Hand Tracking & Game Development

PyCK Project

Aayush Goel 200100004
Amal Thomas 200100020

CONTENTS

INTRODUCTION

This is our implementation of hand tracking to play games via hand gestures .
Libraries used:- Mediapipe , OpenCV , Pygame.

Mediapipe-Hands - MediaPipe Hands utilizes an ML pipeline consisting of multiple models working together: A palm detection model that operates on the full image and returns an oriented hand bounding box. A hand landmark model that operates on the cropped image region defined by the palm detector and returns high-fidelity 3D hand keypoints.

OpenCV - OpenCV is the huge open-source library for computer vision, machine learning, and image processing. It helps 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 and hand actions

Pygame - Pygame is a cross-platform set of modules designed for writing games. It includes computer graphics and sound libraries designed to be used with the python programming language.

PROBLEM STATEMENT

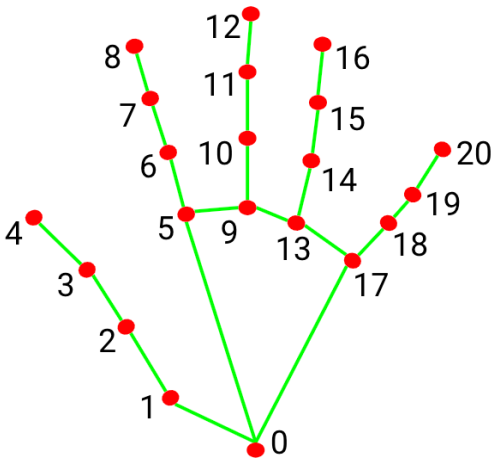
Hand Tracking :- By identifying Specific Landmarks on hand, We can easily track hands. Mediapipe hands utilizes an ML Pipeline Consisting of multiple models working together.

Game Development :- Developing a simple game using pygame that can be easily played using hand gestures.

MAIN ALGORITHM

Hand Tracking

Performed using Libraries OpenCV and Mediapipe. Frames captured from camera are processed using mediapipe. After performing multiple hand processing features multiple hand landmarks are marked .

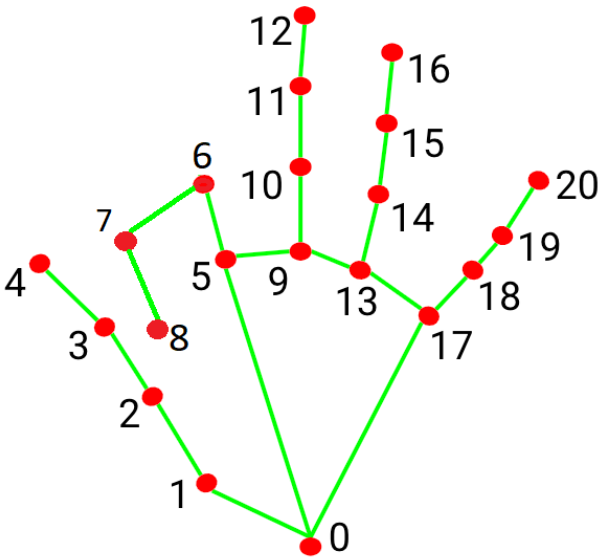


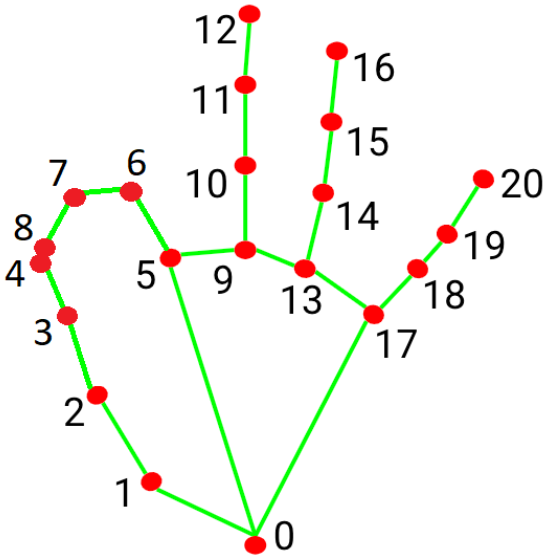
0. WRIST
1. THUMB_CMC
2. THUMB_MCP
3. THUMB_IP
4. THUMB_TIP
5. INDEX_FINGER_MCP
6. INDEX_FINGER_PIP
7. INDEX_FINGER_DIP
8. INDEX_FINGER_TIP
9. MIDDLE_FINGER_MCP
10. MIDDLE_FINGER_PIP
11. MIDDLE_FINGER_DIP
12. MIDDLE_FINGER_TIP
13. RING_FINGER_MCP
14. RING_FINGER_PIP
15. RING_FINGER_DIP
16. RING_FINGER_TIP
17. PINKY_MCP
18. PINKY_PIP
19. PINKY_DIP
20. PINKY_TIP

- Above shown are 21 landmarks found using mediapipe.
- A new list containing the following data is created.

[id, pos_x , pos_y , distance_from_landmarks[0]]

- To count the number of fingers shown, we calculate the distance between some specific landmarks. Distance(0,6)=>Distance between landmarks [0] and [6].
- To detect Index finger distance(0,6) and distance(0,8) are compared and if the distance(0,6) > distance(0,8), then the index finger is folded. Thus, by applying such parameter to all four finger one can find out number of fingers that are folded. One can also detect if Thumb and index finger are joined.





Game Development

Creating the Gaming Window

Creating a gaming window with a size of 800X600 pixels.

Creating the Player

Every time we start the game the player spawns in a random lane. We created the variable Current_Lane

to store the position of the player.

Starting the Game

Code is written such that game will start only if once a key command is received and we also ensured that the user can quit the gaming window by clicking the close(x) button on the window.

Multiple Plays on a Single Run

To allow the user to play as many times as he/she wishes, we included the dynamic portion of the code inside a larger while loop.

Creating the Enemy

We created lists for storing the enemy images, speed, x & y coordinates. We appended these values using a list. Using the random module to position enemies.

User Defined Functions

- show_score(x, y) **:** To display the score on the screen.
- game_over_text() **:** To display the “Game Over” text and instructions to play again.
- enter_to_play() **:** To check whether a key is pressed.
- player(x, y) **:** To display the player.
- enemy(x, y, i) **:** To display an enemy.
- isCollision(enemyY, playerY, i) **:** To check if there is any collision with the enemy.

Main Game Loop

The actual game happens inside the game loop which itself is contained inside the larger while loop. This loop is necessary since we have to keep updating the game

window for the changes to take effect.

Receiving Input Commands

We have created 2 alternative methods to give the input commands to the game :-

1. Through Keypad (Left arrow key or Right arrow key)
2. Through hand gestures

Increasing the Difficulty of Game

Once the enemy reaches the end of the lane it disappears , score is incremented by 1. As the score increases by the units speed of enemy cars also Increases.

Game Over!

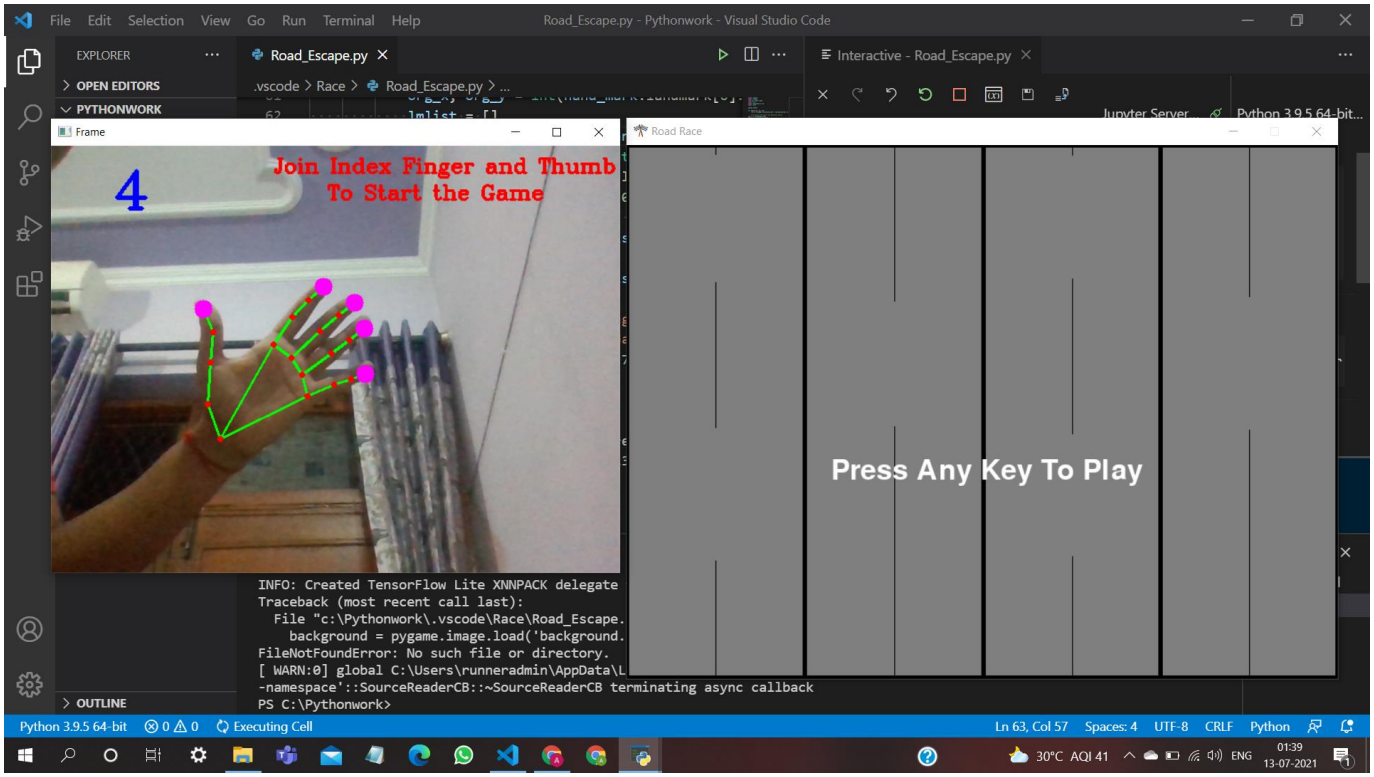
Once the game is over, player is given a chase to play the game again.

Connecting Both algorithms Written Above

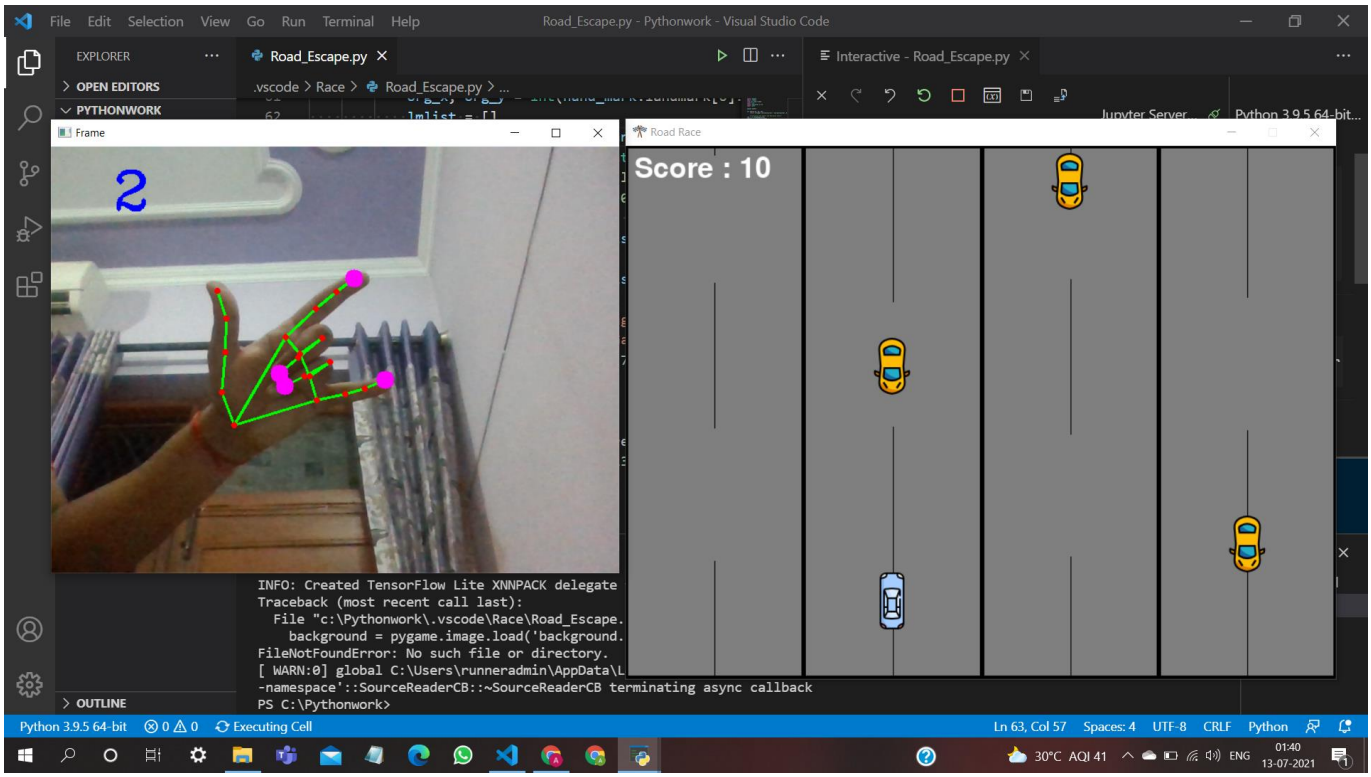
- This Part was the Most difficult to execute.
- Start the game by joining your index finger and thumb.
- Car will move to the lane according to the number of fingers shown. Please note that Thumb doesn't count.
- If the game is over, restart the game by joining your index finger and thumb.
- Running both windows together was difficult to achieve.

RESULTS

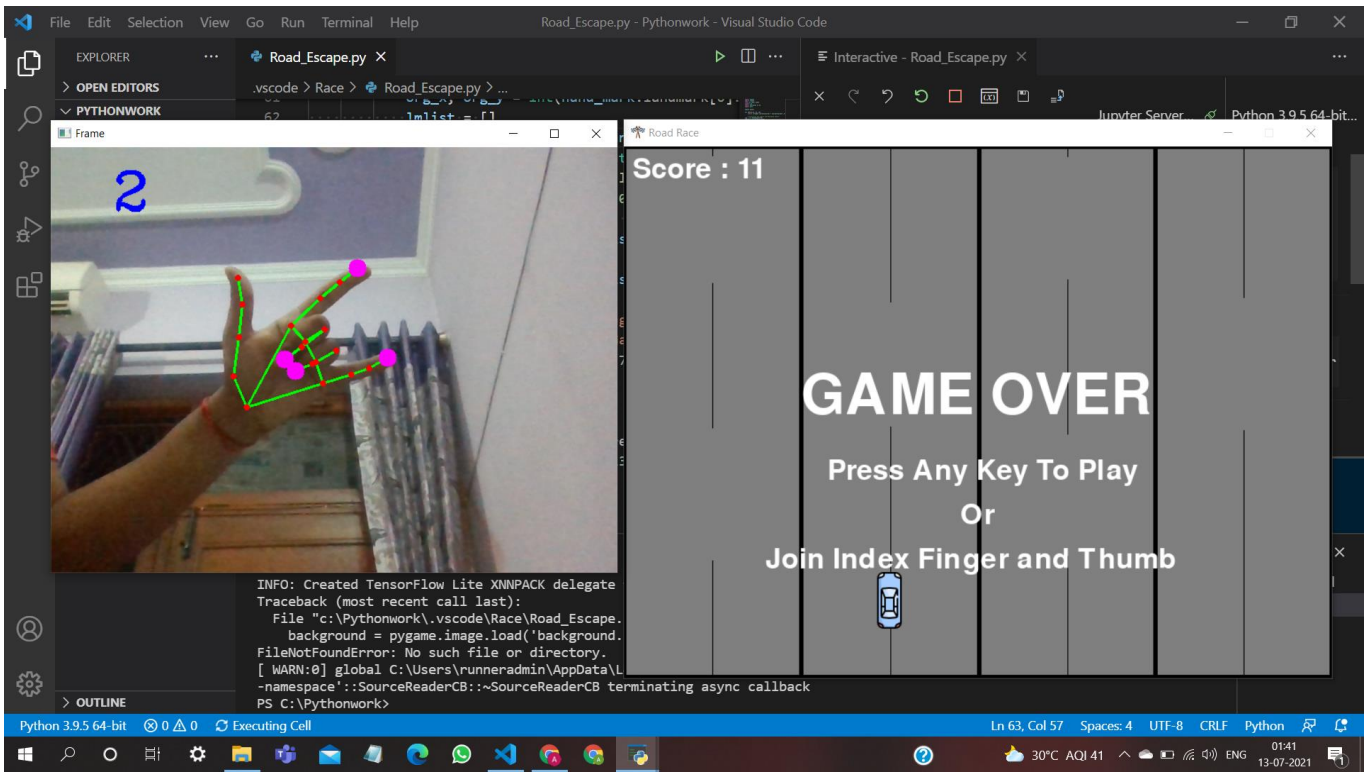
- Both camera and game window will start at the same time.



- After starting the game, control the car with your Fingers.



- After Game is Over, you can restart the game.



REFERENCES

- For documentation on Mediapipe
https://google.github.io/mediapipe/solutions/hands.html
- For Tutorial on Pygame
https://www.youtube.com/watch?v=FfWpgLFMI7w

APPLICATIONS OF HAND TRACKING

- We can even use the hand tracking feature to read hand signs.Example - MNIST sign Language or ASL . By applying specific parameters we can convert hand signs to written texts.
- Hand Tracking has many applications. By combining it with various other modules we can use it to control computer via hand gestures.
- It has many uses in robotics. A robotic arm can be designed to mirror a normal human hand by providing displacement of all 21 landmarks to the robotic hand.