:00 PM**
11	▦	Drawing Activity and Use case Diagrams	5 days?	11/8/16 8:00 AM	17/8/16 5:00 PM
12	▦	Drawing Data Flow Diagram and Flow chart	10 days?	18/8/16 8:00 AM	31/8/16 5:00 PM
13	▦	Implementing Project Module 1	10 days?	2/9/16 8:00 AM	15/9/16 5:00 PM
14	▦	⊟**Preparation for Project Internal Evaluation**	4 days?	**16/9/16 8:00 AM**	**21/9/16 5:00 PM**
15	▦	Prepare PPT Presentation	2 days?	16/9/16 8:00 AM	19/9/16 5:00 PM
16	▦	Prepare 80% of Whitebook Document	2 days?	20/9/16 8:00 AM	21/9/16 5:00 PM
17	▦	Solving Errors and Implementing same	7 days?	22/9/16 8:00 AM	30/9/16 5:00 PM
18	▦	Working on Whitebook	5 days?	3/10/16 8:00 AM	7/10/16 5:00 PM

Fig 6. Semester 7 Timeline Chart

Semester 8 Timeline Chart:



		Name	Duration	Start	Finish
1		⊟**Continue Semester 7 project work**	61 days?	**12/1/17 8:00 AM**	**6/4/17 5:00 PM**
2		Meet Project guide and discuss goals for semester 8	2 days?	12/1/17 8:00 AM	13/1/17 5:00 PM
3	▦	Create flow digaram for remaining Project Module	5 days?	13/1/17 8:00 AM	19/1/17 5:00 PM
4	▦	⊟**IT Colloquium**	6 days?	**19/1/17 8:00 AM**	**26/1/17 5:00 PM**
5	▦	Create Poster Draft for Colloquium	2 days?	19/1/17 8:00 AM	20/1/17 5:00 PM
6	▦	Verify the poster with the project guide	2 days?	20/1/17 8:00 AM	23/1/17 5:00 PM
7	▦	Finalize the poster and submit it to respected incharge	4 days?	23/1/17 8:00 AM	26/1/17 5:00 PM
8	▦	⊟**Implementation**	18 days?	**2/2/17 8:00 AM**	**27/2/17 5:00 PM**
9	▦	Start by first formulating the algorithm	3 days?	2/2/17 8:00 AM	6/2/17 5:00 PM
10	▦	After verifying the algorithm, start with tthe flowchart/control flow	7 days?	6/2/17 8:00 AM	14/2/17 5:00 PM
11	▦	Start coding and create remote functions for gesture recognitions	8 days?	14/2/17 8:00 AM	23/2/17 5:00 PM
12	▦	Execute the program and troubleshoot all the errors and warnings	3 days?	23/2/17 8:00 AM	27/2/17 5:00 PM
13	▦	MID-TERM PROJECT EVALUATION	1 day?	27/2/17 8:00 AM	27/2/17 5:00 PM
14	▦	⊟**Updatation and Altercation**	25 days?	**27/2/17 8:00 AM**	**31/3/17 5:00 PM**
15	▦	Based on the evaluation make required changes in documentation	25 days?	27/2/17 8:00 AM	31/3/17 5:00 PM
16	▦	Add features suggested during internal midterm evaluation	6 days?	2/3/17 8:00 AM	9/3/17 5:00 PM
17	▦	⊟**Start documenting Black Book**	17 days?	**9/3/17 8:00 AM**	**31/3/17 5:00 PM**
18	▦	Discuss with the project guide and start troubleshooting the errors	3 days?	9/3/17 8:00 AM	13/3/17 5:00 PM
19	▦	⊟**Start testing various test cases**	3 days?	**13/3/17 8:00 AM**	**15/3/17 5:00 PM**
20	▦	Test positive test cases	2 days?	13/3/17 8:00 AM	14/3/17 5:00 PM
21	▦	Test negative test cases	2 days?	14/3/17 8:00 AM	15/3/17 5:00 PM
22	▦	Formulate rough draft of the black book and verify it with project gu	4 days?	15/3/17 8:00 AM	20/3/17 5:00 PM
23	▦	Make changes suggested by the project guide	3 days?	20/3/17 8:00 AM	22/3/17 5:00 PM
24	▦	Finalize the black book	8 days?	22/3/17 8:00 AM	31/3/17 5:00 PM
25	▦	Complete the Report Presentation and Verify it with project guide	4 days?	31/3/17 8:00 AM	5/4/17 5:00 PM
26	▦	INTERNAL FINAL PROJECT EVALUATION	1 day?	6/4/17 8:00 AM	6/4/17 5:00 PM

Fig 7. Semester 8 Timeline Chart

Semester 7 Gantt chart:

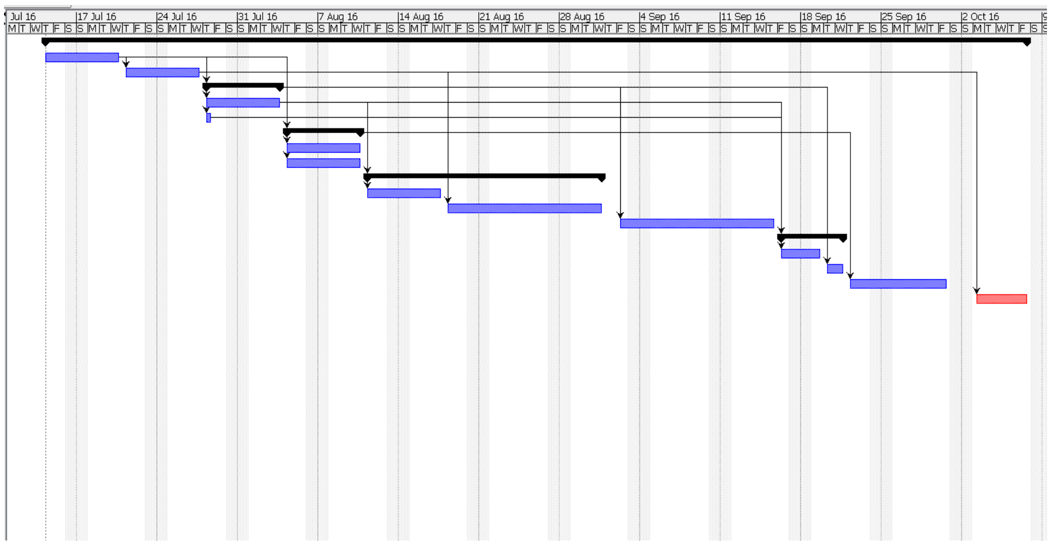


Fig 8. Semester 7 Gantt chart

Semester 8 Gantt chart:

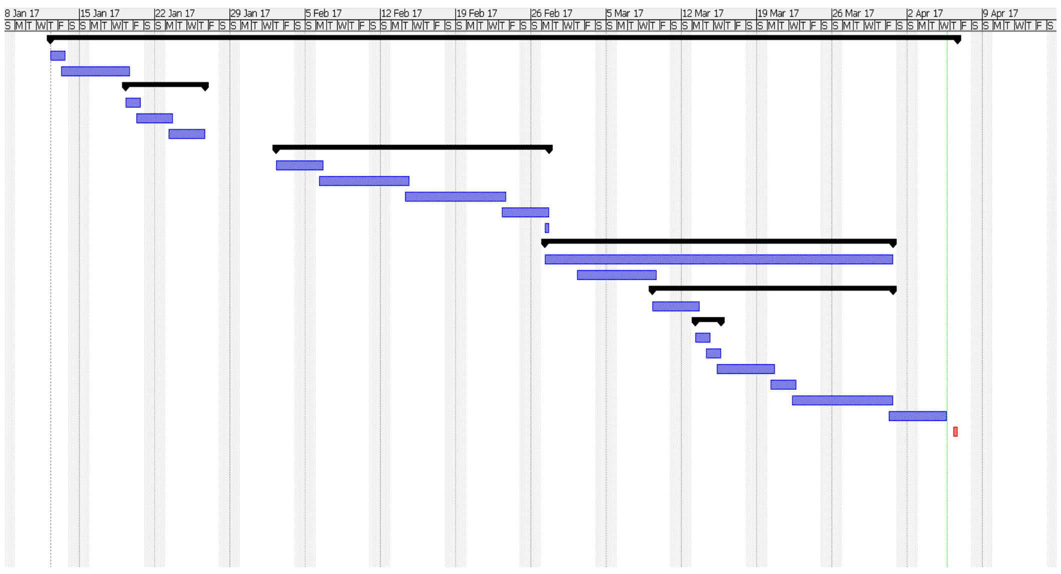
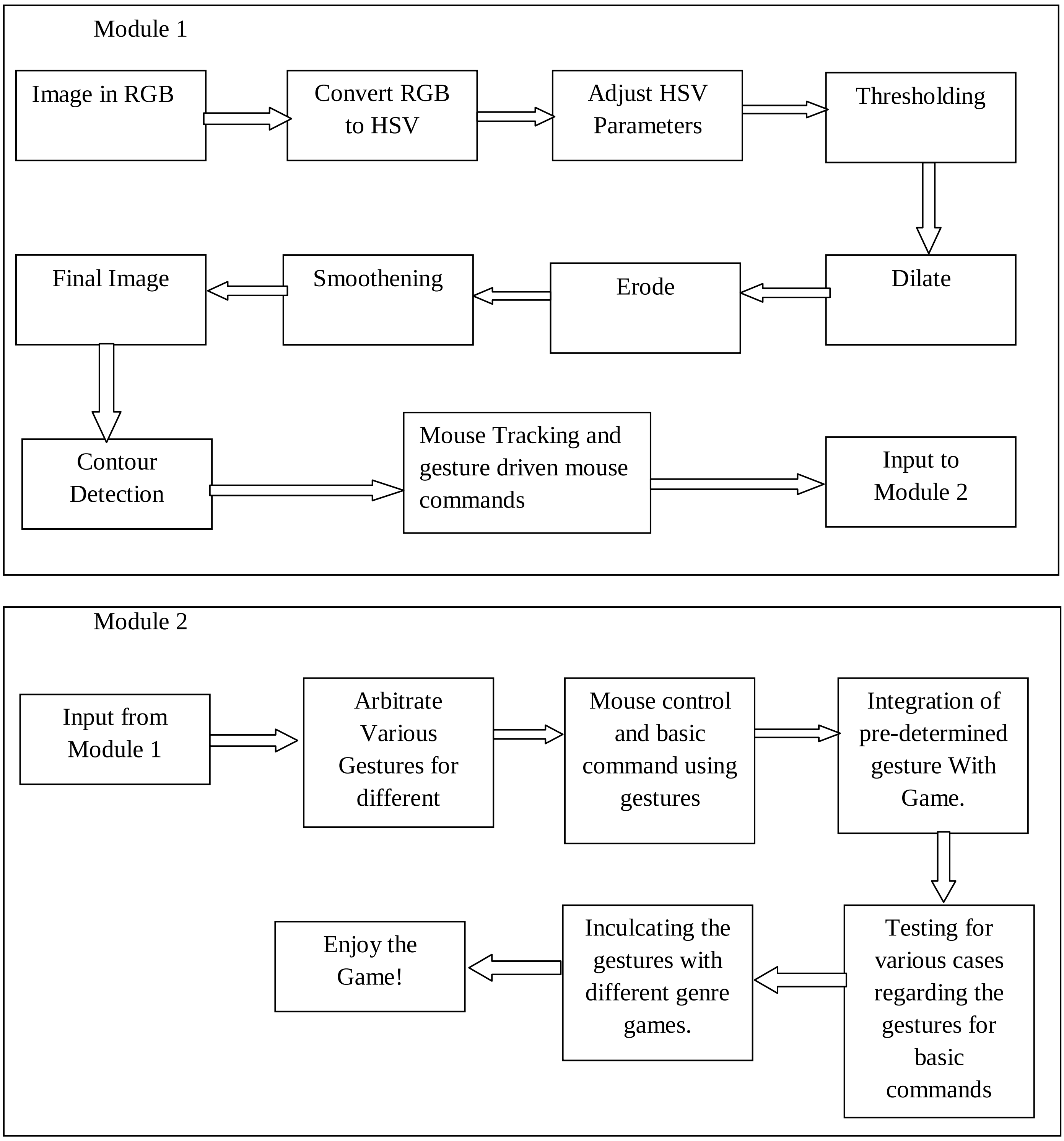


Fig 9. Semester 8 Gantt chart

Chapter 5

Design

5.1 Architectural Design



Module 1
Image in RGB
Convert RGB to HSV
Adjust HSV Parameters
Thresholding
Dilate
Erode
Smoothening
Final Image
Contour Detection
Mouse Tracking and gesture driven mouse commands
Input to Module 2
Module 2
Input from Module 1
Arbitrate Various Gestures for different
Mouse control and basic command using gestures
Integration of pre-determined gesture With Game.
Testing for various cases regarding the gestures for basic commands
Inculcating the gestures with different genre games.
Enjoy the Game!

Fig 10. Project Flow Diagram

Detailed Study of the Module-1 mentioned above is as follow:

1. **Image in RGB:** Image will be accepted as input from the web cam in RGB format.

2. **Conversion of RGB image to HSV:**

   HSV color space is the most suitable color space for color based image segmentation. So, in the proposed application, it is necessary to convert the color space of original image of the video from RGB to HSV image.
   HSV color space is consists of 3 matrices, 'hue', 'saturation' and 'value'. In OpenCV, value range for 'hue', 'saturation' and 'value' are respectively 0-255, 0-255 and 0-255. 'Hue' represents the color, 'saturation' represents the amount to which that respective color is mixed with white and 'value' represents the amount to which that respective color is mixed with black.

3. **Thresholding, Dilate and Erode:**

   Thresholding is the simplest method of image segmentation. From a grayscale image, thresholding can be used to create binary images. Once the image is converted to HSV then it is important to segregate the pixels necessary for the gesture recognition and those that are not useful for the same. In order to do that all the pixel’s values ranging from (MIN, MAX) are replaced by white pixels providing rough image of the actual part required thereby eliminating the non-required part.
   Dilation is one of the basic operations in mathematical morphology. Originally developed for binary images, it has been expanded first to grayscale images, and then to complete lattices. The dilation operation usually uses a structuring element for probing and expanding the shapes contained in the input image.
   Erosion is one of two fundamental operations (the other being dilation) in image processing from which all other morphological operations are based. It was originally defined for binary images, later being extended to grayscale images, and subsequently to complete lattices.

4. Smoothening and Final image:

Image free of all the noise, unwanted pixels or one can call them redundant pixels is referred as smoothen image or final image in this case.

The smoothened image contains only the required part that is highlighted by the program and rest even if present is neglected. This thereby only focuses on the area required whose gesture will be captured for further actions like gesture recognition for mouse controlling or any basic commands using gestures.

5. Mouse Tracking and Gesture driven Mouse commands:

Contour once detected are then used in multiple ways for tracking the mouse cursor using the hand with glove of any RGB color.

The cursor on the screen moves as and when the hand in front of the laptop cam or web camera moves or changes its position.

For example, if the index and middle finger are aligned in proper angle and the one of the finger is tilted slightly, depending on which finger was tilted, left or right mouse click can be performed.

Module-2 being the most complicated and connected module, its detailed study is mentioned in a form of algorithm.

void showimgcontours (Mat & threshedimg, Mat &original)

```
Step 1.   Find Contours using findContours function
Step 2.   Find the convex hull, contours and defects for each contour
Step 3.   Iterate through each contour
Step 4.   Find the area of contour
Step 5.   Store the index of largest contour
Step 6.   Draw Contours using drawContours function
```

5.2 User Interface Design

5.2.1 User-Interface of a Tetris Game:

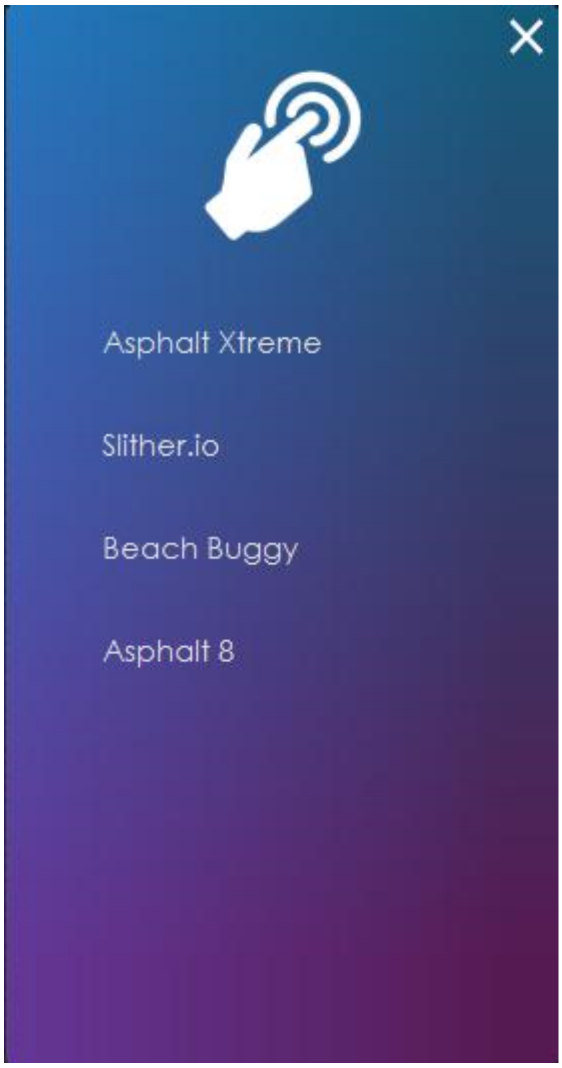


Fig 11. Game Interface

After adjusting the parameters, the contour of the ROI will get detected in this case the glove. This interface will convert the gesture inputs from the web camera into the game inputs using Windows Virtual codes. This interface will be coupled with Track-bar shown in 5.2.2 and Camera feed which will recognize the gestures and the contours.

5.2.2 Track-Bar Interface:

The track-bar interface will be used in adjusting the parameters to convert the webcam video from RGB to HSV color plane and applying morphological operations if needed. This interface provides ranges so that the user can adjust the parameters depending on the background that he/she currently has behind themselves. Thus, it provides more robustness to the application. This interface will also provide the ability to choose the games that the user wants to play with hands.

Chapter 6

Implementation

6.1 Algorithms used

RGB to HSV Conversion:

```
Step1: [Find the max and min values]
        $M = \max(R, G, B)$, $m = \min(R, G, B)$

Step2: [normalized the RGB values to be in the range [0, 1]]
        $r = (M-R)/(M-m)$
        $g = (M-G)/(M-m)$
        $b = (M-B)/(M-m)$

Step3: [Calculate V value]
        $V = \max(R, G, B)$

Step4: [Calculate S value]
        if $M = 0$ then $S = 0$ and $H = 180$ degrees
        if $M <> 0$ then $S = (M - m) / M$

Step5: [Calculate H value]
        if $R = M$ then $H = 60(b-g)$
        if $G = M$ then $H = 60(2+r-b)$
        if $B = M$ then $H = 60(4+g-r)$
        if $H >= 360$ then $H = H - 360$
        if $H < 0$ then $H = H + 360$
        Where H in the range [0,360], S and H in the
        range [0,100]

Step6: [output HSV]
```

Algorithm for Gesture-Recognition and its Integration with Games

```
Step 1.   Start
Step 2.   Declare function prototypes, Boolean toggles and integers storing MIN MAX
          values of HSV and morphological operations.

          Main Function

Step 3.   Create Trackbar to set thresholds for HSV values and morphological operations
Step 4.   Declare VideoCapture Class to capture video
Step 5.   Start video capture
Step 6.   Flip the camera frame 180 degrees for mirror effect
Step 7.   Convert BGR frame to HSV using COLOR_BGR2HSV parameter of cvtColor
          function
Step 8.   Detect the Region Of Interest(ROI) based on HSV range values using inRange
          function
Step 9.   Divide the frame into 6 equisized blocks using line function
Step 10.  If user presses 'm' 2n-1 times then goto Step 16
Step 11.  If user presses 'r' then toggle to real-time frame
Step 12.  If user presses 'c' 2n-1 times then goto Step 17
Step 13.  If user presses 't' 2n-1 times then goto Step 23
Step 14.  If user presses 'k' 2n-1 times then goto Step 46
Step 15.  If user presses 'p' 2n-1 times then goto Step 32 else goto Step 30

           void morphit(Mat &img)

Step 16.  Erode or Dilate the frame using erode() or dilate() with size of structuring element
           determined by the trackbar position set by the user

           void showimgcontours(Mat &threshedimg, Mat &original)

Step 17.  Find Contours using findContours function
Step 18.  Find the convex hull, contours and defects for each contour
Step 19.  Iterate through each contour
```

Step 20. Find the area of contour

Step 21. Store the index of largest contour

Step 22. Draw Contours using drawContours function

void trackFilteredObject(int &x, int &y, Mat threshold, Mat &cameraFeed)

Step 23. Declare two vectors needed for output of findContours

Step 24. Find contours of filtered image using findContours function

Step 25. Use moments to find our filtered object

Step 26. If number of objects greater than MAX_OBJECTS then readjust filter else goto **Step 27**

Step 27. Find the object with largest area by saving a reference area each iteration and compare it to the area in the next iteration

Step 28. Notify the user if the object is found and goto **Step 29**

void drawObject(int x, int y, Mat &frame)

Step 29. Draw crosshairs on the tracked object using line and circle function

void Mouse(int x, int y, Mat &frame)

Step 30. Declare input mouse

Step 31. Move the cursor using specified screen coordinates using SetCursorPos()

void Keyboard(int x, int y, Mat &frame)

Step 32. Declare input keyboard

Step 33. If x co-ordinate is less than 2.25*(FRAME_WIDTH/5) then goto **Step 34**

Step 34. Declare character ‘a’ and goto **Step 41**

Step 35. If x co-ordinate is greater than 2.75*(FRAME_WIDTH/5) then goto **Step 36**

Step 36. Declare character ‘d’ and goto **Step 41**

Step 37. If y co-ordinate is less than (FRAME_HEIGHT/2) then goto **Step 38**

Step 38. Declare character ‘w’ and goto **Step 41**

Step 39. If y co-ordinate is greater than (FRAME_HEIGHT/2) then goto **Step 40**

```
Step 40.  Declare character ‘s’ and goto Step 41
Step 41.  Zero the structure using memset function
Step 42.  Using MapVirtualKeyEx function translate the character into virtual code and
           translate the virtual code into the scan code
Step 43.  Send the key-down event to press the key
Step 44.  Wait for some time
Step 45.  Send the key-up event to release the key
```

void condefects(vector<Vec4i> convexityDefectsSet, vector<Point> mycontour, Mat &original)

```
Step 46.  Find no of fingers using convexityDefectsSet function
Step 47.  If fingers == 1 then goto Step 48
Step 48.  Declare character ‘w’ and goto Step 55
Step 49.  If fingers == 2 then goto Step 50
Step 50.  Declare character ‘s’ and goto Step 55
Step 51.  If fingers == 3 then goto Step 52
Step 52.  Declare character ‘a’ and goto Step 55
Step 53.  If fingers == 4 then goto Step 54
Step 54.  Declare character ‘d’ and goto Step 55
Step 55.  Zero the structure using memset function
Step 56.  Using MapVirtualKeyEx function translate the character into virtual code and
           translate the virtual code into the scan code
Step 57.  Send the key-down event to press the key
Step 58.  Wait for some time
Step 59.  Send the key-up event to release the key


Step 60.  End
```

6.2 Code:

```
#include<iostream>
#include<opencv2\highgui\highgui.hpp>
#include<opencv2\imgproc\imgproc.hpp>
#include<opencv2\core\core.hpp>
#include<opencv2\video\background_segm.hpp>
#include<Windows.h>
#include<XSleep.h>
#include<opencv2\XSleep.cpp>

#pragma comment(lib,"user32")
using namespace cv;
using namespace std;

void on_trackbar(int, void*);
void createTrackbars();
void showimgcontours(Mat &threshedimg, Mat &original);
void toggle(int key);
void morphit(Mat &img);
void condefects(vector<Vec4i> convexityDefectsSet, vector<Point> mycontour, Mat
&frame);
void trackFilteredObject(int &x, int &y, Mat threshold, Mat &cameraFeed);
void drawObject(int x, int y, Mat &frame);
string intToString(int number);
void Keyboard(int x, int y, Mat &frame);
void Mouse(int x, int y, Mat &frame);


bool domorph = false;
bool showchangedframe = false;
bool trackobjstatus = false;
bool showcontours = false;
bool showhull = false;
bool showcondefects = false;
bool ip = false;

int H_MIN = 0;
int H_MAX = 255;
int S_MIN = 0;
int S_MAX = 255;
int V_MIN = 0;
int V_MAX = 255;

int kerode = 1;
int kdilate = 1;


const int MAX_NUM_OBJECTS = 50;

const int FRAME_WIDTH = 640;
const int FRAME_HEIGHT = 480;
const int MIN_OBJECT_AREA = 20 * 20;
const int MAX_OBJECT_AREA = FRAME_HEIGHT*FRAME_WIDTH / 1.5;

Point middle;

int main(void)
{
	createTrackbars();
	on_trackbar(0, 0);
	int x,y;
	Mat frame, hsvframe, rangeframe;
	int key;
	VideoCapture cap(0);
	while ((key = waitKey(1)) != 27)
	{
		toggle(key);
		cap >> frame;

		flip(frame, frame, 180);

		line(frame, Point(2.25*(FRAME_WIDTH/5), 0),
Point(2.25*(FRAME_WIDTH/5), FRAME_HEIGHT), Scalar(0, 255, 0), 2);
	      line(frame, Point(2.75*(FRAME_WIDTH/5), 0), Point(2.75*(FRAME_WIDTH/5),
FRAME_HEIGHT), Scalar(0, 255, 0), 2);
		line(frame, Point(0,FRAME_HEIGHT/2), Point(FRAME_WIDTH,
FRAME_HEIGHT/2), Scalar(0, 255, 0), 2);
		putText(frame, "W"  , Point(309, 120), 1, 2, Scalar(0, 0, 255), 2);
		putText(frame, "W+A", Point(110, 120), 1, 2, Scalar(0, 0, 255), 2);
		putText(frame, "W+D", Point(463, 120), 1, 2, Scalar(0, 0, 255), 2);
		putText(frame, "S"  , Point(309, 360), 1, 2, Scalar(0, 0, 255), 2);
		putText(frame, "S+A", Point(110, 360), 1, 2, Scalar(0, 0, 255), 2);
		putText(frame, "S+D", Point(463, 360), 1, 2, Scalar(0, 0, 255), 2);

		cvtColor(frame, hsvframe, COLOR_BGR2HSV);
		inRange(hsvframe, Scalar(H_MIN, S_MIN, V_MIN), Scalar(H_MAX,
S_MAX, V_MAX), rangeframe);



		if (domorph)
			morphit(rangeframe);

		if (trackobjstatus)
			trackFilteredObject(x, y, rangeframe, frame);

		if (showcontours)
			showimgcontours(rangeframe, frame);

		if (showchangedframe)
			imshow("Camera", frame);
```

```
                else
                        imshow("Camera", rangeframe);

                if(ip && trackobjstatus)
                        Keyboard(x,y,frame);
                else if(!ip && trackobjstatus)
                        Mouse(x,y,frame);
        }
}


void on_trackbar(int, void*)
{
        if (kerode == 0)
                kerode = 1;
        if (kdilate == 0)
                kdilate = 1;
}
void createTrackbars()
{
        String trackbarWindowName = "TrackBars";
        namedWindow(trackbarWindowName, CV_WINDOW_AUTOSIZE);
        createTrackbar("H_MIN",  trackbarWindowName,  &H_MIN,   H_MAX,
on_trackbar);
        createTrackbar("H_MAX",  trackbarWindowName,  &H_MAX,   H_MAX,
on_trackbar);
        createTrackbar("S_MIN",  trackbarWindowName,  &S_MIN,   S_MAX,
on_trackbar);
        createTrackbar("S_MAX",  trackbarWindowName,  &S_MAX,   S_MAX,
on_trackbar);
        createTrackbar("V_MIN",  trackbarWindowName,  &V_MIN,   V_MAX,
on_trackbar);
        createTrackbar("V_MAX",  trackbarWindowName,  &V_MAX,   V_MAX,
on_trackbar);
        createTrackbar("Erode",  trackbarWindowName,  &kerode,  31,    on_trackbar);
        createTrackbar("Dilate", trackbarWindowName,  &kdilate, 31,    on_trackbar);
}


void morphit(Mat &img)
{
        erode(img, img, getStructuringElement(MORPH_RECT, Size(kerode, kerode)));
        dilate(img, img, getStructuringElement(MORPH_RECT, Size(kdilate, kdilate)));
}




void toggle(int key)
{
        if (key == 'm')
```

```
                domorph = !domorph;
        if (key == 'r')
                showchangedframe = !showchangedframe;
        if (key == 't')
                trackobjstatus = !trackobjstatus;
        if (key == 'c')
                showcontours = !showcontours;
        if (key == 'h')
                showhull = !showhull;
        if (key == 'k')
                showcondefects = !showcondefects;
        if (key == 'p')
                ip = !ip;
}


void showimgcontours(Mat &threshedimg, Mat &original)
{
        double largest_area = 0;
        int largest_contour_index = 0;
        vector<vector<Point> > contours;
        vector<Vec4i> hierarchy;



        for (int i = 0; i < contours.size(); i++)
        {
                convexHull(Mat(contours[i]), hull[i], false);
                convexHull(Mat(contours[i]), inthull[i], false);

                if (inthull[i].size()>3)
                        convexityDefects(contours[i], inthull[i], defects[i]);
        }

        for (int i = 0; i< contours.size(); i++)
        {
                double a = contourArea(contours[i], false);
        if (a>largest_area)

                {
                        largest_area = a;
                        largest_contour_index = i;
        }

}



void trackFilteredObject(int &x, int &y, Mat threshold, Mat &cameraFeed)
{

        Mat temp;
```

```
	threshold.copyTo(temp);

	vector< vector<Point> > contours;
	vector<Vec4i> hierarchy;

	double refArea = 0;

	bool objectFound = false;

	if (hierarchy.size() > 0)
	{
		int numObjects = hierarchy.size();
		if (numObjects)
		{
			for (int index = 0; index >= 0; index = hierarchy[index][0])
			{

				if (area > MIN_OBJECT_AREA &&
area<MAX_OBJECT_AREA && area>refArea)
				{
					x = moment.m10 / area;
					y = moment.m01 / area;
					objectFound = true;
					refArea = area;
				}
				else
					objectFound = false;
			}

			if (objectFound == true)
			{
				putText(cameraFeed, "Tracking Glove", Point(0, 50), 2, 1,
Scalar(0, 255, 0), 2);
				drawObject(x, y, cameraFeed);

				middle.x = x;
				middle.y = y;
			}
		}
		else
			putText(cameraFeed, "TOO MUCH NOISE! ADJUST FILTER",
Point(0, 50), 1, 2, Scalar(0, 0, 255), 2);
	}
}


void drawObject(int x, int y, Mat &frame)
{


	circle(frame, Point(x, y), 20, Scalar(0, 255, 0), 2);
```

```
        if (y - 25>0)
                line(frame, Point(x, y), Point(x, y - 25), Scalar(0, 255, 0), 2);
        else
                line(frame, Point(x, y), Point(x, 0), Scalar(0, 255, 0), 2);
        if (y + 25<FRAME_HEIGHT)
                line(frame, Point(x, y), Point(x, y + 25), Scalar(0, 255, 0), 2);
        else
                line(frame, Point(x, y), Point(x, FRAME_HEIGHT), Scalar(0, 255, 0), 2);
        if (x - 25>0)
                line(frame, Point(x, y), Point(x - 25, y), Scalar(0, 255, 0), 2);
        else
                line(frame, Point(x, y), Point(0, y), Scalar(0, 255, 0), 2);
        if (x + 25<FRAME_WIDTH)
                line(frame, Point(x, y), Point(x + 25, y), Scalar(0, 255, 0), 2);
        else
                line(frame, Point(x, y), Point(FRAME_WIDTH, y), Scalar(0, 255, 0), 2);

        putText(frame, intToString(x) + "," + intToString(y), Point(x, y + 30), 1, 1, Scalar(0,
255, 0), 2);

}
void Keyboard(int x,int y,Mat &frame)
{
                INPUT key;
        if(x < (2.25*(FRAME_WIDTH/5)))
        {
                char ch = 'a';
                memset(&key,0,sizeof(INPUT));
                key.type = INPUT_KEYBOARD;
                key.ki.dwExtraInfo = GetMessageExtraInfo();

                for(int i=0;i<12;i++)
                {
                        SendInput(1, &key, sizeof(INPUT));

                        key.ki.dwExtraInfo = GetMessageExtraInfo();
                        SendInput(1, &key, sizeof(INPUT));
                }
        }

        if(x > (2.75*(FRAME_WIDTH/5)))
        {
                char ch = 'd';

                memset(&key,0,sizeof(INPUT));
                key.type = INPUT_KEYBOARD;
                key.ki.dwExtraInfo = GetMessageExtraInfo();

                for(int i=0;i<12;i++)
                {
                        SendInput(1, &key, sizeof(INPUT));
                        key.ki.dwExtraInfo = GetMessageExtraInfo();
                        SendInput(1, &key, sizeof(INPUT));
                }
        }

                if(y < (FRAME_HEIGHT/2))
                {
                char ch = 'w';

                memset(&key,0,sizeof(INPUT));
                key.type = INPUT_KEYBOARD;
                key.ki.dwExtraInfo = GetMessageExtraInfo();

                for(int i=0;i<6;i++)
                {
                        SendInput(1, &key, sizeof(INPUT));
                        key.ki.dwExtraInfo = GetMessageExtraInfo();
                        SendInput(1, &key, sizeof(INPUT));
                }
        }

                if(y > (FRAME_HEIGHT/2))
                {
                char ch = 's';

                memset(&key,0,sizeof(INPUT));
                key.type = INPUT_KEYBOARD;
                key.ki.dwExtraInfo = GetMessageExtraInfo();

                for(int i=0;i<6;i++)
                {
                        SendInput(1, &key, sizeof(INPUT));
                        key.ki.dwExtraInfo = GetMessageExtraInfo();
                        SendInput(1, &key, sizeof(INPUT));
                }
        }

}
void Mouse(int x,int y,Mat &frame)
{

                key.type = INPUT_MOUSE;
                SetCursorPos(1366/400*(middle.x-100),800/200*(middle.y-200));

}

string intToString(int number){
        std::stringstream ss;
        ss << number;
        return ss.str();
}
```

```
void condefects(vector<Vec4i> convexityDefectsSet, vector<Point> mycontour, Mat
&original)
{
        Point2f mycenter;
        float radii;
        long double fingers=0;
        INPUT key;
        minEnclosingCircle(mycontour,mycenter,radii);
        cout << "==start==" << endl;
        for (int cDefIt = 0; cDefIt < convexityDefectsSet.size(); cDefIt++)
        {

                if (depth>11 && ptStart.y<mycenter.y)
                {
                        circle(original, ptStart, 4, CV_RGB(255, 0,0), 4);
                        fingers++;
                }
        }

   if(fingers == 1)
        {
                char ch = 'w';

                memset(&key,0,sizeof(INPUT));
                key.type = INPUT_KEYBOARD;
                key.ki.dwExtraInfo = GetMessageExtraInfo();

                for(int i=0;i<5;i++)
                {
                        key.ki.dwFlags = KEYEVENTF_SCANCODE;
                        SendInput(1, &key, sizeof(INPUT));
                        key.ki.dwExtraInfo = GetMessageExtraInfo();
                }
        }

        putText(original,"Fingers :
"+to_string(fingers),Point(50,100),2,2,CV_RGB(0,255,0),4,8);
}
```

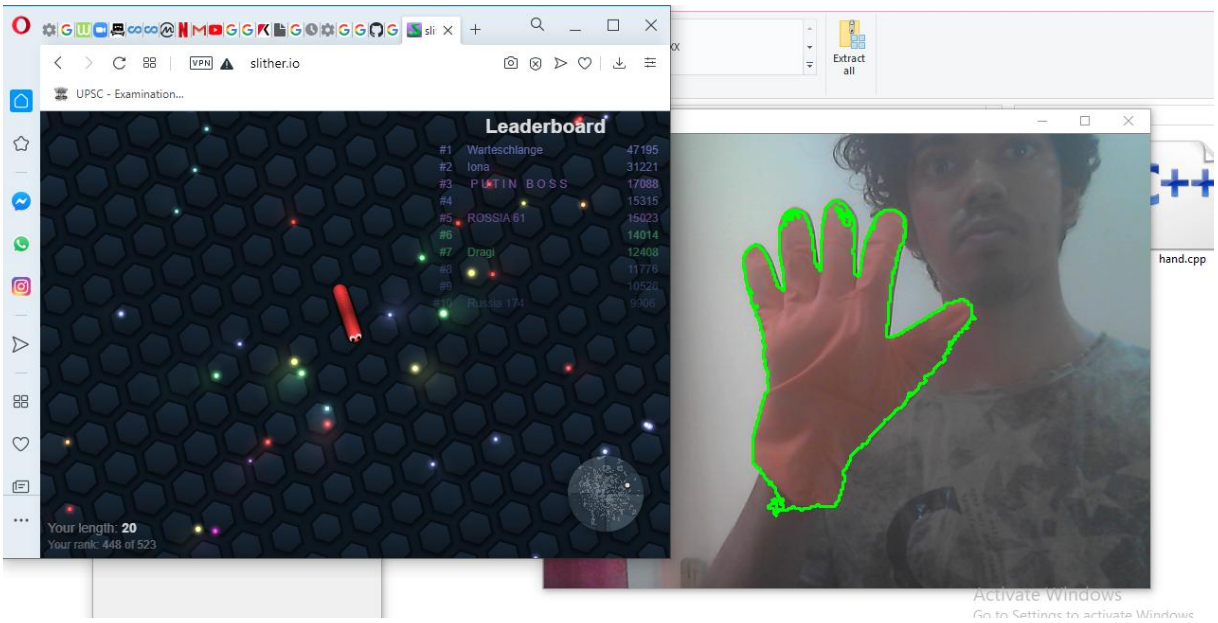
Chapter 7

Testing

7.1 TEST CASES

Test Case ID	01
Revision History	-
Purpose	Test-Case regarding working of project on Online Games- Slither.io
Result	Positive Test-Case
Steps	1. Start the Online-Game (In this case-slither.io) 2. Feed Gesture through camera Input 3. Control the game through movement of hand
Outcome	Project works on Online Games.

Table 7. Test Case 1



Chapter 8

Results and Discussions

While using Sleep() in our code, the application appeared to block. This is because the Sleep() function does not process the message pump, and due to this your application gives the appearance of "hanging" for some time and gives a frame rate of around 10 FPS. Thus we used an alternative function XSleep() which keeps processing the message pump to ensure that all message are posted even while the Sleep is in progress and thus increases the frame rate to around 25 FPS. The XSleep() code uses only Win32 calls, and so it can be used in both MFC and Win32 applications.

Using for loop instead of while in the program is not related to the GUI but the simulation of the keyboard input. We observed that when combination of key inputs were to be simulated i.e W+A, S+A, W+D or S+D using while loop introduced a 300 ms of delay when switching between the inputs whereas for loop didn’t delay the input mapping.

Build-in camera i.e the webcam provided with most of the laptops provides a far better fps rate than the USB plugged-in cameras. The build-in camera provides a frame rate of around 30 FPS while the frame rate drops to around 5 FPS when using a USB camera. This leads to an increase in the latency of keyboard simulation while controlling the games when using USB camera.

Chapter 9

Conclusions

Hand tracking is the main idea behind the implementation of gesture recognition using webcam. The report throws light on all the major aspects of gesture recognition including the concept of morphological operations like erode and dilate, background conversion and filtering.

In this project, we have proposed a vision based hand gesture recognition using a simple system connected with a web camera. Minimum hardware is required to detect hand and counting number of fingers and translate the gestures into commands. People have been trying to use hands as an interface to computers. This application gives user an easy way to interact with computer games with only their hands.

With more robust hand detection, this application can replace the mouse. Future work is required to improve the tracking speed. Future work not only includes improvement of designed strategy but also considering more challenges such as dynamic gestures involving both hands and multiple cameras. Final objective involves gestures with a high degree of freedom; which may require detection of articulated fingers.

Chapter 10

Future Scope

The current system deals with games that have minimal controls so that the system can recognize gestures accurately. The future versions of the system can be designed such that it can be used to control more complex controls while also maintaining the accuracy e.g. – controls in multiple axis, single gestures such that it can mimic multiple controls accurately.

Accuracy and speed of the system is utmost important and hence improving those two areas will always be a considered in any future version. A more efficient way to add and play games that are compatible will be made so that the user is not limited to the presently listed games.

The major enhancement of this system is the implementation of the letting the user select keyboard mapping which will make this system more intuitive and user friendly. This will make this system more useful and efficient.

Appendix

Erosion

In $\mathbb{Z}^d$ binary morphology, an image is viewed as a subset of a Euclidean space $\mathbb{R}^d$ or the integer grid ,for some dimension d.

The basic idea in binary morphology is to probe an image with a simple, pre-defined shape, drawing conclusions on how this shape fits or misses the shapes in the image. This simple "probe" is called structuring element, and is itself a binary image (i.e., a subset of the space or grid).

Let E be a Euclidean space or an integer grid, and A a binary image in E. The erosion of the binary image A by the structuring element B is defined by:

$$A \ominus B = \{z \in E | B_z \subseteq A\},$$

where Bz is the translation of B by the vector z, i.e., $B_z = \{b + z | b \in B\}$, $\forall z \in E$.

When the structuring element B has a center (e.g., a disk or a square), and this center is located on the origin of E, then the erosion of A by B can be understood as the locus of points reached by the center of B when B moves inside A. For example, the erosion of a square of side 10, centered at the origin, by a disc of radius 2, also centered at the origin, is a square of side 6 centered at the origin.

The erosion of A by B is also given by the expression: $A \ominus B = \bigcap_{b \in B} A_{-b}$.

Dilate

In binary morphology, dilation is a shift-invariant (translation invariant) operator, strongly related to the Minkowski addition.

A binary image is viewed in mathematical morphology as a subset of a Euclidean space Rd or the integer grid Zd, for some dimension d. Let E be a Euclidean space or an integer grid, A a binary image in E, and B a structuring element regarded as a subset of Rd.

The dilation of A by B is defined by:

$$A \oplus B = \bigcup_{b \in B} A_b,$$

where A_b is the translation of A by b.

Dilation is commutative, also given by: $A \oplus B = B \oplus A = \bigcup_{a \in A} B_a$.

If B has a center on the origin, then the dilation of A by B can be understood as the locus of the points covered by B when the center of B moves inside A. The dilation of a square of size 10, centered at the origin, by a disk of radius 2, also centered at the origin, is a square of side 14, with rounded corners, centered at the origin. The radius of the rounded corners is 2.

$$A \oplus B = \{z \in E \mid (B^s)_z \cap A \neq \varnothing\}$$

The dilation can also be obtained by:

where Bs denotes the symmetric of B, that is, $B^s = \{x \in E \mid -x \in B\}$

Literature Cited

[1] Ibraheem, Noor A., et al. "Understanding color models: a review" *ARPN Journal of Science and Technology* 2.3 (2012): 265-275.

[2] Oliveira, V. A., and A. Conci. "Skin Detection using HSV color space" *H. Pedrini, & J. Marques de Carvalho, Workshops of Sibgrapi*. 2009.

[3] R.Chandel, G. Gupta. "Image Filtering Algorithms and Techniques: A Review." *International Journal of Advanced Research in Computer Science and Software Engineering* 3.10 (2013): 198-202.

[4] G.R Manish, and P.K Kadbe. "Real time finger tracking and contour detection for gesture recognition using OpenCV" *Industrial Instrumentation and Control (ICIC), 2015 International Conference on*. IEEE, 2015.

[5] S.Satoshi, K.Abe "Topological structural analysis of digitized binary images by border following." *Computer Vision, Graphics, and Image Processing* 30.1 (1985): 32-46.

Acknowledgments

We hereby take the privilege to present our project "**HAND GESTURE RECOGNITION FOR IMMERSIVE GAMING**". We are very grateful to all those individuals for their kind support who have helped in making our endeavour a success.

Firstly, we are highly indebted to our Institute, Ap.Shah institute of technology and the Information technology department for providing us with this learning opportunity along with the required resources to accomplish our task.

We would like to extend our gratitude to our respected Principal **Dr. Uttam Kolekar** and our H.O.D**. Mr.Kiran Deshpande**

We would like to sincerely thank our project guide **Mr. Ganesh Gourshete** for her guidance and constant supervision as well as for providing necessary information concerning the betterment of the project.

We are also grateful to the teaching and non-teaching staff, those who have directly or indirectly aided us to make this report a success.

Divyam Singh

PrathameshKhandare

Shahbaz Shaikh