## UNIVERSTY OF MUMBAI

**PROJECT REPORT ON**

**HAND BASED GESTURE GAME**

**SUBMITTED**

**BY**

# Vishwakarma Nitin Surendra

# Roll no. 11

# Yadav Pankaj Balindar

# Roll no. 12

****

## DTSS COLLEGE OF COMMERCE

### UNDER THE GUIDANCE OF

### MRS.HANEENA SONEY

##### MSCIT SEM-I [2023-2024]



Ref No: Date:

**Certificate**

###### This is to certify that the project entitled **HAND BASED GESTURE GAME** is under taken at the **D.T.S.S COLLEGE OF COMMERCE & SCIENCE** by **Vishwakarma Nitin Surendra.**

###### In partial fulfillment to M. Sc.IT degree (Semester I) Examination had not been submitted for any other examination and does not form part of any other course under gone by the candidate. It is further certified that he has completed all required phases of project.

###### Signature of Internal Guide Signature of External

HOD/In-Charge/Co-Ordinator



Ref No: Date:

**Certificate**

###### This is to certify that the project entitled **HAND BASED GESTURE GAME** is under taken at the **D.T.S.S COLLEGE OF COMMERCE & SCIENCE** by **Yadav Pankaj Balindar.**

###### In partial fulfillment to M. Sc.IT degree (Semester I) Examination had not been submitted for any other examination and does not form part of any other course under gone by the candidate. It is further certified that he has completed all required phases of project.

###### Signature of Internal Guide Signature of External

HOD/In-Charge/Co-Ordinator

**ABSTRACT**

User interaction is an essential feature in the design of an interactive game. Most existing games receive inputs from users via conventional devices such as keyboard, mouse, joystick and paddle. More recent games make use of infrared beams from user's devices, the stylus from touch screen, or pressure-sensing pads to provide rich contextual sensing and interactions. In this paper, we propose the use of hand gestures as the basis for users to directly interact with game objects that are rendered across a flat plasma or lcd display. It forms a new paradigm of interaction in which the physical movements of hands in the form of hand gestures are coordinated along with the virtual objects in the game. Thus, the user effectively becomes a ldquoinput devicerdquo. We make use of a low-cost web camera that is mounted over the gaming screen display to provide image-feed to the hand tracking and gesture recognition system, called germane, which employs the hull-point analysis algorithm for gesture recognition. A working prototype of germane has been developed to validate its operations on several common gestures. Performance evaluation results of germane are also presented.

I

**Acknowledgement**

In completing this project report on project title **Hand Based Gesture Game**. I had to take the help and guideline of a few respected people, who deserve my greatest gratitude.

The completion of this project report gives me much Pleasure. I would like to show my gratitude to **MRS. HANEENA SONEY** for giving me a good guideline for project throughout numerous consultations. I would also like to expand my deepest gratitude to all those who have directly and indirectly guided us in writing this project report.

Many people, especially my classmates and friends themselves, have made valuable comments and suggestions on this proposal which gave me inspiration to improve my project. Here I thank all the people for their help directly and indirectly to complete this project report.

II

## CONTENT

## Chapter No. Title Page No.

## INTRODUCTION……………………………………………………01

## . Background………………………………………………….........02

## . Objective…………………………………………………….........03

## . Purpose & Scopes………………………………………………...04

## SYSTEM ANALYSIS…………………………………………….......05

## 2.1. Existing System………………………………………………..….06

## 2.2. Proposed System…………………………………………….……07

## 2.3. Requirement Analysis……………………………………….……08

## 2.4. Hardware Requirements…………………………………….…….09

## 2.5. Software Requirements……………………………………...........09

## 2.6. Justification of selection of technology…………………….…….10

## SYSTEM DESIGN……………………………………………….…...11

## 3.1. Module Division………………………………………….……....12

## 3.2. Block Diagram....……………………………...…………….……13

## 3.2. Data Flow Diagram...….………………………….………....……14

## 3.3. ER Diagram……………………………………….…...…….........15

## IMPLEMENTATION AND TESTING ………………….………….16

## 4.1. Codes…….………………………….……………..…….……...….16

## 4.1.1. Main.py……………………………………………………16

## 4.1.2. Button.py…………………………………………………..25

## 4.2. Testing Approach.………………………………………..….…….26

## 4.2.1 Unit Testing…………………………………………….…..26

## 4.2.2 Integration Testing…………………………………………26

## 5. RESULT & DISCUSSIONS……………………………………..…….27

## 6. CONCLUSION & FUTURE WORK…………………………..……..30

## 7. REFERENCES…………………………………………………..……..31

III

## 1. INTRODUCTION

To create the program that will perform hand tracking, we will need two Python libraries. These are OpenCV and Media Pipe.

We will use OpenCV to perform operations associated with computer vision. We will use Media Pipe to perform the actual hand detection and tracking on our input image. We will finally need an IDE. For this tutorial, we will use the PyCharm IDE.

This tutorial will be divided into two parts. The first part will focus on how to create a program that does hand tacking. The second part will focus on how to turn the program into a module. A person using Windows, Linux, or macOS can follow through.

Computer technology has tremendously grown over the past decade and has become a necessary part of everyday live. The primary computer accessory for Human Computer Interaction (HCI) is the keyboard. The keyboard is not suitable for HCI in some real-life situations, such as with Human Robot Interaction (HRI). The most natural and intuitive technique for HCI, that is a viable replacement for the computer keyboard is with the use of hand gestures. This system is therefore aimed at investigating and developing a Computer Control (CC) system using hand gestures. In order to harness the full potential of a webcam, it can be used for vision-based CC, which would effectively eliminate the need for a computer keyboard. The usefulness of a webcam can also be greatly extended to other HCI application such as a sign language database or motion controller. Over the past decades there have been significant advancements in HCI technologies for gaming purposes, such as the Microsoft Kinect and Nintendo Wii. These gaming technologies provide a more natural and interactive means of playing video games. Motion controls is the future of gaming and it have tremendously boosted the sales of video games, such as the Nintendo Wii which sold over 50 million consoles within a year of its release. HCI using hand gestures is very intuitive and effective for one-to-one interaction with computers and it provides a Natural User Interface (NUI). There has been extensive research towards novel devices and techniques for cursor control using hand gestures. Besides HCI, hand gesture recognition is also used in sign language recognition, which makes hand gesture recognition even more significant.

Pg. 01

**1.1 BACKGROUND**

In the code above, we use the if statement to check whether a hand is detected. We then use the first for loop to enable us work with one hand at a time.

The second for loop helps us get the hand landmark information which will give us the x and y co-ordinates of each listed point in the hand landmark diagram. This loop will also give us the id of each point.

We will then find the height, width, and channel of our image using the image. Shape function. We finally get the central positions of the identified hand points.

In the code above, we circle the hand point number 20. This is the tip of the pinkie finger.

Feel free to use the number of the hand point you want to circle as they are listed on the hand landmark diagram. We then draw the hand landmarks and the connections between them on the input image.

Pg. 02

**1.2 OBJECTIVES**

Hand tracking **allows us to develop numerous programs that use hand movement and** orientation **as their input**. We tend to write the same code in different projects to perform hand tracking as part of our program. Creating a hand tracking module solves this problem since we write the code once.

First objective of this project is to create a complete system to detect, recognize and interpret the hand gestures through computer vision Second objective of the project is therefore to provide a new low-cost, high speed and color image acquisition system.

Purpose & Scope.

This application can be considered as a starter to the HCI application grounds, therefore large upgradations can be done to it. This application can be extended out to the mouse cursor control also using some more HCI concepts & Open CV Algorithms. Much more accuracy can be obtained with the help of Neural Networks based logics. Tracking performance can be improved to ensure better results. The accuracy of the hand gesture recognition can be improved if the template matching hand gesture recognition method is used with a machine learning classifier. This will take a lot longer to implement, but the accuracy of the gesture recognition will improve.

Hand tracking is the process in which a computer uses computer vision to detect a hand from an input image and keeps focus on the hand's movement and orientation. Hand tracking allows **us to develop numerous programs that use hand movement and orientation as their** input.

Pg. 03

**1.3 PURPOSE & SCOPE**

* **PURPOSE**

Hand gesture-based games purpose is to make minimal use of keyboard and use hand gestures to control or to play the game. The main purpose behind the game is that user will enjoy while playing the game. It will also improve the movement of hand. It becomes user friendly after making the use hand gesture recognition.

* **SCOPE**

This application can be considered as a starter to the HCI application grounds, therefore large upgradations can be done to it. This application can be extended out to the mouse cursor control also using some more HCI concepts & Open CV Algorithms. Much more accuracy can be obtained with the help of Neural Networks based logics. Tracking performance can be improved to ensure better results. The accuracy of the hand gesture recognition can be improved if the template matching hand gesture recognition method is used with a machine learning classifier. This will take a lot longer to implement, but the accuracy of the gesture recognition will improve.

Pg. 04

**2. SYSTEM ANALYSIS**

The background of the implementation lies in the idea to develop an interesting gaming application for the children in which they can get immersed and have a fabulous experience. The aim was to deliver the following features to the user an accurate object recognition to prevent false results. A Negligible response time to react to object movement. The game should be addictive and provide a strong addiction to succeed and stay in the game. Every object class has its own special features that help in classifying the object. Object recognition is that sub-domain of computer vision which helps in identifying objects in an image or video sequence. Computer vision is such sort of research field which attempts to see and speaks to the 3D data for world items. Its fundamental reason for existing is remaking the visual parts of 3D protests in the wake of breaking down the 2D data extricated. Genuine 3D objects are spoken to by 2D pictures. The procedure of item discovery examination is to decide the number, area, size, position of the articles in the input image. With more efficient algorithms, objects can even be recognized even when they are partially obstructed from the direct view. Various approaches to this task have been implemented in the past years. HCI is the domain of computer science which deals with how users interact with their systems and how the user experience and the user interfaces can be optimized to improve user satisfaction. Hence the newer generations must be revealed to the newer technologies & enjoyments.

Pg. 05

**2.1 EXISTING SYSTEM**

### MediaPipe

MediaPipe is a framework mainly used for building audio, video, or any time series data. With the help of the MediaPipe framework, we can build very impressive pipelines for different media processing functions.

This article was published as a part of the **Data Science Blogathon** OpenCV is a library used for computer vision applications. With help of OpenCV, we can build an enormous number of applications that work better in real-time. Mainly it is used for image and video processing.

More information about OpenCV can be acquired here ( <https://opencv.org/> ) Along with OpenCV, we are going to use the MediaPipe library.

#### **MediaPipe:**

MediaPipe is a framework mainly used for building audio, video, or any time series data. With the help of the MediaPipe framework, we can build very impressive pipelines for different media processing functions.

Some of the major applications of MediaPipe.

* Multi-hand Tracking
* Face Detection
* Object Detection and Tracking
* Objectron: 3D Object Detection and Tracking
* AutoFlip: Automatic video cropping pipeline etc.

Basically, the MediaPipe uses a single-shot palm detection model and once that is done it performs precise key point localization of 21 3D palm coordinates in the detected hand region.

The MediaPipe pipeline utilizes multiple models like, a palm detection model that returns an oriented hand bounding box from the full image. The cropped image region is fed to a hand landmark model defined by the palm detector and returns high-fidelity 3D hand key points.

Pg. 06

**2.2 PROPOSED SYSTEM**

The system can be broken down in four main components,

1. Hand Pivot Position Region

2. Hand Position Detection

3. Cursor Control Hand Pivot Position Region: This is the region that will be marked for both the hands of the user to mark the pivot positions for the purpose of making a center point for the movement region of the hands. This region marked for both hands will be used for the entire task of tracking the hand movement gesture for performing the control over the front-end application.

4. Hand Tracking & Gesture Recognition: On the entire system, this is the very important task to be done appropriately. The method used for finding the hand center has the advantage of being simple and easy to implement. The center of the hand is determined from the Hand Pivot Position Region’s center. This center serves as the center of the hand to be tracked for gesture movement.

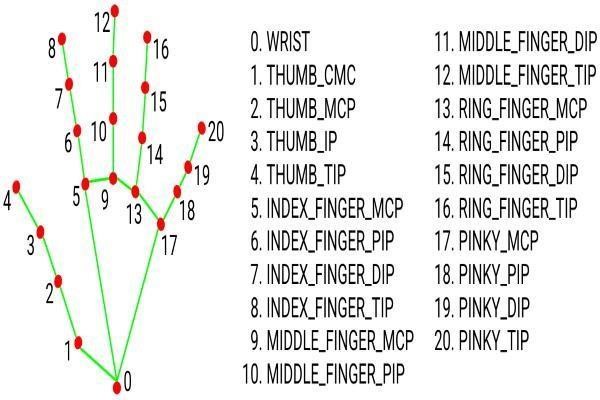
Pg. 07

**2.3 REQUIREMENT ANALYSIS**

Before we jump into coding, let us discuss how MediaPipe performs hand tracking. Hand tracking using MediaPipe involves two stages:

* **Palm Detection:** MediaPipe works on the complete input image and provides a cropped image of the hand.
* **Hand Landmarks Identification:** MediaPipe finds the 21 hand landmarks on the cropped image of the hand.

The 21 hand points that MediaPipe identifies are shown in the image below:



Pg. 08

**2.4 HARDWARE REQUIREMENTS**

* **Processor:** Intel® Core (TM) i5-8265U CPU@ 1.60GHz 1.80GHz
* **RAM:** 4 GB
* **System Type:** 64-bit Operating System, x64-based processor

**2.5 SOFTWARE REQUIREMENTS**

1. Python Software.
2. Libraries Included Python: OpenCV, MediaPipe.

To create the program that will perform hand tracking, we will need two Python libraries. These are **OpenCV and MediaPipe.** We will use OpenCV to perform operations associated with computer vision. We will use MediaPipe to perform the actual hand detection and tracking on our input image.

Pg. 09

**2.6 JUSTIFICATION OF PLATFORM**

|  |  |  |
| --- | --- | --- |
| **Justification** | **Hardware** | **Software** |
| So, the hardware of the project contains the x64 bit processors and 4GB RAM which is supportable by the both the software and it gives a productive and time saving. Which make for a developer to easily do code without memorizing another step. The error detection mechanism in PyCharm its time saving because it is not required to run the code again and again. File format and other settings are easily found able. Python is one of the easy and more efficient for Machine Learning and Artificial Intelligence.  In other hand Python and PyCharm both are supporting html, CSS, JavaScript, bootstrap, jQuery. | 64-bit Operating System, x64-based processor. | Python-3.9.0,  PyCharm IDLE. |

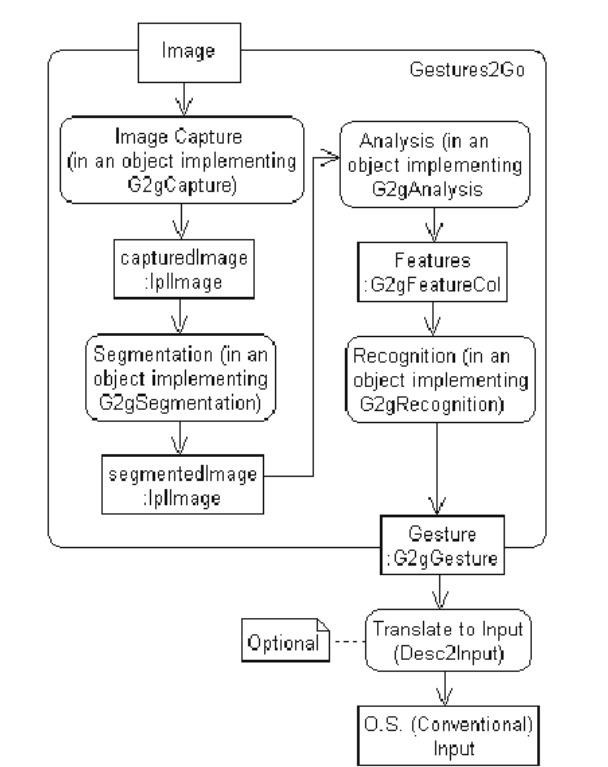
Pg. 10

**3. SYSTEM DESIGNS**

The input design of the system is a very clean and simple approach for the cause of simplicity & endless enjoyment. approach for the cause of simplicity & endless enjoyment. The user just needs to have a PC installed with a Webcam or just a laptop that has a built-in Webcam. The input environment consists of only the User PC System, a properly installed Web Camera system & the Player (user) himself. Except of all this requirement, the user must have a game application (that supports keyboard game input) running alongside of the Hand Gesture Controlled Gaming application. The input to the system comprises of only the user’s hand pivot positions & then the hand movement gestures. The rest of the entire system works in the background silently providing the user an endless enjoyable & addictive gaming experience. The input to the system is the HCI Input i.e. the user’s hand movement gesture. The input to the system is provided through a Webcam that captures continuous frames, creating a video, that tracks the center of the hand which moves away making a gesture. This same video frame is also displayed on the application UI. The webcam continuously captures frames, through which the motion of the user’s hand gesture is tracked, & processed to map it to the keyboard cursor input for the alongside running game application**.**

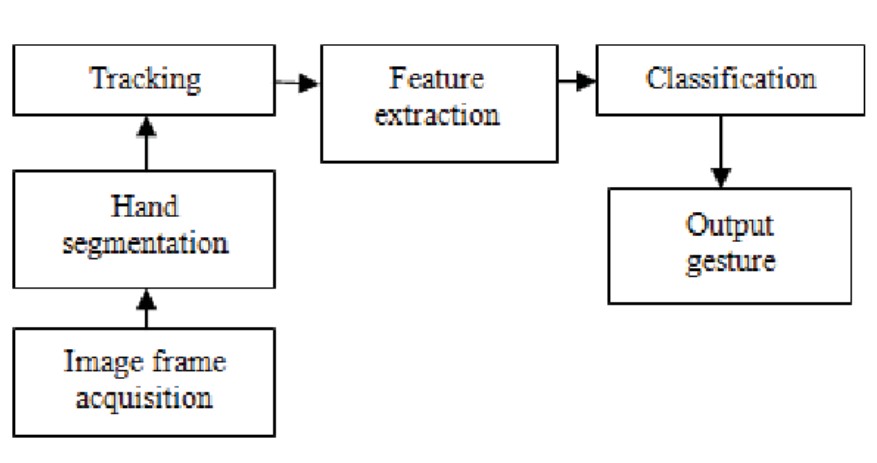
Pg. 11

**3.1 MODULE DIVISION**



Pg. 12

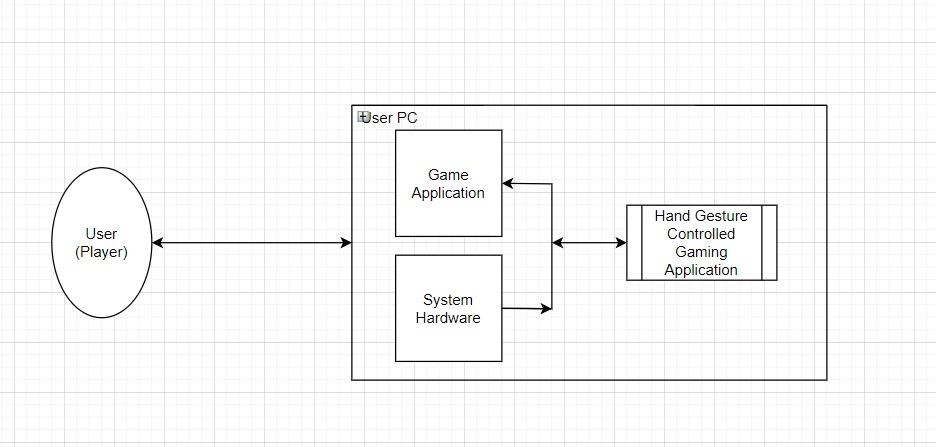
**3.2 BLOCK DIAGRAM**



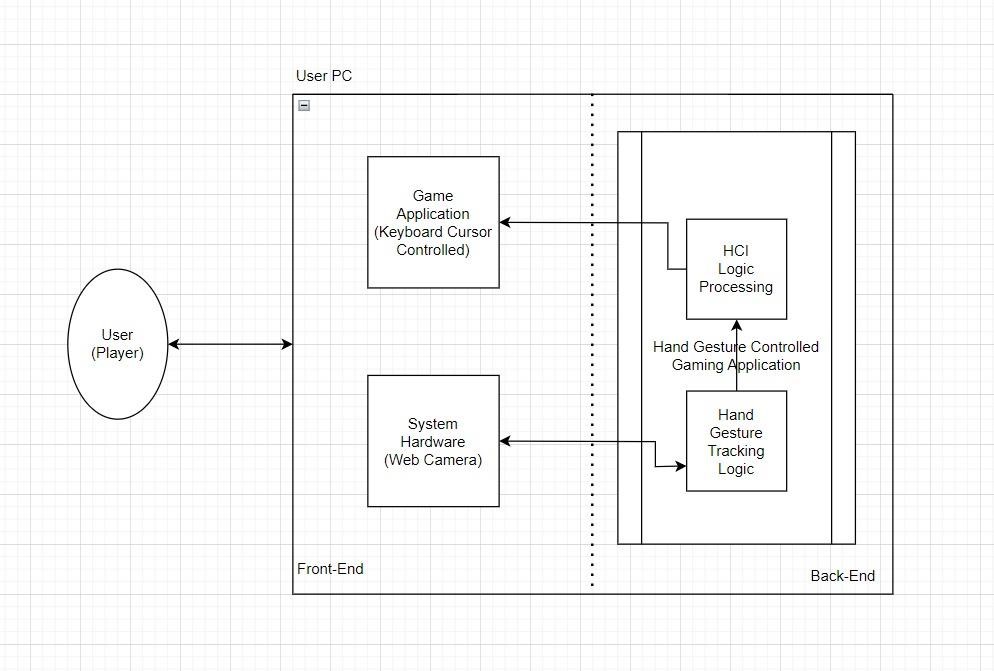
Pg. 13

**3.3 DATA FLOW DIAGRAMS**

### 0 - Level DFD

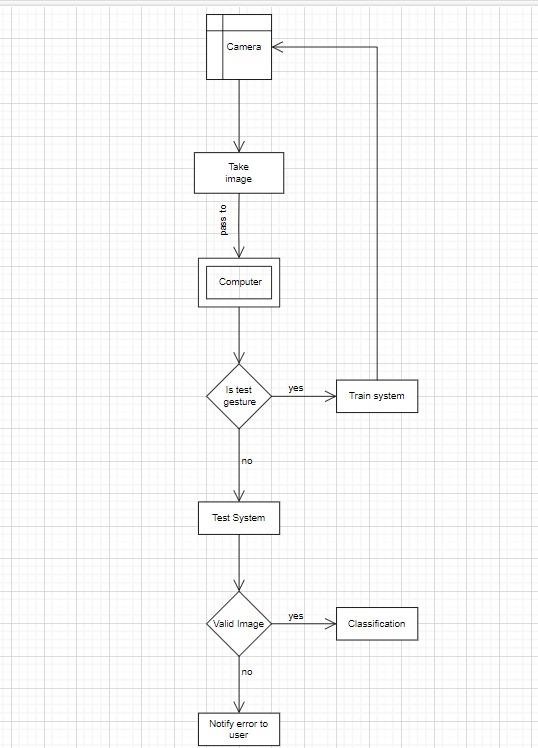


#### **Level DFD**



Pg. 14

**3.4 ER DIAGRAM**



Pg. 15

**4.** **IMPLEMENTATION AND TESTING**

**4.1 Codes**

**4.1.1 Main.py**

import sys

from button import Button

import random

import pygame

import cv2

import NumPy as np

from cvzone.HandTrackingModule import HandDetector

import time

pygame.init()

SCREEN = pygame.display.set\_mode((1280, 720))

pygame.display.set\_caption("Balloon Pop-Up")

BG = pygame.image.load("assets/background.png")

def get\_font(size):  # Returns Press-Start-2P in the desired size

    return pygame.font.Font("assets/vaca.ttf", size)

def balloon():

    # Initialize

    pygame.init()

    sfx\_volume = 0.0

    # Create Window/Display

    width, height = 1280, 720

    window = pygame.display.set\_mode((width, height))

    pygame.display.set\_caption("Balloon Pop Game")

    # Initialize Clock for FPS

    fps = 30

    clock = pygame.time.Clock()

    # Webcam

    cap = cv2.VideoCapture(0)

    cap.set(3, 1280)  # width

    cap.set(4, 720)  # height

    # Images

    imgBalloon = pygame.image.load('assets/BalloonRed.png').convert\_alpha()

Pg. 16

    rectBalloon = imgBalloon.get\_rect()

    rectBalloon.x, rectBalloon.y = 500, 300

    # Variables

    speed = 15

    score = 0

    startTime = time.time()

    totalTime = 60

    # Detector

    detector = HandDetector(detectionCon=0.8, maxHands=1)

    def resetBalloon():

        rectBalloon.x = random.randint(100, img.shape[1] - 100)

        rectBalloon.y = img.shape[0] + 50

    # Main loop

    start = True

    while start:

        OPTIONS\_MOUSE\_POS = pygame.mouse.get\_pos()

        # Apply Logic

        timeRemain = int(totalTime - (time.time() - startTime))

        if timeRemain < 0:

            window.fill((255, 255, 255))

            font = pygame.font.Font('assets/Marcellus-Regular.ttf', 50)

            textScore = font.render(f'Your Score: {score}', True, (50, 50,

255))

            textTime = font.render(f'Time UP', True, (50, 50, 255))

            window.blit(textScore, (450, 350))

            window.blit(textTime, (530, 275))

            image = pygame.image.load('assets/Restart.jpg')

# Set the size for the image

            DEFAULT\_IMAGE\_SIZE = (100, 100)

            # Scale the image to your needed size

            image = pygame.transform.scale(image, DEFAULT\_IMAGE\_SIZE)

            OPTIONS\_Restart = Button(image, pos=(640,560),text\_input="",

                               font=get\_font(60), base\_color="Black",

hovering\_color="Light Green")

            OPTIONS\_Restart.changeColor(OPTIONS\_MOUSE\_POS)

            OPTIONS\_Restart.update(SCREEN)

            cap.release()

Pg. 17

            for event in pygame.event.get():

                    if event.type == pygame.MOUSEBUTTONDOWN:

                        if OPTIONS\_Restart.checkForInput(OPTIONS\_MOUSE\_POS):

                               main\_menu()

           # return "Finished"

        else:

            # OpenCV

            success, img = cap.read()

            img = cv2.flip(img, 1)

            hands, img = detector.findHands(img, flipType=False)

            rectBalloon.y -= speed  # Move the balloon up

            # check if balloon has reached the top without pop

            if rectBalloon.y < 0 :

                resetBalloon()

                speed += 1

            if hands:

                hand = hands[0]

                x, y = hand['lmList'][8][0:2]

                if rectBalloon.collidepoint(x, y):

                    resetBalloon()

                    score += 10

                    speed += 1

            imgRGB = cv2.cvtColor(img, cv2.COLOR\_BGR2RGB)

            imgRGB = np.rot90(imgRGB)

            frame = pygame.surfarray.make\_surface(imgRGB).convert()

            frame = pygame.transform.flip(frame, True, False)

            window.blit(frame, (0, 0))

            window.blit(imgBalloon, rectBalloon)

            font = pygame.font.Font('assets/Marcellus-Regular.ttf', 50)

            textScore = font.render(f'Score: {score}', True, (255,255,255))

            textTime = font.render(f'Time: {timeRemain}', True, (255,255,255))

            window.blit(textScore, (35, 35))

            window.blit(textTime, (1000, 35))

        # Update Display

        pygame.display.update()

Pg. 18

# Set FPS

        clock.tick(fps)

def balloon1():

    # Initialize

    pygame.init()

    # Create Window/Display

    width, height = 1280, 720

    window = pygame.display.set\_mode((width, height))

    pygame.display.set\_caption("Balloon Pop Game")

    # Initialize Clock for FPS

    fps = 30

    clock = pygame.time.Clock()

    # Webcam

    cap = cv2.VideoCapture(0)

    cap.set(3, 1280)  # width

    cap.set(4, 720)  # height

    # Images

    imgBalloon = pygame.image.load('assets/BalloonRed.png').convert\_alpha()

    rectBalloon = imgBalloon.get\_rect()

    rectBalloon.x, rectBalloon.y = 500, 300

    imgBalloon1 = pygame.image.load('assets/blueballoon

(1).png').convert\_alpha()

    rectBalloon1 = imgBalloon1.get\_rect()

    rectBalloon1.x, rectBalloon1.y = 400, 300

    imgBalloon2 =pygame.image.load('assets/green\_balloon.png').convert\_alpha()

    rectBalloon2 = imgBalloon2.get\_rect()

    rectBalloon2.x, rectBalloon2.y = 600, 300

    # Variables

    speed = 15

    score = 0

    startTime = time.time()

    totalTime = 60

    # Detector

    detector = HandDetector(detectionCon=0.8, maxHands=1)

Pg. 19

    def resetBalloon():

        rectBalloon.x = random.randint(100, img.shape[1] - 100)

        rectBalloon.y = img.shape[0] + 50

    def resetBalloon1():

        rectBalloon1.x = random.randint(100, img.shape[1] - 100)

        rectBalloon1.y = img.shape[0] + 50

    def resetBalloon2():

        rectBalloon2.x = random.randint(100, img.shape[1] - 100)

        rectBalloon2.y = img.shape[0] + 50

    # Main loop

    start = True

    while start:

        OPTIONS\_MOUSE\_POS = pygame.mouse.get\_pos()

        # Apply Logic

        timeRemain = int(totalTime - (time.time() - startTime))

        if timeRemain < 0:

            window.fill((255, 255, 255))

            font = pygame.font.Font('assets/Marcellus-Regular.ttf', 50)

            textScore = font.render(f'Your Score: {score}', True, (50, 50,

255))

            textTime = font.render(f'Time UP', True, (50, 50, 255))

            window.blit(textScore, (450, 350))

            window.blit(textTime, (530, 275))

            image = pygame.image.load('Restart.jpg')

# Set the size for the image

            DEFAULT\_IMAGE\_SIZE = (100, 100)

            # Scale the image to your needed size

            image = pygame.transform.scale(image, DEFAULT\_IMAGE\_SIZE)

            # Set a default position

            OPTIONS\_Restart = Button(image, pos=(640,560),

                              text\_input="", font=get\_font(75),

base\_color="Black", hovering\_color="Light Green")

            OPTIONS\_Restart.changeColor(OPTIONS\_MOUSE\_POS)

            OPTIONS\_Restart.update(SCREEN)

            cap.release()

            for event in pygame.event.get():

                    if event.type == pygame.MOUSEBUTTONDOWN:

                        if OPTIONS\_Restart.checkForInput(OPTIONS\_MOUSE\_POS):

Pg. 20

                                main\_menu()

        else:

            # OpenCV

            success, img = cap.read()

            img = cv2.flip(img, 1)

            hands, img = detector.findHands(img, flipType=False)

            rectBalloon.y -= speed  # Move the balloon up

            rectBalloon1.y -= speed

            rectBalloon2.y -= speed

            # check if balloon has reached the top without pop

            if rectBalloon.y < 0 :

                resetBalloon()

                speed += 1

            if rectBalloon1.y < 0 :

                resetBalloon1()

                speed += 1

            if rectBalloon2.y < 0 :

                resetBalloon2()

                speed += 1

            if hands:

                hand = hands[0]

                x, y = hand['lmList'][8][0:2]

                if rectBalloon.collidepoint(x, y):

                    resetBalloon()

                    score += 10

                    speed += 1

                if rectBalloon1.collidepoint(x, y):

                    resetBalloon1()

                    score += 10

                    speed += 1

                if rectBalloon2.collidepoint(x, y):

                    resetBalloon2()

                    score += 10

                    speed += 1

            imgRGB = cv2.cvtColor(img, cv2.COLOR\_BGR2RGB)

            imgRGB = np.rot90(imgRGB)

            frame = pygame.surfarray.make\_surface(imgRGB).convert()

            frame = pygame.transform.flip(frame, True, False)

            imgRGB1 = cv2.cvtColor(img, cv2.COLOR\_BGR2RGB)

            window.blit(frame, (0, 0))

            window.blit(imgBalloon, rectBalloon)

            window.blit(imgBalloon1, rectBalloon1)

Pg. 21

            window.blit(imgBalloon2, rectBalloon2)

            font = pygame.font.Font('assets/Marcellus-Regular.ttf', 50)

            textScore = font.render(f'Score: {score}', True, (255,255,255))

            textTime = font.render(f'Time: {timeRemain}', True, (255,255,255))

            window.blit(textScore, (35, 35))

            window.blit(textTime, (1000, 35))

        # Update Display

        pygame.display.update()

        # Set FPS

        clock.tick(fps)

def play():

    while True:

        PLAY\_MOUSE\_POS = pygame.mouse.get\_pos()

        SCREEN.fill("black")

        SCREEN.blit(balloon())

        PLAY\_BACK = Button(image=None, pos=(640, 460),

                           text\_input="BACK", font=get\_font(75),

base\_color="White", hovering\_color="Green")

        PLAY\_BACK.changeColor(PLAY\_MOUSE\_POS)

        PLAY\_BACK.update(SCREEN)

        for event in pygame.event.get():

            if event.type == pygame.QUIT:

                pygame.quit()

                sys.exit()

            if event.type == pygame.MOUSEBUTTONDOWN:

                if PLAY\_BACK.checkForInput(PLAY\_MOUSE\_POS):

                    main\_menu()

        pygame.display.update()

def options():

    # BG = pygame.image.load("assets/background.png")

    while True:

        OPTIONS\_MOUSE\_POS = pygame.mouse.get\_pos()

        BG = pygame.image.load("assets/background.png")

        SCREEN.fill("white")

        OPTIONS\_Level1 = Button(image=None, pos=(640,200),

                              text\_input="Level 1", font=get\_font(75),

base\_color="Black", hovering\_color="Light Green")

Pg. 22

        OPTIONS\_Level1.changeColor(OPTIONS\_MOUSE\_POS)

        OPTIONS\_Level1.update(SCREEN)

        OPTIONS\_Level2 = Button(image=None, pos=(640,300),

                              text\_input="Level 2", font=get\_font(75),

base\_color="Black", hovering\_color="Light Green")

        OPTIONS\_Level2.changeColor(OPTIONS\_MOUSE\_POS)

        OPTIONS\_Level2.update(SCREEN)

        #SCREEN.blit(OPTIONS\_TEXT, OPTIONS\_RECT)

        OPTIONS\_BACK = Button(image=None, pos=(640, 500),

                              text\_input="BACK", font=get\_font(75),

base\_color="Black", hovering\_color="Light Green")

        OPTIONS\_BACK.changeColor(OPTIONS\_MOUSE\_POS)

        OPTIONS\_BACK.update(SCREEN)

        for event in pygame.event.get():

            if event.type == pygame.QUIT:

                pygame.quit()

                sys.exit()

            if event.type == pygame.MOUSEBUTTONDOWN:

                if OPTIONS\_BACK.checkForInput(OPTIONS\_MOUSE\_POS):

                    main\_menu()

            if event.type == pygame.MOUSEBUTTONDOWN:

                if OPTIONS\_Level1.checkForInput(OPTIONS\_MOUSE\_POS):

                    balloon()

            if event.type == pygame.MOUSEBUTTONDOWN:

                if OPTIONS\_Level2.checkForInput(OPTIONS\_MOUSE\_POS):

                    balloon1()

        pygame.display.update()

def main\_menu():

    while True:

        SCREEN.blit(BG, (0, 0))

        MENU\_MOUSE\_POS = pygame.mouse.get\_pos()

        MENU\_TEXT = get\_font(100).render("MAIN MENU", True, "#b68f40")

        MENU\_RECT = MENU\_TEXT.get\_rect(center=(640, 100))

        PLAY\_BUTTON = Button(image=pygame.image.load("assets/Play Rect.png"),

pos=(640, 250),text\_input="PLAY", font=get\_font(75),

base\_color="#d7fcd4", hovering\_color="White")

Pg. 23

        OPTIONS\_BUTTON = Button(image=pygame.image.load("assets/Options

Rect.png"), pos=(640, 400),text\_input="OPTIONS",

font=get\_font(75), base\_color="#d7fcd4", hovering\_color="White")

        QUIT\_BUTTON = Button(image=pygame.image.load("assets/Quit Rect.png"),

pos=(640, 550),text\_input="QUIT", font=get\_font(75),

base\_color="#d7fcd4", hovering\_color="White")

        SCREEN.blit(MENU\_TEXT, MENU\_RECT)

        for button in [PLAY\_BUTTON, OPTIONS\_BUTTON, QUIT\_BUTTON]:

            button.changeColor(MENU\_MOUSE\_POS)

            button.update(SCREEN)

        for event in pygame.event.get():

            if event.type == pygame.QUIT:

                pygame.quit()

                sys.exit()

            if event.type == pygame.MOUSEBUTTONDOWN:

                if PLAY\_BUTTON.checkForInput(MENU\_MOUSE\_POS):

                    play()

                if OPTIONS\_BUTTON.checkForInput(MENU\_MOUSE\_POS):

                    options()

                if QUIT\_BUTTON.checkForInput(MENU\_MOUSE\_POS):

                    pygame.quit()

                    sys.exit()

        pygame.display.update()

main\_menu()

**For Assets of Balloon, Buttons, Letters and Images click below link 🔗**

**👇**

[**https://drive.google.com/drive/folders/1ClvitSdR6qDNQMj5dyPpNSkLqu6EpWFD?usp=drive\_link**](https://drive.google.com/drive/folders/1ClvitSdR6qDNQMj5dyPpNSkLqu6EpWFD?usp=drive_link)

Pg. 24

**4.1.2 Button.py**

class Button():

    def \_\_init\_\_(self, image, pos, text\_input, font, base\_color, hovering\_color):

        self.image = image

        self.x\_pos = pos[0]

        self.y\_pos = pos[1]

        self.font = font

        self.base\_color, self.hovering\_color = base\_color, hovering\_color

        self.text\_input = text\_input

        self.text = self.font.render(self.text\_input, True, self.base\_color)

        if self.image is None:

            self.image = self.text

        self.rect = self.image.get\_rect(center=(self.x\_pos, self.y\_pos))

        self.text\_rect = self.text.get\_rect(center=(self.x\_pos, self.y\_pos))

    def update(self, screen):

        if self.image is not None:

            screen.blit(self.image, self.rect)

        screen.blit(self.text, self.text\_rect)

    def checkForInput(self, position):

        if position[0] in range(self.rect.left, self.rect.right) and position[1] in range(self.rect.top, self.rect.bottom):

            return True

        return False

    def changeColor(self, position):

        if position[0] in range(self.rect.left, self.rect.right) and position[1] in range(self.rect.top, self.rect.bottom):

            self.text = self.font.render(self.text\_input, True, self.hovering\_color)

        else:

            self.text = self.font.render(self.text\_input, True, self.base\_color)

Pg. 25

**4.2 Testing Approach**

### 4.2.1 Unit Testing

The unit test focused on the internal processing logic. All statements in a module have been exercised at least once. The interface module was tested to ensure that information properly flowed into and out of the program unit under test. All the buttons in the game are working properly.

Both the levels are working properly. Balloon are properly bursting by the Finger.

All the modules are working properly and the game is working smoothly.

#### **4.2.2 Integration Testing**

Integration testing is a technique for constructing the software architecture and conducting tests to uncover errors with interface. The objective of testing was to crosscheck for components fully functional or not according to design. Thus, I integrated all my unit components and saw if the system worked as a whole properly or not.

The information flows between the components were checked once again.

* **RECOVERY TESTING:**

System fails in many ways but recovery must be properly performed. For example, when a person trains the system with his gestures and the system fails to do so, it gives an error message indicating that the system was not properly trained and it keeps on doing so until it gets a valid gesture (that’s acceptable to the system for differentiation). When the system is properly trained, only then we can expect that it will give us accurate results.

* **SENSITIVITY TESTING:**

Invalid input classes that may cause instability or improper processing. It was found during sensitivity testing that if the system was once fully trained for all the gesture types, it gave accurate results, otherwise if it were just trained for a single or two gestures and then tested, it performs erroneous processing.

Pg. 26

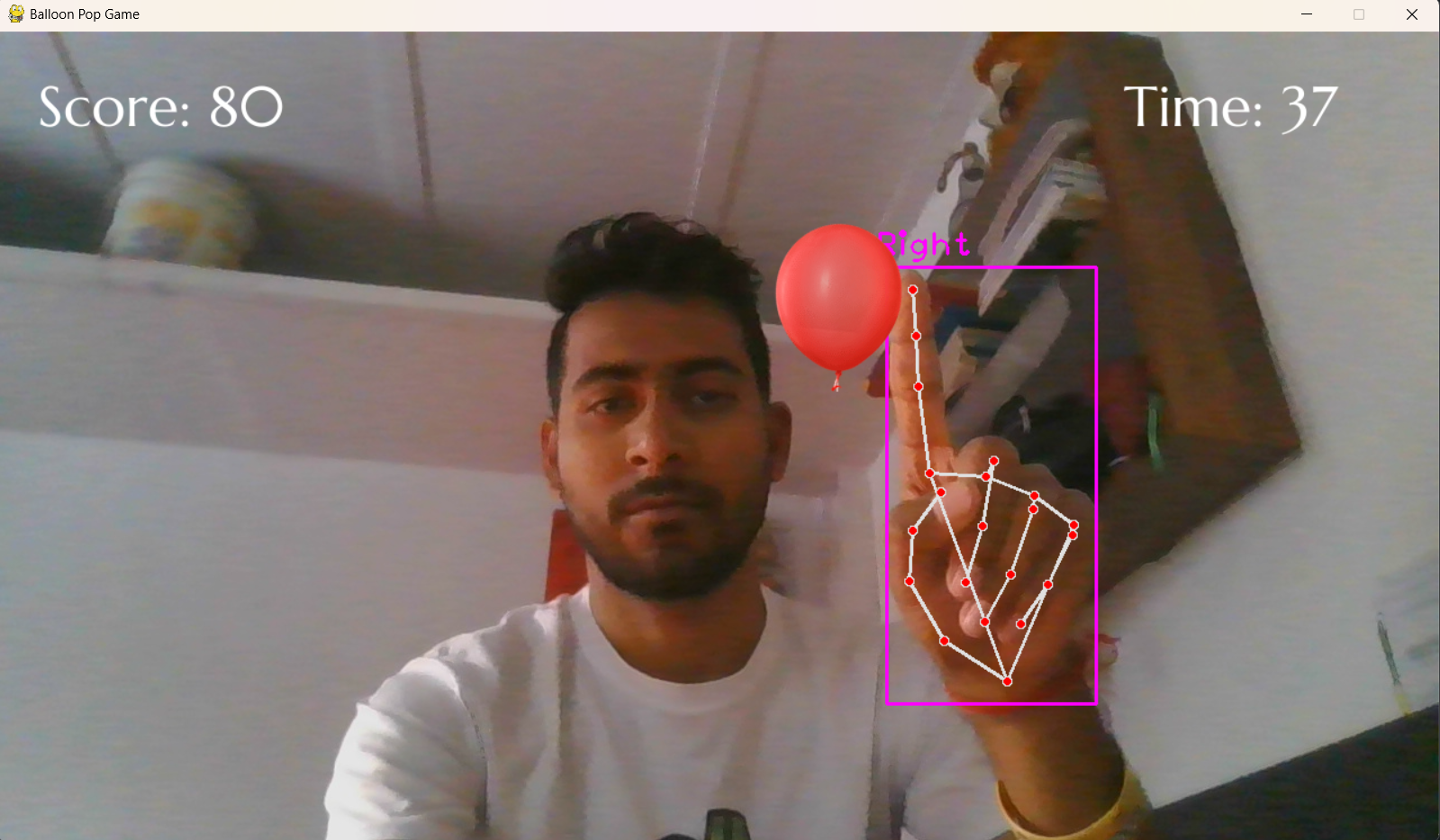
**5. RESULTS AND DISCUSSIONS**

The output will be shown live score. The major constraint of the system is that it must be operated in a well-lit environment. This is the main reason why the system cannot completely replace the computer keyboard, since it is very common for computers to be used in outdoor environments with poor lighting condition.

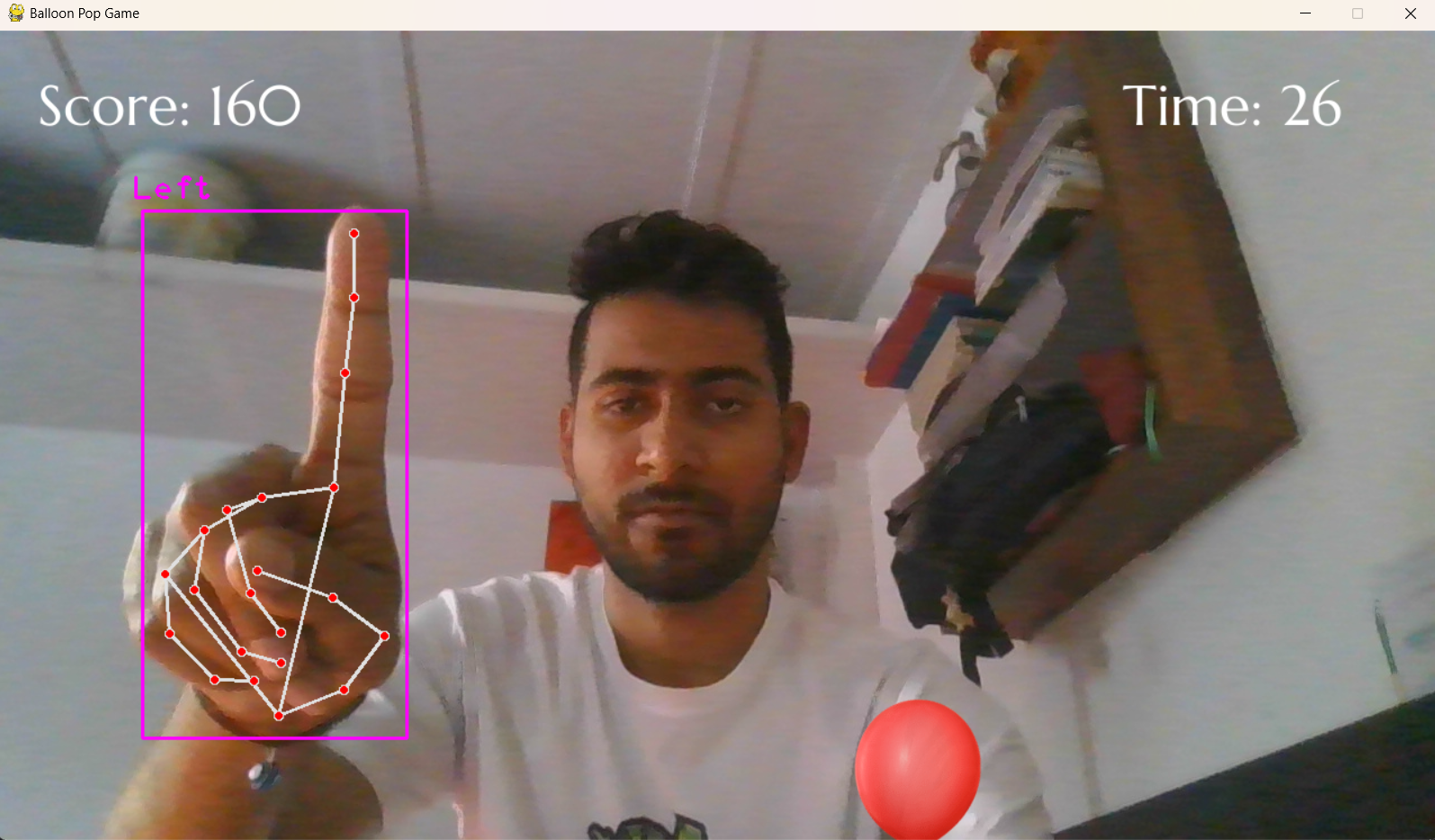


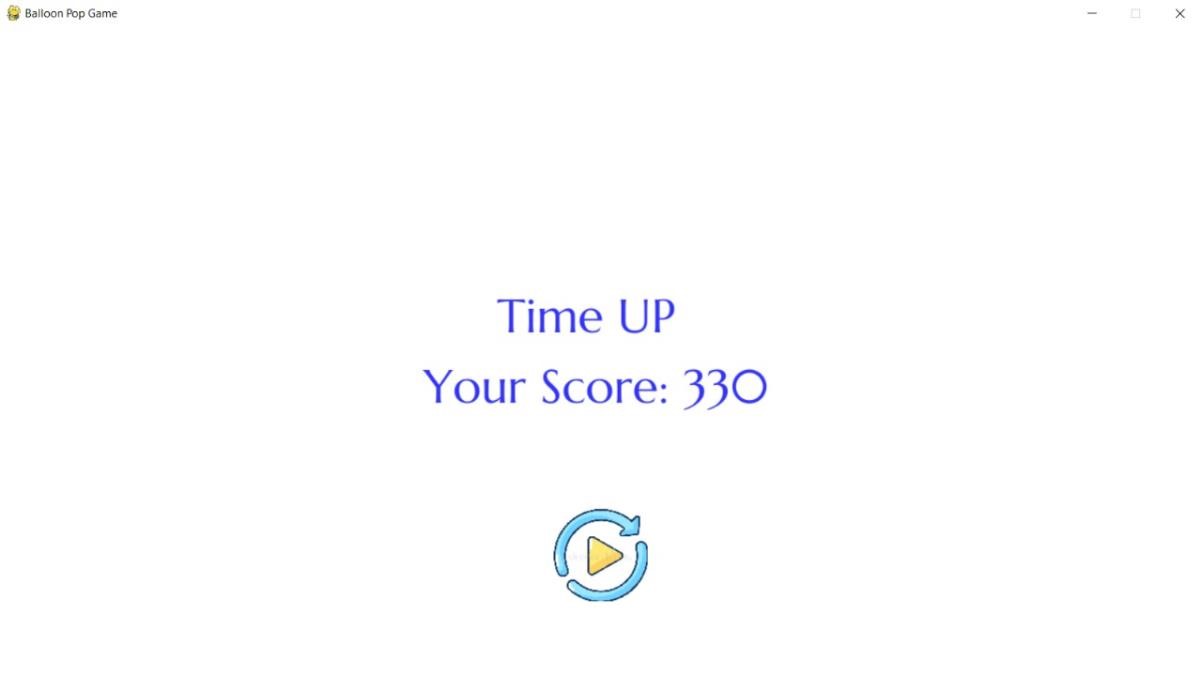
Pg. 27





Pg. 28





Pg. 29

**6. CONCLUSION AND FUTURE WORK**

* **CONCLUSION**

The machine vision-based keyboard cursor control using hand gesture system is developed in the Python language, using the Open CV library. The system is able to control the movement of a keyboard cursor by tracking the user’s hand for playing a game. The keyboard cursor functions will perform by using different hand gestures. The system has the potential of being a viable replacement for the computer keyboard, however due to the constraints encountered; it cannot completely replace the computer keyboard. The accuracy of the hand gesture recognition can be improved, if the template matching hand gesture recognition method is used with a machine learning classifier. Open CV mostly stretches towards real-time vision applications and takes advantage of MMX (Multimedia Extension) and SSE (Streaming SIMD Extensions) instructions when available.

* **FUTURE WORK**

This application can be considered as a starter to the HCI application grounds, therefore large upgradations can be done to it. This application can be extended out to the mouse cursor control also using some more HCI concepts & Open CV Algorithms. Much more accuracy can be obtained with the help of Neural Networks based logics. Tracking performance can be improved to ensure better results. The accuracy of the hand gesture recognition can be improved if the template matching hand gesture recognition method is used with a machine learning classifier. This will take a lot longer to implement, but the accuracy of the gesture recognition will improve.

Pg. 30

**7. REFERENCES**

* [**Www.google.com**](http://Www.google.com)
* [**Www.Youtube.com**](http://Www.Youtube.com)
* [**https://pypi.org/**](https://pypi.org/)
* [**https://mediapipe.org/**](https://mediapipe.org/)
* [**https://www.geeksforgeeks.org/**](https://www.geeksforgeeks.org/)
* [**https://www.simplilearn.com/computer-vision-article**](https://www.simplilearn.com/computer-vision-article)
* [**https://github.com/**](https://github.com/)
* [**https://learnopencv.com/**](https://learnopencv.com/)
* [**https://chat.openai.com/**](https://chat.openai.com/)
* [**Www.Wikipedia.org**](http://Www.Wikipedia.org)

Pg. 31