GameController using HandGestures & OpenCV

Project Report Submitted in partial fulfilment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

Submitted by

Roll No	Name	Dept.	Year
2300080386	AHMADULLAH LASKAR	AI&DS	Y-23
2300080146	GNV MAHESH	AI&DS	Y-23
2300080100	S PAVAN SAI RAHUL	AI&DS	Y-23
2300080086	P SAILESH VIVEKANANDA	AI&DS	Y-23

Course Code:23SDAD10A

Course title: Computer Vision Using Open CV

Under the guidance of

Dr.Anuradha Thati

Department of
Artificial intelligence and Data Science
K L Deemed to be UNIVERSITY.
Green Fields, Vaddeswaram, Andhra Pradesh 522501
2025-26

KL DEEMED TO BE UNIVERSITY

Green Fields, Vaddeswaram, Andhra Pradesh 522501

CERTIFICATE

This is to certify that the **COMPUTER VISION USING OPENCV** course project report entitled **"GameController using HandGesture"** submitted by Ahmadullah Laskar (2300080386), GNV MAHESH(2300080146) , P Sailesh Vivekananda(2300080086), S Pavan S Rahul(2300080100) in partial fulfillment of the requirements for the award of the Degree Bachelor of Technology in "Artificial Intelligence & Data Science" is a Bonafide record of the work carried out under our guidance and supervision at K L Deemed to be University during the academic year 2025-2026.

Signature of Guide **Head of The Department**

Dr. Anuradha Thati

ACKNOWLEDGEMENT

We are greatly indebted to our KL Deemed to be University that has provided a healthy environment to drive us to achieve our ambitions and goals. We would like to express our sincere thanks to our project guide Anuradha Thati for the guidance, support and assistance they have provided in completing this project.

We are thankful to our Head of the Department **Dr.V.S.V.Prabhakar, Professor, Dept. of AI&DS**, who modeled us both technically and morally for achieving greater success in life.

We are very glad to have the support given by our principal, who inspired us with his words filled with dedication and discipline towards work.

Finally, we owe a lot to the teaching and non-teaching staff of the **Dept. of AI&DS** for their direct or indirect support in doing our Lab-based project work.

Ahmadullah Laskar (2300080386)

G N V Mahesh (2300080146)

P Sailesh Vivekananda (2300080086)

S Pavan Sai Rahul (2300080100)

INDEX

ABSTRACT

The project **“Game Controller using Hand Gestures and OpenCV”** is designed to create an innovative and interactive system that allows users to control games through hand gestures instead of traditional input devices like keyboards or joysticks. This approach aims to enhance the gaming experience by making it more natural, immersive, and accessible.

The system uses a webcam or camera module to capture real-time hand movements, which are processed using **OpenCV** and image processing techniques. The preprocessing phase includes steps such as background subtraction, skin color segmentation, contour detection, and noise removal to accurately isolate the hand region. Once detected, specific hand gestures—such as palm, fist, or directional movements—are recognized using contour analysis and convex hull methods. These gestures are then mapped to corresponding game controls or actions.

By integrating **computer vision and gesture recognition**, the project eliminates the need for physical controllers, enabling touch-free interaction. The system can be extended to various gaming genres and can also assist differently-abled users by providing an alternative mode of input. This project demonstrates how **OpenCV and Python** can be utilized to develop a real-time, intelligent, and user-friendly control system, contributing to advancements in **human-computer interaction (HCI)** and **interactive gaming technologies**.

PROJECT DESCRIPTION

The project **“Game Controller using Hand Gestures and OpenCV”** aims to develop an innovative, vision-based control system that enables users to interact with computer games and applications through hand gestures instead of traditional input devices like keyboards, mice, or joysticks. The main objective is to create an intuitive, touch-free, and immersive user experience that enhances human-computer interaction. The system uses a **webcam** to capture real-time video streams, which are processed using **OpenCV** and **MediaPipe** for accurate hand and finger tracking.

The captured frames undergo preprocessing operations such as color space conversion, noise reduction, and landmark detection to identify key hand positions and orientations. Based on these detected landmarks, specific gestures are mapped to control commands.
For example, in the **Subway Surfers/Temple Run controller**, the system divides the screen into zones—moving the hand left, right, up, or down triggers corresponding game movements. In the **car racing controller**, the rotation of the hand determines the steering direction, while open or closed hand gestures control acceleration and braking. A third module extends this concept to **mouse cursor control**, enabling cursor movement and click actions through hand gestures.

By integrating **computer vision and gesture recognition**, the project demonstrates a practical application of **Human-Computer Interaction (HCI)**. It eliminates the dependency on physical controllers, offering an accessible and engaging interface, especially beneficial for differently-abled users. The system’s real-time responsiveness and accuracy make it suitable for gaming, robotics, and virtual reality applications, showcasing how **OpenCV and Mediapipe** can bridge the gap between physical gestures and digital control.

SOFTWARE AND HARDWARE REQUIREMENT

Software requirements:

- Operating System: Windows 10 / 11 or Ubuntu (64-bit)
- Programming Language: Python 3.8 or above
- Development Environment: Jupyter Notebook / Google Colab / VS Code/ Thonny(*)

Libraries and Packages

- **OpenCV:** Used for real-time image acquisition, preprocessing, and hand detection through computer vision techniques.
- **MediaPipe:** Provides robust hand landmark detection and tracking, enabling accurate gesture recognition.
- **NumPy:** Handles numerical operations, coordinate calculations, and angle estimation for gesture control.
- **PyAutoGUI:** Simulates keyboard and mouse actions, allowing gesture-based interaction with games and desktop environments.
- **Time:** Manages cooldown intervals and real-time responsiveness between gesture detections.
- **Math / NumPy (trigonometric functions):** Used to calculate steering angles and gesture directions.

Additional Tools

- **Anaconda / Jupyter Notebook:** For managing packages and development in a controlled environment.
- **Google Colab:** Used for testing and running the system with webcam support and GPU acceleration when needed.

Hardware description:

- Processor: Intel Core i3 or above (i5/i7 recommended)
- RAM: Minimum 4 GB (8 GB or more recommended for faster model training)
- Storage: Minimum 1 GB free disk space
- GPU (Optional): NVIDIA GPU with CUDA support for accelerated computation (useful for large datasets)
- Camera: High-resolution digital camera or smartphone camera for capturing leaf images

Technologies Used

1. **Python:**

   Serves as the core programming language for the project. Its simplicity and versatility make it ideal for developing real-time computer vision applications. Python supports key libraries such as OpenCV, MediaPipe, and PyAutoGUI, which are essential for gesture detection and system control.

2. **OpenCV (Open Source Computer Vision Library):**

   Used for capturing and processing live video frames from the webcam. It performs essential image processing tasks such as color space conversion, contour detection, and region segmentation. OpenCV also enables real-time visualization of gesture tracking and control zones.

3. **MediaPipe:**

   A framework by Google used for detecting and tracking hand landmarks with high precision. It identifies finger joints and hand orientations, which are crucial for recognizing gestures like movement, rotation, and open or closed hand states.

4. **NumPy (Numerical Python):**

   Provides efficient handling of numerical operations, coordinate calculations, and vector-based computations. It is used to determine hand angles, distances, and steering directions based on landmark coordinates.

5. **PyAutoGUI:**

   Enables the simulation of keyboard and mouse inputs, allowing hand gestures to control games or computer applications. It bridges the gap between visual gesture detection and actual system actions like pressing arrow keys or moving the cursor.

6. **Time Library:**

   Used to manage cooldown intervals, response delays, and timing between gesture-based actions to ensure smooth real-time control.

7. **Computer Vision & Gesture Recognition Techniques:**

   Includes image preprocessing, contour detection, angle estimation, and motion tracking. These techniques collectively help interpret user gestures accurately and map them to game commands such as *left, right, jump, accelerate,* or *brake*.

8. **Development Environment (Jupyter Notebook / Google Colab):**

   Used for testing, debugging, and running the system in an interactive environment. Google Colab provides easy setup and access to webcam inputs for real-time demonstrations.

Description of Dataset

The project **“Game Controller using Hand Gestures and OpenCV”** uses **real-time video frames captured from a webcam** instead of a traditional labeled dataset. Each frame acts as an input sample representing different hand gestures or positions, which are processed and classified in real time using **OpenCV** and **MediaPipe**. These gestures correspond to specific control actions such as *left, right, jump, accelerate,* or *brake* in various games.

- **Format:** Real-time frames captured from webcam (RGB images)
- **Resolution:** Varies depending on camera input (typically 640×480 pixels)
- **Color Space:** RGB
- **Annotations:** Gesture classes (e.g., Left, Right, Up, Down, Accelerate, Brake) are defined through code logic rather than labeled images
- **Data Type:** Continuous stream of video frames converted into NumPy arrays
- **Preprocessing Applied:**
    - Conversion from BGR to RGB color space
    - Noise removal and background filtering
    - Hand landmark detection using MediaPipe
    - Region segmentation and coordinate extraction

Data Source

The real-time gesture data is captured **directly from the system’s webcam** during execution.

However, for offline training or testing, **Kaggle’s Hand Gesture Recognition Dataset** (by *Monk AI*) or **Sign Language MNIST Dataset** can also be used as a reference dataset to simulate predefined gestures.

Purpose and Use

This dataset (or real-time data stream) is used to:

- Detect and recognize human hand gestures for game control
- Map visual inputs to simulated keyboard/mouse actions using PyAutoGUI
- Demonstrate real-time gesture-based interaction for Human-Computer Interaction (HCI) and gaming applications

Algorithm Implemented

1. Overview

The project implements a **computer vision–based gesture recognition algorithm** that allows users to control games using **hand movements** instead of traditional input devices.
It combines **OpenCV** for video processing, **MediaPipe** for real-time hand landmark detection, and **PyAutoGUI** for simulating keyboard inputs.
The algorithm works in real-time using a webcam feed and maps specific gestures to corresponding game actions such as moving left, right, jumping, or firing.

2. Step-by-Step Algorithm

Step 1: Image Acquisition

- The webcam continuously captures real-time video frames.
- Each frame serves as input to the gesture recognition system.

Step 2: Hand Detection (Using MediaPipe)

- The MediaPipe library detects the hand in each frame and extracts **21 landmark points** from the hand (fingertips, joints, wrist, etc.).
- These landmarks are used to track the hand's position and movement direction.

Step 3: Feature Extraction

- From the landmark coordinates, features such as **finger count**, **distance between landmarks**, and **hand position (x, y)** are calculated.
- These extracted features represent specific gestures (like open palm, closed fist, or directional movement).

Step 4: Gesture Recognition Logic

- Based on the position and angle of the landmarks:
    - If the hand moves **left**, the system triggers a **Left Arrow key** event.
    - If the hand moves **right**, it triggers a **Right Arrow key** event.
    - Raising a hand may simulate a **Jump (Spacebar)**.
    - A specific gesture (e.g., closed fist) can be mapped to **Fire/Attack (Ctrl)**.
- These mappings are implemented using **conditional statements** that interpret gesture patterns in real time.

Step 5: Action Execution (Using PyAutoGUI)

- The recognized gesture is translated into the corresponding **keyboard or mouse event** through PyAutoGUI.
- These simulated inputs control the game in the same way as physical key presses.

3. Algorithm Workflow Summary

1. **Input:** Real-time video feed from webcam.
2. **Detect:** Hand landmarks using MediaPipe.
3. **Extract:** Landmark coordinates and gesture features.
4. **Recognize:** Match pattern to predefined gesture rules.
5. **Control:** Trigger corresponding keyboard/mouse action using PyAutoGUI.

4. Advantages of the Implemented Algorithm

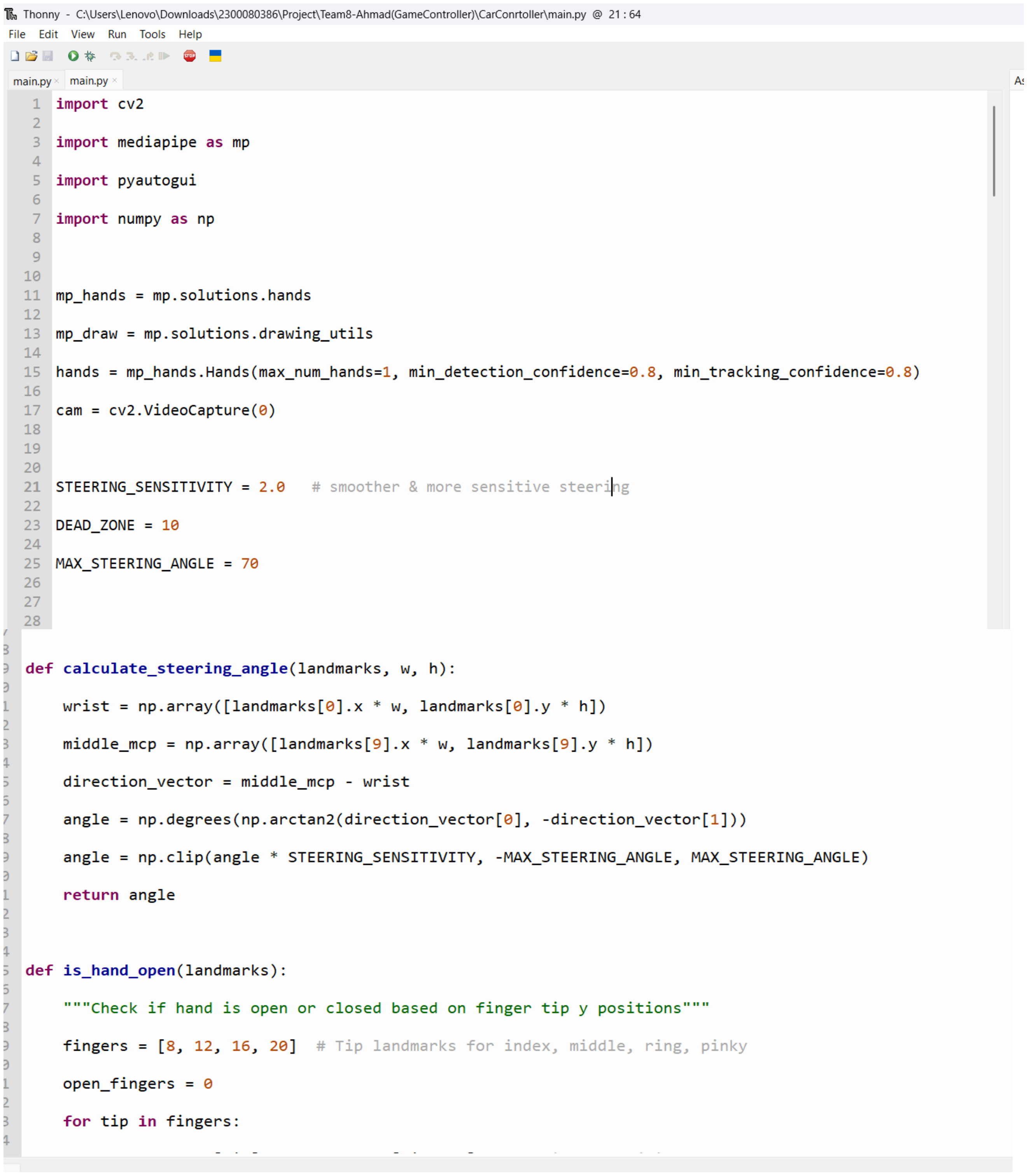
- Enables **touch-free, interactive game control**.
- Works in **real-time** with low latency.
- **Cost-effective** — no external sensors or controllers needed.
- Can be easily **extended to other applications** like robotics or AR/VR control.

5. Algorithm Used

A **rule-based gesture recognition algorithm** combining

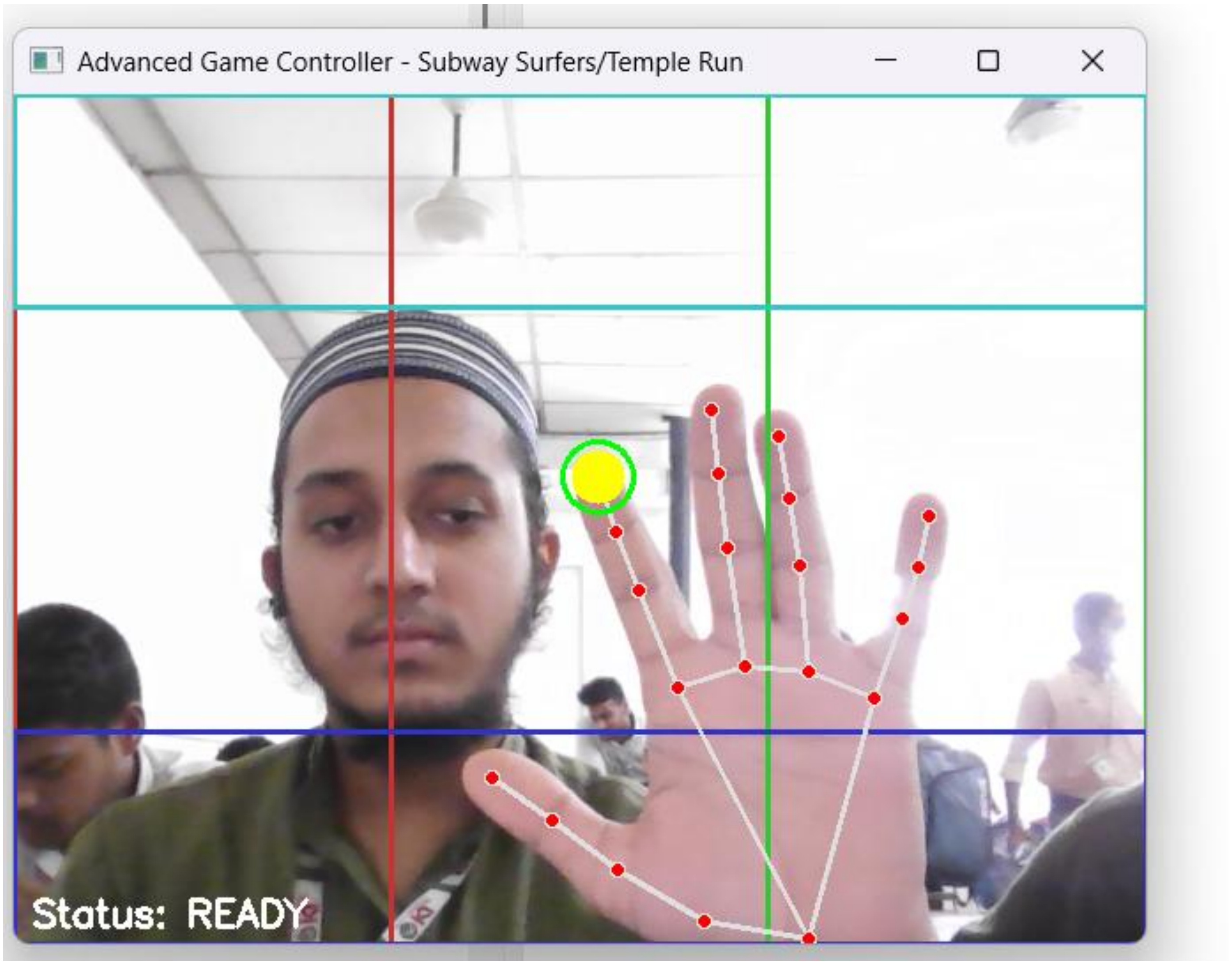
- **OpenCV** for image preprocessing,
- **MediaPipe** for hand landmark detection, and
- **PyAutoGUI** for input simulation,
  designed to control game functions using natural hand movements.

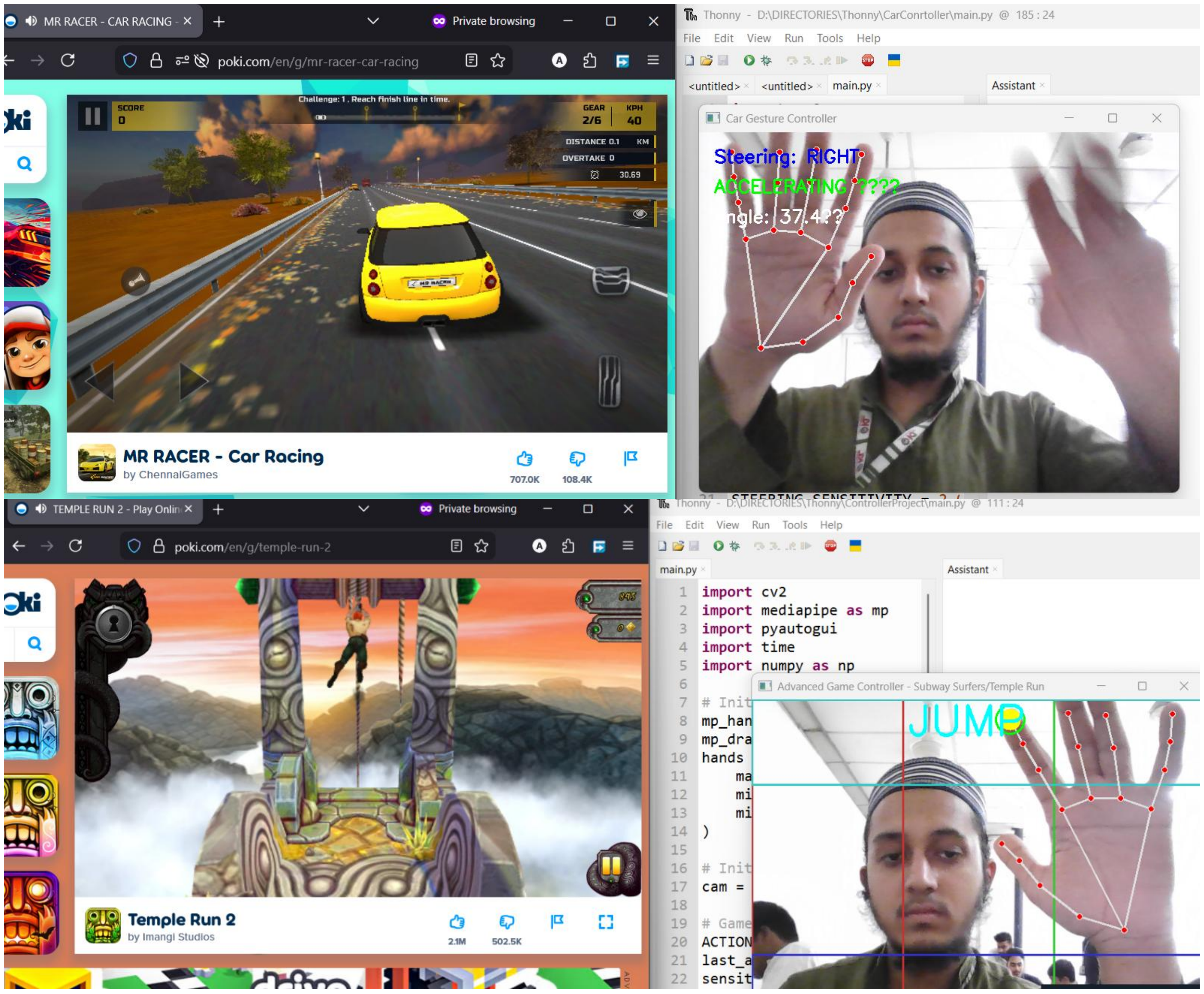
SCREENSHOTS



Thonny - C:\Users\Lenovo\Downloads\2300080386\Project\Team8-Ahmad(GameController)\CarConrtoller\main.py @ 21 : 64

```
import cv2

import mediapipe as mp

import pyautogui

import numpy as np


mp_hands = mp.solutions.hands

mp_draw = mp.solutions.drawing_utils

hands = mp_hands.Hands(max_num_hands=1, min_detection_confidence=0.8, min_tracking_confidence=0.8)

cam = cv2.VideoCapture(0)


STEERING_SENSITIVITY = 2.0   # smoother & more sensitive steering

DEAD_ZONE = 10

MAX_STEERING_ANGLE = 70


def calculate_steering_angle(landmarks, w, h):

    wrist = np.array([landmarks[0].x * w, landmarks[0].y * h])

    middle_mcp = np.array([landmarks[9].x * w, landmarks[9].y * h])

    direction_vector = middle_mcp - wrist

    angle = np.degrees(np.arctan2(direction_vector[0], -direction_vector[1]))

    angle = np.clip(angle * STEERING_SENSITIVITY, -MAX_STEERING_ANGLE, MAX_STEERING_ANGLE)

    return angle


def is_hand_open(landmarks):

    """Check if hand is open or closed based on finger tip y positions"""

    fingers = [8, 12, 16, 20]  # Tip landmarks for index, middle, ring, pinky

    open_fingers = 0

    for tip in fingers:
```


Advanced Game Controller - Subway Surfers/Temple Run
Status: READY


MR RACER - CAR RACING
poki.com/en/g/mr-racer-car-racing
MR RACER - Car Racing
by ChennaiGames
Thonny - D:\DIRECTORIES\Thonny\CarConrtoller\main.py @ 185 : 24
Car Gesture Controller
Steering: RIGHT
ACCELERATING ????
ngle: 37.4??
TEMPLE RUN 2 - Play Onlin
poki.com/en/g/temple-run-2
Temple Run 2
by Imangi Studios
Thonny - D:\DIRECTORIES\Thonny\ControllerProject\main.py @ 111 : 24
Advanced Game Controller - Subway Surfers/Temple Run
JUMP

```
import cv2
import mediapipe as mp
import pyautogui
import time
import numpy as np

# Init
mp_han
mp_dra
hands
    ma
    mi
    mi
)

# Init
cam =

# Game
ACTION
last_a
sensit
```

CONCLUSION

The project **“Game Controller using Hand Gestures and OpenCV”** successfully demonstrates how **computer vision** and **machine learning-based hand tracking** can replace traditional input devices like keyboards or game controllers.
By leveraging **OpenCV** for image capture and processing, **MediaPipe** for real-time hand landmark detection, and **PyAutoGUI** for triggering key events, the system enables users to **control games using simple hand gestures** such as moving left, right, up, and down.
This approach provides an **intuitive, touch-free, and interactive gaming experience**, showcasing the potential of **AI-driven human-computer interaction**. The system works by detecting the position and movement of the player’s hand through a webcam, mapping these gestures to specific keyboard actions that control game behavior.
The project achieved smooth gesture recognition with minimal delay, demonstrating high responsiveness and accuracy under proper lighting and background conditions. It eliminates the need for physical controllers, making it accessible, cost-effective, and fun to use.
In summary, this project highlights how **computer vision and real-time gesture tracking** can transform conventional gaming into an immersive and interactive experience. With further improvements—such as adding more gestures, improving noise filtering, and integrating with deep learning models for gesture recognition—this system can be expanded into **virtual reality, robotics control, and assistive technology** applications.

FUTURE SCOPE

1. Integration with Deep Learning Models (CNN):
   Future versions of the system can incorporate Convolutional Neural Networks (CNNs) or Recurrent Neural Networks (RNNs) for gesture classification, improving accuracy across varying lighting and background conditions.
2. Multi-Hand and Multi-Gesture Support:
   Extend the system to detect and process two hands simultaneously and recognize complex gestures for advanced gameplay actions like combos or dual-hand controls.
3. 3D Hand Pose Estimation:
   Integrating MediaPipe's 3D hand tracking or depth cameras (e.g., Intel RealSense or Kinect) can enable more immersive control, including distance-based actions such as zooming or speed control.
4. Custom Gesture Training:
   Users could record and train their own custom gestures for personalized game controls, making the system adaptable to different users and gaming genres.
5. Integration with AR/VR Systems:
   Incorporating the gesture control module into Augmented Reality (AR) and Virtual Reality (VR) environments could allow full-body interactive gaming without any physical controller.
6. Cross-Platform Compatibility:
   The system can be expanded to support mobile devices, Unity-based games, and web browsers, enhancing its usability across gaming platforms.
7. Performance Optimization:
   Optimizing the algorithm for GPU acceleration and using lightweight ML models could further improve frame rate and reduce latency during gameplay.

REFERENCES

- OpenCV Documentation — *Open Source Computer Vision Library*
  https://docs.opencv.org/
- MediaPipe by Google — *Hand Tracking Solution*
  https://developers.google.com/mediapipe
- PyAutoGUI Documentation — *Keyboard and Mouse Control Automation*
  https://pyautogui.readthedocs.io/
- NumPy Documentation — *Numerical Python for Mathematical Computation*
  https://numpy.org/doc/
- Adrian Rosebrock, *"Real-Time Hand Gesture Recognition with OpenCV and MediaPipe"*, PyImageSearch Blog, 2023.
- Google Research, *"MediaPipe Hands: On-Device Real-Time Hand Tracking"*, Proceedings of CVPR Workshops, 2020.