Report On

Motion controlled game system using computer vision

Submitted in partial fulfillment of the requirements of the Mini project in
Semester IV of Second Year Artificial Intelligence and Data Science (AI&DS)

By

Parth Puri (Roll No. 24)
Divyah Mandavia (Roll No. 11)
Devashree Pawar (Roll No. 22)

Mentor
Prof. Priyanka Bhoir

University of Mumbai

Vidyavardhini's College of Engineering & Technology

Department of Artificial Intelligence and Data Science (AI&DS)

(A.Y. 2021-22)

Vidyavardhini's College of Engineering & Technology

Department of Artificial Intelligence and Data Science (AI&DS)

CERTIFICATE

This is to certify that the Mini Project entitled "**Motion controlled game system using computer vision**" is a Bonafede work, **Parth Puri (Roll No. 24), Devashree Pawar (Roll No. 22) and Divyah Mandavia (Roll No. 11)** submitted to the University of Mumbai in partial fulfillment of the requirement for the award of the degree of "Bachelor of Engineering" in Semester IV of Second Year "**Artificial Intelligence and Data Science (AI&DS)**".

_____

Prof. Name Surname
Mentor

_____ _____

Dr Thaksen Parvat
Co-ordinator of Department

Dr. H.V. Vankudre
Principal

Vidyavardhini's College of Engineering & Technology

Department of Artificial Intelligence and Data Science (AI&DS)

Mini Project Approval

This Mini Project entitled "**Motion controlled game system using computer vision"** by **Parth Puri (Roll No. 24), Devashree Pawar (Roll No. 22) and Divyah Mandavia (Roll No. 11)** is approved for the degree of **Bachelor of Engineering** in in Semester IV of Second **Year Artificial Intelligence and Data Science (AI&DS)**

Examiners

1……
(Internal Examiner Name & Sign)

2……
(External Examiner name & Sign)

Date:

Place:

Contents

Abstract:

This research deals with presenting a design of a game controller interface based on visual processing. Visuals define the world, each visual has its own story, it contains a lot of crucial information that can be useful in many ways. This information can be obtained with the help of the technique known as Visual Processing. It is the core part of computer vision which plays a crucial role in many real-world examples like robotics, self-driving cars, and object detection. The mechanical and electric approach of playing a video game has been changed, i.e., a joystick, with camera inputs. This research helps in exploring a field of artificial intelligence, i.e., Computer Vision. Computer vision is a field of artificial intelligence (AI) that enables computers and systems to derive meaningful information from digital images, videos and other visual inputs — and take actions or make recommendations based on that information. Computer vision works much the same as human vision, except humans have a head start. Human sight has the advantage of lifetimes of context to train how to tell objects apart, how far away they are, whether they are moving and whether there is something wrong in an image. Motivation is one of the key factors when having to do exercises for hand mobility, because they have to be done consistently. The application is visually attractive and simple to use. We propose a human-computer interaction (HCI) system for gaming purposes via a camera. HCI researchers observe the ways humans interact with computers and design technologies that allow humans to interact with computers in novel ways. The whole system consists of three modules: hand tracking, gesture recognition and integration with the controls in our system, specifically, hand detection is based entirely on computer vision.

Acknowledgement:

It gives us immense pleasure to put forward the summary of our project titled as “Motion controlled game system using computer vision”. We would like to thank our respected teachers who immensely contributed in the completion of this project and guided us through our difficulties. We would like to show our gratitude towards our mentor and all the teachers involved in this project for giving numerous consultations without which the project wouldn’t have been completed. Many people, especially our team members itself and our classmates, gave us valuable suggestions which boosted us and contributed towards the betterment of the project. We thank everyone who played a vital role in helping us directly or indirectly in the development of this project.

List of Figures:

1.)Introduction:

1.1) Introduction

This project presents a design of a game controller interface based on visual processing. Visual processing allows us to transform and manipulate thousands of Visual at a time and extract useful insights from them. It has a wide range of applications in almost every field. Python is one of the widely used programming languages for this purpose. Its amazing libraries and tools help in achieving the task of visual processing very efficiently. Through this article, you will learn about classical algorithms, techniques, and tools to process the image and get the desired output. In vision-based interfaces for video games, gestures are used as commands for the games instead of pressing buttons on a keyboard or moving a mouse.

Computer vision-based interface to a game holds a promise of rich natural interaction and thus provides a realistic gaming experience. In this interface, the unintentional movement and continuous gestures must be supported to provide the user with a more natural interface. To explore the vision-based systems we have worked on the game of car racing. We have coded this project using Python. We have included keyinput, mediapipe and OpenCV libraries.

This project is considered to be under the field of gesture recognition which is a field which is gaining widespread popularity as it makes it very easy for the user to get some particular tasks done. MediaPipe offers ready-to-use yet customizable Python solutions as a prebuilt Python package. NumPy is a Python library used for working with arrays. It also has functions for working in the domain of linear algebra, Fourier transform, and matrices. NumPy was created in 2005 by Travis Oliphant. It is an open-source project and you can use it freely. NumPy stands for Numerical Python.

OpenCV is the huge open-source library for computer vision, machine learning, and image processing and now it plays a major role in real-time operation which is very important in today's systems. By using it, one can process images and videos to identify objects, faces, or even handwriting of a human. When integrated with various libraries, such as NumPy, python is capable of processing the OpenCV array structure for analysis. To Identify image pattern and its various features we use vector space and perform mathematical operations on these features. Gesture recognition has indeed created very widespread use in the gaming industry and is surely moving ahead.

This module provides various time-related functions. For related functionality, see also the DateTime and calendar modules.

1.2) Problem Statements and Objectives:

The current game system consists of manual input using optical devices such as keyboard and mouse, the implementation of a system that recognizes the motion gestures to perform various operations that will make the current system effortless.

1.3) Scope:

This project has wide-ranging applications over the gaming performance of a player. It directly establishes Human Computer Interaction (HCI). It takes the gaming experience of a player to a next level. The background of the implementation can be also used to develop an interesting gaming application for the children in which they can get immersed and have a fabulous experience. It can be also implemented in various games according to the controls which is one of the best things about this project. As well as gaming can be more smooth, entertaining, realistic, etc.

2.) Literature Survey:

2.1) Survey of existing system/SRS:

[1] In this paper, we propose a human-computer interaction (HCI) system for entertainment or education via a depth-sensing camera. The whole system is comprised of three modules: hand detection, hand tracking, and gesture recognition. In our system, specifically, hand detection is based entirely on computer vision and does not use any markers. We utilize the Kalman filter and depth data from the Kinect to predict the hand position making the tracking smooth and robust. And we extract the apparent gestural features to recognize gestures, which are adaptable and quick.
Advantage: The most significant advantage of this paper is that its gesture-based HCI system is its natural interaction.
Disadvantage: This system works a little bit slow.

[2] This paper proposes a motivational application for people who have suffered hand trauma and aims to help them to do exercises for regaining hand mobility.
Advantages: The primary purpose of the application is to incite the user to perform as many hand exercises as possible.
Disadvantages: The main drawback consists in the fact that the system needed a 6-camera module and hand-held sensors to detect movement.

[3] This paper has been taken up to demonstrate an emerging field of Human-Computer Interaction i.e., Gesture Recognition. Gesture recognition has found its way into many applications ranging all across from basic home automation to navigation and gaming. In this paper, I have explored the capabilities of gesture recognition by demonstrating the classic Snake game as a gesture-controlled snake.
Advantages: This paper gives light on conceivable outcomes of utilizing computer vision to take care of genuine issues.
Disadvantages: The disadvantage of this paper system requirement.

[4] This research presents a design of a game controller interface based on visual processing. The mechanical and electrical interfaces of a video game can be replaced with a camera input. Tests were performed for a car simulation game. The game player operates a mock wheel with colored markings on it to the
camera.
Advantages: The main advantage of this paper is that it helps fast pace gamers to gain more control over the game as compared to traditional methods.
Disadvantages: This system can get more complex by adding additional gestures.

2.2) Limitation Existing system:

The system has a limitation on hand gestures. Only static gestures are supported. In addition, there are restrictions on multi-hand tracking and recognition. And multi-user cooperation or multi-hand competition is becoming more and more of a trend. The system gets slow which causes FPS fluctuations and it happens a frequent time.

2.3) Mini Project Contribution:

A quick, lightweight method in terms of computation resources for detecting and recognizing a set of gestures, under different light conditions. A game controlled by the gestures mentioned above, which aims to motivate the user into doing hand exercises for hand mobility.

This will take the gaming world to the next level. It will enhance the control over the game compared to the traditional method controller.

3.) Proposed System:

3.1) Introduction:

The project is implemented completely using Python. Libraries like python 3.x, cv2, math, key input and mediapipe, NumPy, etc. have contributed much towards this project. This project is solely based on a gesture-controlled remote controller which helps in playing different games on the computer. Setting up the key binding using the python libraries and then integrating the key bindings with the game gets the controller ready. The main and the most important thing to complete the project is a camera for capturing and also tracking the movements of the user is a need of the project. Implementing the landmarks on the hand is the most important thing in this project. The camera works like an interaction bridge between the system and the user. Eventually, that makes the whole system work efficiently and perfectly.

3.2) Process Design:

Live video capture using a camera is the first step. Implementing landmark points on hand is the most important thing for tracking the movements of the user. The next thing is setting up key binding using the python library. Importing libraries like cv2, math, keyinput and mediapipe are the most important libraries in the project. Integrating the key bindings with the code. Integrating the code with OpenCV helps to recognize the movements of the user. Setting up a data set for the game. Setting up the code as an application. And one of the most important steps is Testing the game makes the controller the best of the best.

3.3) Details of Hardware and Software:

Recommended System Requirements

Software Requirements – Python 3.7 or above is required.

Hardware Requirements - Processor intel core i5 5^{th} gen and above. And a graphics card is required.

3.4) Experiment and Results for Validation and Verification:

Experiment: Code-

MotoinController.py

```python
import math
import keyinput
import cv2
import mediapipe as mp
import time

mp_drawing = mp.solutions.drawing_utils
mp_drawing_styles = mp.solutions.drawing_styles
mp_hands = mp.solutions.hands
font = cv2.FONT_HERSHEY_SIMPLEX

# 0 For webcam input:
cap = cv2.VideoCapture(1)

# Initializing current time and precious time for calculating the FPS
previousTime = 0
currentTime = 0


with mp_hands.Hands(
    model_complexity=0,
    min_detection_confidence=0.5,
    min_tracking_confidence=0.5) as hands:
  while cap.isOpened():
    success, image = cap.read(cv2.WINDOW_NORMAL)
    fp = cv2.flip(image, 1)  # --------> flipped_fp

    # Calculating the FPS
    currentTime = time.time()
    fps = 1 / (currentTime-previousTime)
    previousTime = currentTime

    # Displaying background colour for fps
    cv2.rectangle(fp, (0, 0), (120,40), (0,0,0), -1)

    # Displaying FPS on the image     -------> Frames per second
    cv2.putText(fp, str(int(fps))+" FPS", (0, 30), cv2.FONT_HERSHEY_COMPLEX, 1, (0,255,0), 2)

    if not success:
      print("Ignoring empty camera frame.")
      # If loading a video, use 'break' instead of 'continue'.
      continue

    # To improve performance, optionally mark the fp as not writeable to
    fp.flags.writeable = False
    fp = cv2.cvtColor(fp, cv2.COLOR_BGR2RGB)
    results = hands.process(fp)
    fpHeight, fpWidth, _ = fp.shape


    # Draw the hand annotations on the fp.
    fp.flags.writeable = True
    fp = cv2.cvtColor(fp, cv2.COLOR_RGB2BGR)
    co=[]
    if results.multi_hand_landmarks:
      for hand_landmarks in results.multi_hand_landmarks:
        mp_drawing.draw_landmarks(
            fp,
            hand_landmarks,
            mp_hands.HAND_CONNECTIONS,
```

```
            mp_drawing_styles.get_default_hand_landmarks_style(),
            mp_drawing_styles.get_default_hand_connections_style())
        for point in mp_hands.HandLandmark:
            if str(point) == "HandLandmark.WRIST":
                normalizedLandmark = hand_landmarks.landmark[point]
                pixelCoordinatesLandmark = mp_drawing._normalized_to_pixel_coordinates(normalizedLandmark.x,
                                                                                       normalizedLandmark.y,
                                                                                       fpWidth, fpHeight)

                try:
                    co.append(list(pixelCoordinatesLandmark))
                except:
                    continue

    if len(co) == 2:
        # print(co)
        xm, ym = (co[0][0] + co[1][0]) / 2, (co[0][1] + co[1][1]) / 2
        radius = 150
        try:
            m=(co[1][1]-co[0][1])/(co[1][0]-co[0][0])
        except:
            continue
        a = 1 + m ** 2
        b = -2 * xm - 2 * co[0][0] * (m ** 2) + 2 * m * co[0][1] - 2 * m * ym
        c = xm ** 2 + (m ** 2) * (co[0][0] ** 2) + co[0][1] ** 2 + ym ** 2 - 2 * co[0][1] * ym - 2 * co[0][1] * co[0][
            0] * m + 2 * m * ym * co[0][0] - 22500

        # centre horizontal line or diameter of the circle
        xa = (-b + (b ** 2 - 4 * a * c) ** 0.5) / (2 * a)
```

```
        xb = (-b - (b ** 2 - 4 * a * c) ** 0.5) / (2 * a)
        ya = m * (xa - co[0][0]) + co[0][1]
        yb = m * (xb - co[0][0]) + co[0][1]

        if m!=0:
            ap = 1 + ((-1/m) ** 2)
            bp = -2 * xm - 2 * xm * ((-1/m) ** 2) + 2 * (-1/m) * ym - 2 * (-1/m) * ym
            cp = xm ** 2 + ((-1/m) ** 2) * (xm ** 2) + ym ** 2 + ym ** 2 - 2 * ym * ym - 2 * ym * xm * (-1/m) + 2 * (-1/m) * ym * xm - 22500
            try:
                xap = (-bp + (bp ** 2 - 4 * ap * cp) ** 0.5) / (2 * ap)
                xbp = (-bp - (bp ** 2 - 4 * ap * cp) ** 0.5) / (2 * ap)
                yap = (-1 / m) * (xap - xm) + ym
                ybp = (-1 / m) * (xbp - xm) + ym

            except:
                continue

        cv2.circle(img=fp, center=(int(xm), int(ym)), radius=radius, color=(15,185,255), thickness=15)

        l = (int(math.sqrt((co[0][0] - co[1][0]) ** 2 + (co[0][1] - co[1][1]) ** 2)) - 150) // 2
        cv2.line(fp, (int(xa), int(ya)), (int(xb), int(yb)), (15,185,255), 20)


        if co[0][0] < co[1][0] and co[0][1] - co[1][1] > 65:
            # When the slope is negative, we turn left.
            print("Turning Left")
            keyinput.release_key('s')
            keyinput.release_key('a')
            keyinput.press_key('a')
```

```
            keyinput.press_key('a')
            cv2.putText(fp, "Turning Left", (130,30), font, 0.8, (0, 0, 255), 2, cv2.LINE_AA)
            cv2.line(fp, (int(xap), int(yap)), (int(xm), int(ym)), (255, 0, 0), 20)

        elif co[1][0] > co[0][0] and co[1][1] - co[0][1] > 65:
            print("Turning Right")
            keyinput.release_key('s')
            keyinput.release_key('d')
            keyinput.press_key('d')
            cv2.putText(fp, "Turn Right", (130,30), font, 0.8, (0, 0, 255), 2, cv2.LINE_AA)
            cv2.line(fp, (int(xbp), int(ybp)), (int(xm), int(ym)), (255, 0, 0), 20)

        else:
            print("keeping straight")
            keyinput.release_key('s')
            keyinput.release_key('a')
            keyinput.release_key('d')
            keyinput.press_key('w')
            cv2.putText(fp, "keep straight", (130,30), font, 0.8, (0, 255, 0), 2, cv2.LINE_AA)
            if ybp>yap:
                cv2.line(fp, (int(xbp), int(ybp)), (int(xm), int(ym)), (15,185,255), 20)
            else:
                cv2.line(fp, (int(xap), int(yap)), (int(xm), int(ym)), (15,185,255), 20)

    if len(co)==1:
        print("Reverse")
        keyinput.release_key('a')
        keyinput.release_key('d')
        keyinput.release_key('w')
        keyinput.press_key('s')
```

```
        cv2.putText(fp, "Reverse", (130,30), font, 1.0, (0, 0, 255), 2, cv2.LINE_AA)

    cv2.imshow('Motion controlled game system using computer vision', fp)


    if cv2.waitKey(5) & 0xFF == ord('q'):
        break
cap.release()
```

Result:

1. Holding a steering wheel-> Go straight

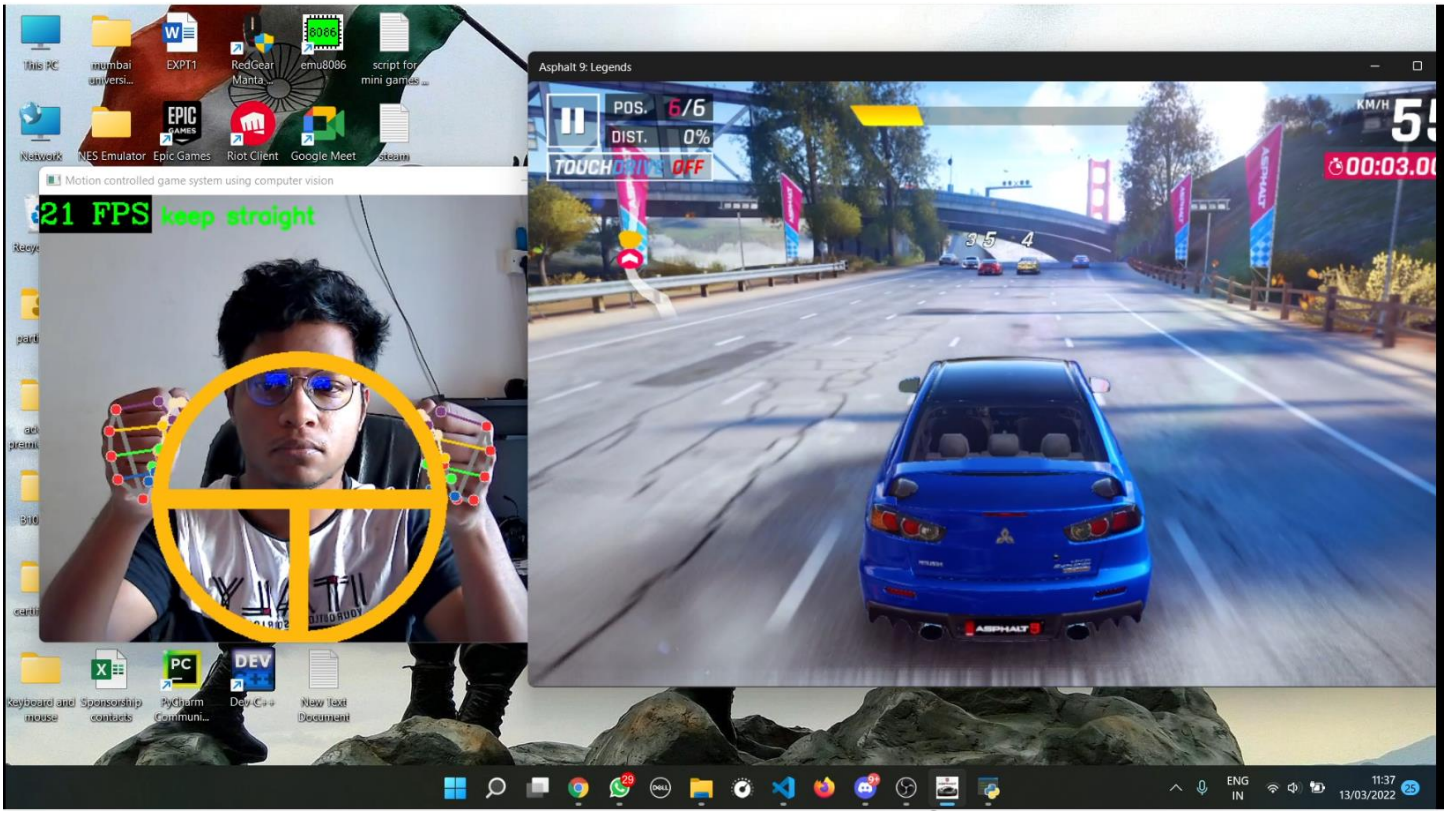

FIG 3.4.1

2. Fists turning right -> turn right

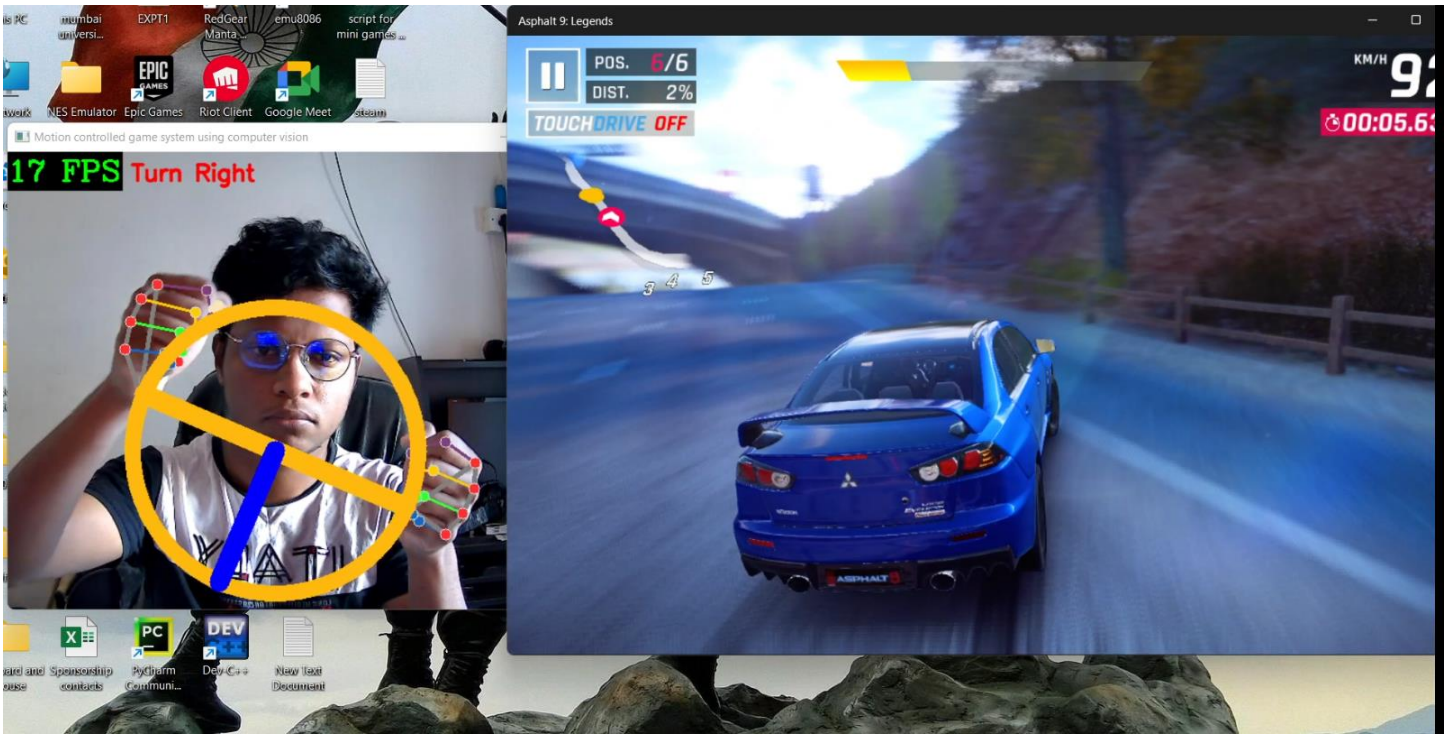

FIG 3.4.2

3. Fists turning left -> turn left

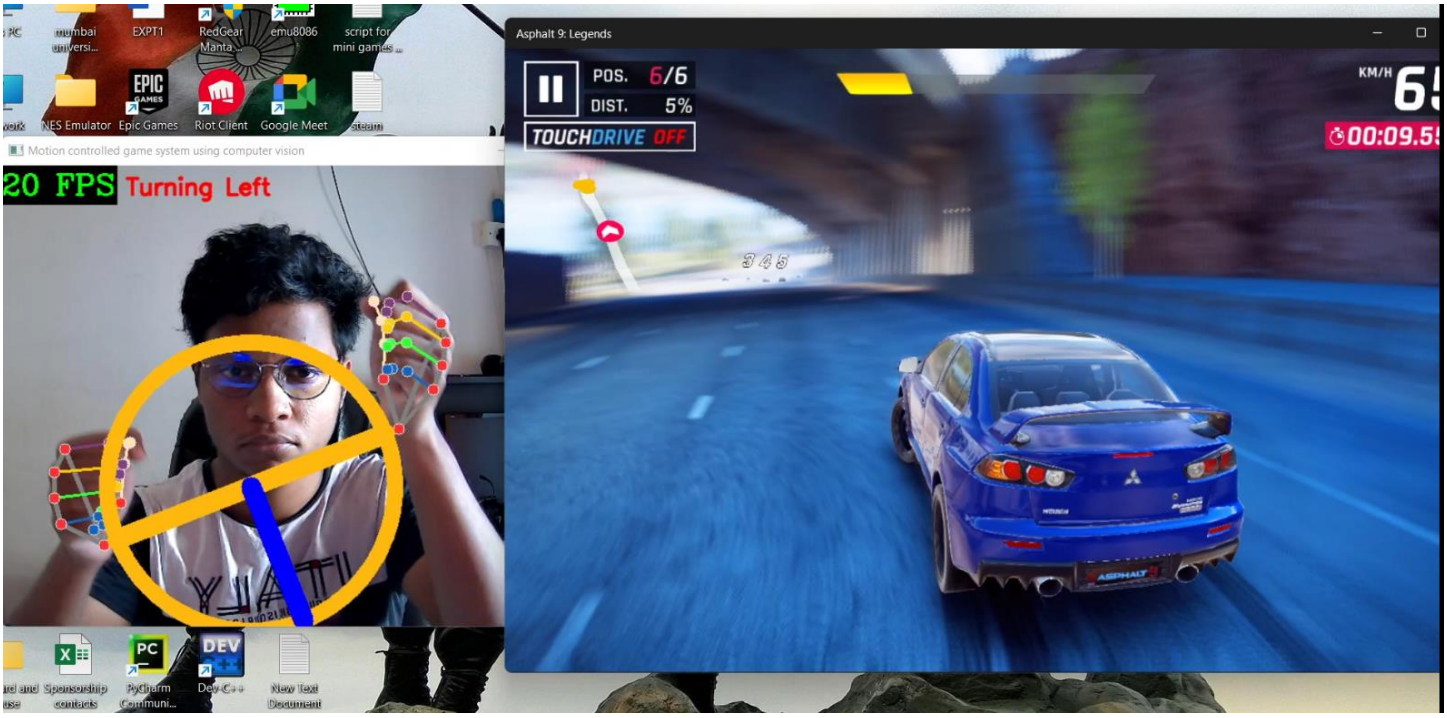

FIG 3.4.3

4. One hand -> reverse

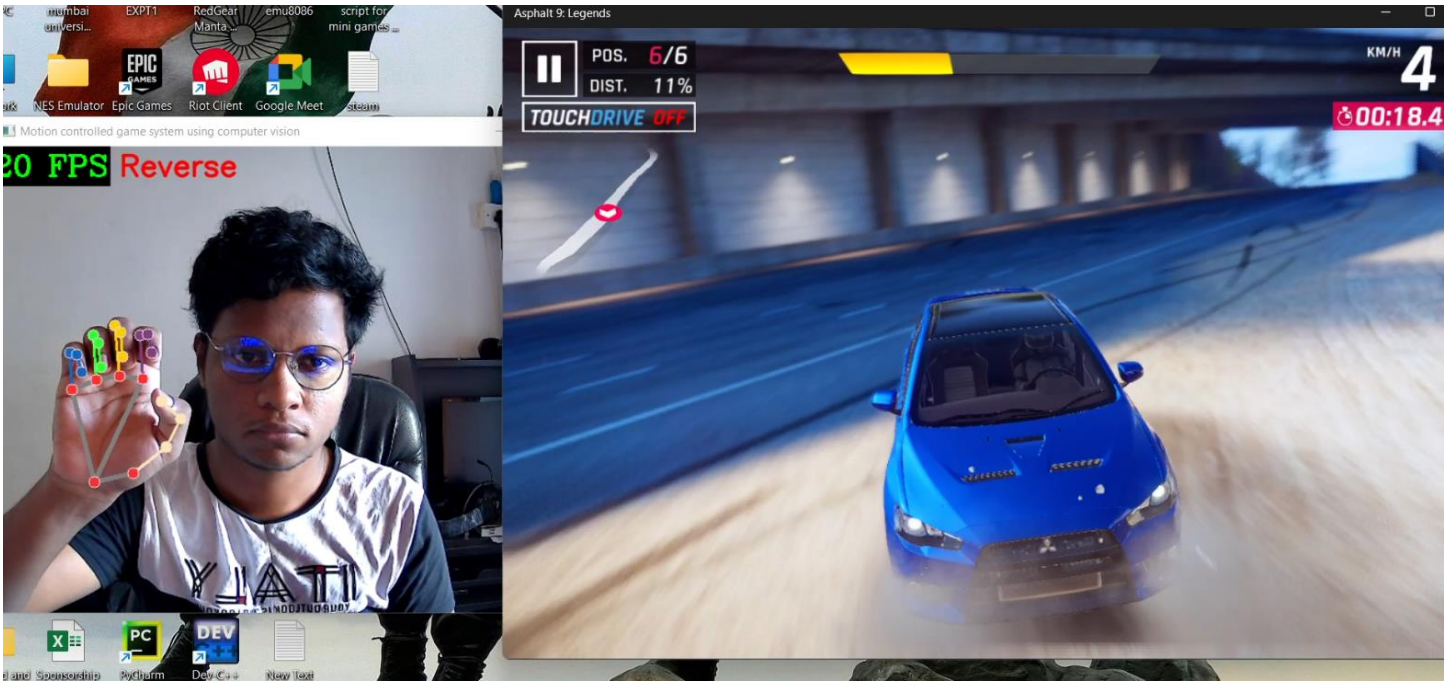

FIG 3.4.4

3.6) Analysis:

Gesture recognition has found its way into many applications ranging all across from basic home automation to navigation and gaming. the field of gesture recognition which is a field which is gaining widespread popularity as it makes it very easy for the user to get some particular tasks done. Gesture recognition has slowly but surely created a stronghold in the gaming industry and is now moving beyond it in every aspect. Together with computer vision, it is enabling a much better user experience and dynamic and user-friendly interfaces.

3.7) Conclusion:

Conclusion:

With the help of OpenCV it has become very easy to implement hand gestures which have made this task faster and simpler. This project helps us to build and learn various aspects of programming. This also makes gaming a life-changing experience.

3.8) References

REFERENCES

[1] Y. Wen, K. Zhang, Z. Li, and Y. Qiao, “A Discriminative Feature Learning Approach for Deep Face Recognition,” Proc. of European Conference on Computer Vision (ECCV), pp. 499-515, 2016.

[2] Y. Li, “Hand gesture recognition using Kinect,” Proc. of IEEE Control and Decision Conference (ICDC), pp. 196-199, 2017.

[3] J. Wu and J. Cheng, “Bayesian Co-Boosting for Multi-modal Gesture Recognition,” Journal of Machine Learning Research, vol. 15, no. 1, pp. 3013-3036, 2017.

[4] Q. Pu, S. Gupta, S. Gollakota, and S. Patel, “Whole-home gesture recognition using wireless signals,” Proc. of International Conference on Mobile Computing & NETWORKING ACM, pp.27-38, 2013.

[5] C. H. Morimoto and M. R. M. Mimica, “Eye gaze tracking techniques for interactive applications ☆,” Computer Vision & Image Understanding, vol. 98, no. 1, pp. 4-24, 2005.