“Real-Time AI Virtual Mouse System Using Computer Vision.”

Team Members:	
1. Rajarshi Saha	20BCT0163
2. Girish Joshi	20BCT0132
3. Mehak Chaudhary	20BCT0120
4. Omkar Deshpande	20BEI0058

1. INTRODUCTION

The goal of the Hand Tracking System using OpenCV2 project is to create a system that can track hands in real-time and recognise gestures. This project seeks to recognise different hand gestures for interactive apps by precisely detecting and tracking a user's hand movements in real time.
Due to their promise in a variety of domains, including human-computer interaction, virtual reality, robotics, and sign language identification, hand tracking systems have attracted a lot of attention recently. These technologies make it possible for people to operate gadgets or software by using hand gestures and movements, enabling natural and intuitive interaction between humans and machines.
The open-source computer vision library OpenCV2 offers a full range of tools and techniques for image processing, object detection, and object tracking. It is the perfect option for creating a hand tracking system since it provides powerful and effective tools for hand detection, segmentation, and feature extraction.
The primary goal of this project is to use OpenCV2 to develop a real-time hand tracking system that can precisely identify and track the user's hand even in challenging surroundings with a variety of occlusions and lighting conditions. The system also intends to recognise a selection of predetermined hand gestures, enabling the user to issue commands or carry out specified actions.

The following are the main elements of the hand tracking system:
Hand detection entails locating the hand inside the frames of the collected footage and identifying it. For hand detection, a variety of methods can be used, such as machine learning-based approaches or skin colour modelling.
Hand segmentation: After the hand is identified, the system must separate the hand region from the surrounding area. For the hand movements to be accurately tracked and the necessary features to be extracted, this stage is essential.
Hand Tracking: Segmentation is followed by hand tracking, which monitors the hand's location and motion over a series of frames. For effective hand tracking, one can make use of tracking techniques like Kalman filters, particle filters, or template matching.

Gesture Recognition: To categorise the hand motions into specified gestures, the system uses gesture recognition algorithms. For precise gesture detection, machine learning methods like Hidden Markov Models (HMMs) or Convolutional Neural Networks (CNNs) can be used.

The hand tracking system seeks to deliver a smooth and engaging user experience by merging these elements. It can be applied to touchless user interfaces, augmented reality interactions, sign language interpretation, and virtual reality games.
Developing a real-time hand tracking and gesture recognition system is the main goal of the Hand Tracking System using OpenCV2 project. The system intends to precisely detect and track hand movements by utilising OpenCV2, allowing users to interact with gadgets and apps in a natural and intuitive way.

2. LITERATURE SURVEY

S.No.	TITLE	AUTHOR	FINDINGS	CHALLENGES
1.	"Real-time hand tracking and gesture recognition using depth data"	Moesen et al. (2018):	This paper presents a real-time hand tracking and gesture recognition system using depth data acquired from a depth sensor. The system is based on OpenCV2 and utilizes a combination of depth segmentation and hand pose estimation techniques.	• System Limitations and Future Directions • real-time hand tracking and gesture recognition using depth data.
2.	"Robust Hand Detection and Tracking	Ye et al. (2014)	This paper presents a robust hand detection and tracking system in	

	in Real Time"		real time. OpenCV2 is employed for hand detection and tracking using color and depth cues. The authors propose a method based on skin color modeling and depth information for accurate hand tracking.	
3.	Static and Dynamic Hand Gesture Recognition in Depth Data Using Dynamic Time Warping	Guillaume Plouffe et al. ()	Deep Convolutional Neural Networks" likely presents a new method for image classification using deep convolutional neural networks (CNNs). The authors introduce the challenges of image classification and emphasize the importance of accurate and efficient algorithms. They propose their unique approach, which may involve modifications or enhancements to the traditional CNN architecture. The authors conduct experiments using benchmark datasets, comparing	• Limited training data • Gesture ambiguity • Noise and sensor limitations • Occlusion

			their method with existing approaches	
4.	Improving Hand Gesture Recognition Using 3D Combined Features	Mahmoud Elmezain et al.	focuses on enhancing the accuracy and performance of hand gesture recognition. The authors propose a novel approach that leverages 3D combined features, which integrate different aspects of depth data to capture more comprehensive information about hand movements. By combining multiple features, such as shape, motion, and spatial relationships, the proposed method aims to address the challenges of variability, occlusion, noise, and gesture ambiguity in hand gesture recognition.	• Evaluation metrics • Dimensionality reduction • Gesture variability • Real-time processing
5.	Smart Wearable Hand Device for Sign Language Interpretatio n System	B. G. Lee et al.	presents a system that utilizes a smart wearable hand device for interpreting sign language. The authors propose a	• User adaptation and personalizatio n

	with Sensors Fusion		novel approach that combines sensor fusion techniques to accurately capture and interpret hand gestures. By integrating data from multiple sensors, such as accelerometers, gyroscopes, or flex sensors, the system aims to improve the recognition and understanding of sign language gestures. The paper may describe the design and implementation of the wearable device, along with the algorithms and methods used for sensor fusion.	<ul><li>Sensor accuracy and calibration</li><li>Real-time processing and latency</li><li>**Sensor fusion and synchronization:** Properly combining and fusing data from different sensors while maintaining temporal alignment is critical for accurate interpretation of sign language.</li><li>**Gesture recognition and classification:** Recognizing and classifying different sign language gestures accurately can be challenging due to the high variability and complexity of</li></ul>

				hand movements.
6.	Virtual Mouse Implementation using Open CV	PG Scholar,Dept of CS et al.	The suggested system makes use of real-time video input from a camera and use image processing methods to recognise hand gestures and movements. The user may manipulate the pointer on a computer screen without a real mouse by tracking the position and gestures of their hand and translating those movements into matching mouse operations. Experimental results show the efficiency and precision of the virtual mouse implementation, providing a potential substitute for traditional mice for those with physical limitations or in situations where they are impractical.	• **Privacy and Security:** Capturing and processing video input from a user's environment raises privacy and security concerns. • **Latency and Responsiveness:** Achieving low latency and high responsiveness is crucial for a virtual mouse system to provide a seamless user experience. • **Hand Detection and Tracking:** Accurately detecting and tracking the user's hand in real-time can be challenging due to variations in lighting conditions, hand shapes,

				and occlusions.
7.	AI Gesture Recognition	Joel Abram Santhosh et al.	Using artificial intelligence (AI) methods, the paper "AI Gesture Recognition" proposes a revolutionary method of gesture recognition. The authors suggest a method that uses deep learning algorithms to instantly recognise and categorise hand motions. The system achieves great accuracy in recognising a variety of hand gestures by training a convolutional neural network (CNN) on a huge dataset of labelled gesture images. The suggested method takes into account issues such different lighting conditions, hand angles, and occlusions. Results from experiments show how good the AI gesture recognition system is, showing its	**Data Acquisition and Annotation:** Acquiring a large dataset of labeled gesture images for training the convolutional neural network (CNN) can be a time-consuming and labor-intensive task**.**

			potential use in a variety of industries like robotics, virtual reality, and HCI. The study provides insights into the developments and advances in the growing body of research in AI-based gesture detection.	
8.	A Real-Time Hand Gesture Recognition System for Daily Information Retrieval from Internet	Sheng-Yu Peng et al.	A real-time hand gesture recognition system is presented in the paper "A Real-Time Hand Gesture Recognition System for Daily Information Retrieval from Internet" that enables users to retrieve information from the internet using simple hand gestures. The suggested system detects and recognises hand movements recorded by a camera by combining computer vision techniques with machine learning algorithms. Users can carry out operations like	• **Gesture Variability:** Hand gestures can vary significantly across individuals, making it challenging to develop a system that accurately recognizes a wide range of gestures. • **Lighting and Environmental Conditions:** The performance of the hand gesture recognition system may be affected by varying lighting conditions,

			looking for news, weather updates, or social networking updates by mapping particular motions to predetermined search terms. The testing results show the system's potential for improving routine information retrieval tasks by offering a hands-free and natural interaction technique, demonstrating its effectiveness and efficiency.	cluttered backgrounds, and occlusions • **User Fatigue and Adaptability:** Performing hand gestures continuously for information retrieval tasks may lead to user fatigue over time. • **False Positives and False Negatives:** The hand gesture recognition system may encounter issues with false positives (incorrectly recognizing a gesture) or false negatives (failing to recognize a correct gesture).
10.	Real-time Hand Tracking and Finger Tracking for Interaction	Shahzad Malik et al.	In real-world circumstances, the research accurately detects and tracks hands and fingers using computer vision techniques	• **User Adaptation and Training:** Users may need to adapt their hand movements

			and machine learning algorithms. The results show how well the system can recognise and interpret complex hand movements, enabling natural interaction with virtual surroundings. The study demonstrates the potential of hand and finger tracking as a natural and immersive input technique, opening up opportunities for a variety of applications including gaming, virtual reality, and human-computer interaction.	and gestures to align with the tracking system's requirements. • **Real-Time Processing Constraints:** Achieving real-time hand and finger tracking involves computational challenges. The system must process and analyze video input rapidly to provide responsive tracking results • **Lighting and Environmental Variations:** The system's accuracy and reliability may be affected by variations in lighting conditions, background clutter, and other environmental factors.

3. THEOROTICAL ANALYSIS

The theoretical analysis of this project provides insights into the underlying algorithms, system architecture, and performance expectations. It serves as a foundation for further development, optimization, and practical implementation of the real-time AI virtual mouse system using computer vision techniques.

# gesture_detection.py

- **HandRecog class methods:**
  Convert Mediapipe Landmarks to recognizable Gestures. The get_signed_dist, get_dist, and get_dz are methods that calculate distances between two points in different ways. They use the detected landmarks' x, y, and z coordinates to find distances, which helps in gesture recognition.
- **set_finger_state:**
  This method calculates the finger state (open or closed) based on the distances between certain landmarks. For each finger, the distance ratio between two sets of landmarks is calculated, and if the ratio exceeds a threshold, the finger is considered open. The binary representation of open fingers is stored in the 'self.finger' variable. Function to find Gesture Encoding using current finger_state. Finger_state: 1 if finger is open, else 0

- **get_gesture:**
  This method identifies the gesture based on the finger states and some additional conditions. Depending on the finger states and distances between certain landmarks, it classifies gestures such as PINCH_MAJOR, PINCH_MINOR, V_GEST, and TWO_FINGER_CLOSED.

  The method also handles fluctuations due to noise by maintaining a frame count for consistent gestures.
- **Controller class:** Executes commands according to detected gestures.

- **getpinchylv and getpinchxlv: These** methods calculate the y and x level differences of the pinch gesture by comparing the coordinates of landmark 8 (tip of the index finger) to the starting coordinates of the pinch.

- **changesystemvolume:** This method adjusts the system volume based on the pinch level calculated from the pinch gesture.

- **get_position:** This method calculates the cursor position based on the hand_result, specifically landmark 9 (between index finger and thumb). It stabilizes the cursor using dampening techniques by taking into account the previous cursor position and distance traveled.

- **pinch_control_init:** This method initializes the pinch control by setting starting x and y coordinates, pinch level, previous pinch level, and frame count.

- **pinch_control:** This method handles pinch control actions, which can be either vertical or horizontal based on the detected hand gesture. It maintains a frame count for consistent pinch gestures.

- **handle_controls:** This method executes different actions based on the detected hand gestures, such as moving the cursor, clicking, double-clicking, scrolling, and changing system volume. The actions are performed using the PyAutoGUI library and involve calculations based on hand landmarks.

# handle_controls: This method has been updated to use the execute_action method to execute the appropriate action for a given gesture.

  - **GestureController class methods:**

__init__: Initializes the GestureController object by setting the mode, capturing video from the default camera, and retrieving the camera's frame height and width.

- **classify_hands:** This static method classifies the detected hands as left or right, and updates the hr_major and hr_minor attributes based on the handedness.

- # process_frame: This method processes a video frame, detects hand landmarks, updates the HandRecog objects for major and minor hands, and calls the handle_controls method of the Controller class to perform actions based on the detected gestures. The processed frame with hand landmarks is returned.

# app.py

- capture_frames function: This function captures video frames using the GestureController object and calls the process_frame method to process each frame.

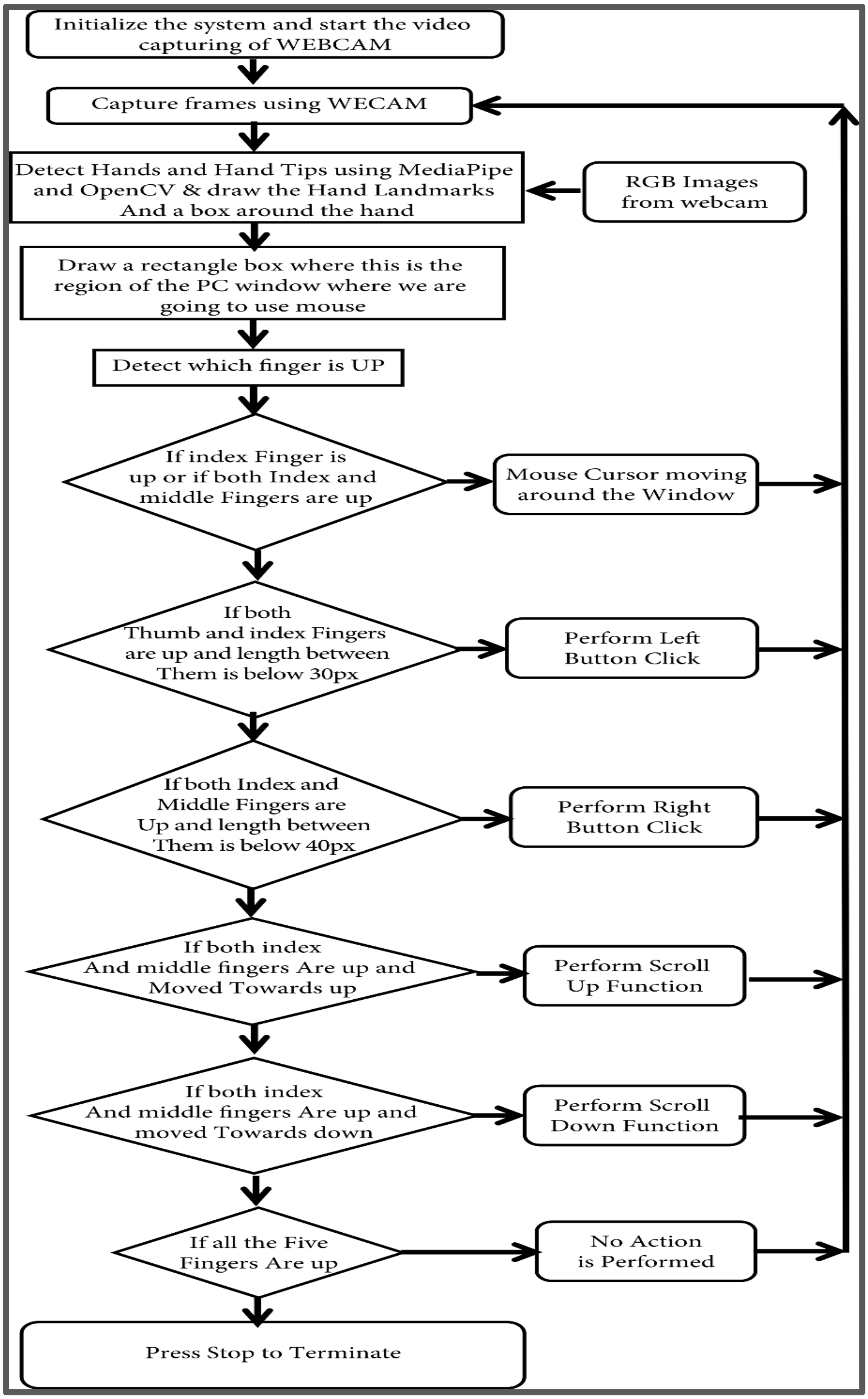
- gen function: This function generates video frames in the form of byte strings that can be used for streaming purposes. It reads frames from the GestureController object, encodes them as JPEG images, and returns the byte strings.

- **Create the camera object:** This snippet of code illustrates a Python programme that uses the Flask framework to construct a gesture-recognition virtual mouse controller. It creates routes for several pages, including the settings page, the virtual mouse controller, and the home page. To display camera frames, the /video_feed route returns a video stream. Based on the form data that is received, **the update_gesture_mappings** route updates the gesture mappings. The activation and deactivation of gesture detection are controlled by the **start_gesture_detection** and stop_gesture_detection routes. On the local host, the programme is running with debugging turned on. In conclusion, this code creates a web application that allows users to manipulate settings and gesture mappings while using a virtual mouse using hand gestures.

4. EXPERIMENTAL INVESTIGATIONS

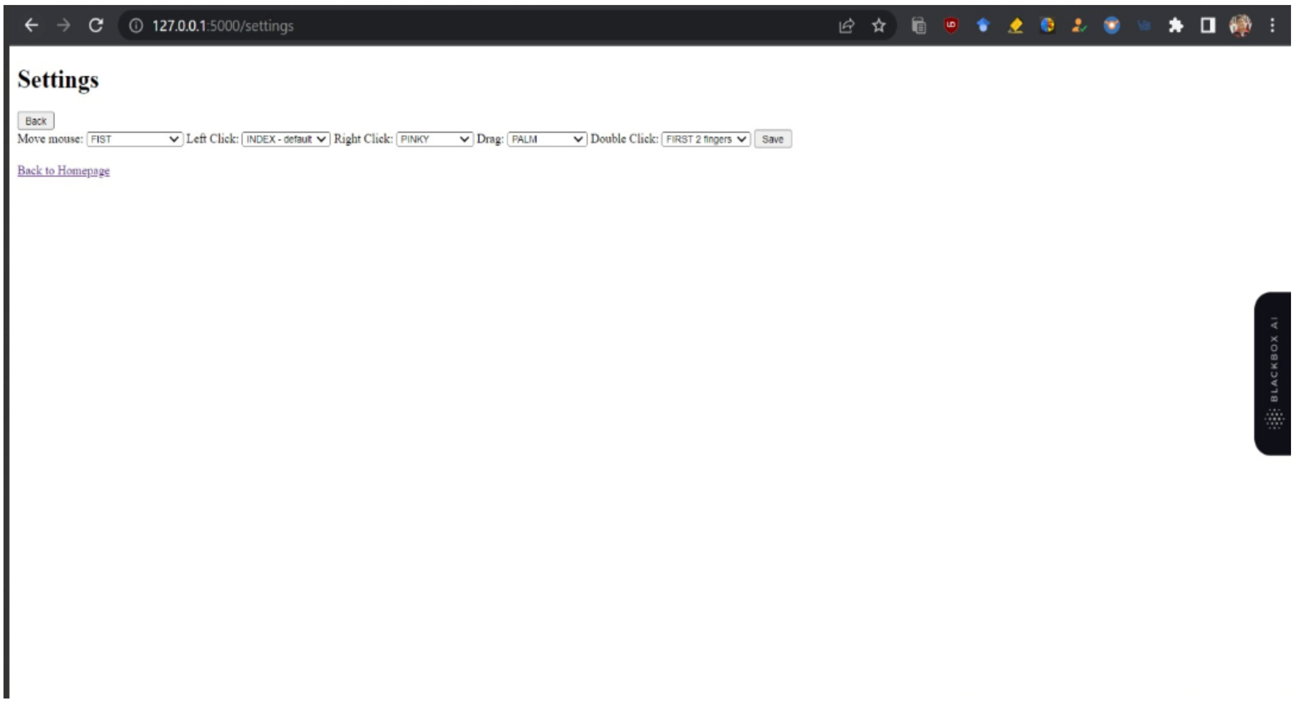
- The initial version of this project involved using if cases inside the while true function to detect current gesture, however having the landmark variables assigned inside the if case led to errors many times as sometime the variables were not being assigned at same time and were not available for distance calculation.
- Then in the next version, we assigned v-gesture for mouse movement and index finger for left-click but they were conflicting when distance was getting calculated due to poor fps input from camera sometimes.
- Hold-mode introduced in next version to keep mouse-position still while clicking.
- Distance values observed to set Thresholds for recognising Gestures.
- Fist gesture for mouse movement introduced and index finger for left click.
- Right-click assigned with pinky finger raising gesture.
- Flask app built and camera-input from webcam passed through flask application.
- Basic html website that caters to function developed.

5. FLOWCHART


Initialize the system and start the video capturing of WEBCAM
Capture frames using WECAM
Detect Hands and Hand Tips using MediaPipe and OpenCV & draw the Hand Landmarks And a box around the hand
RGB Images from webcam
Draw a rectangle box where this is the region of the PC window where we are going to use mouse
Detect which finger is UP
If index Finger is up or if both Index and middle Fingers are up
Mouse Cursor moving around the Window
If both Thumb and index Fingers are up and length between Them is below 30px
Perform Left Button Click
If both Index and Middle Fingers are Up and length between Them is below 40px
Perform Right Button Click
If both index And middle fingers Are up and Moved Towards up
Perform Scroll Up Function
If both index And middle fingers Are up and moved Towards down
Perform Scroll Down Function
If all the Five Fingers Are up
No Action is Performed
Press Stop to Terminate

6. RESULT

WebPage:


127.0.0.1:5000/settings

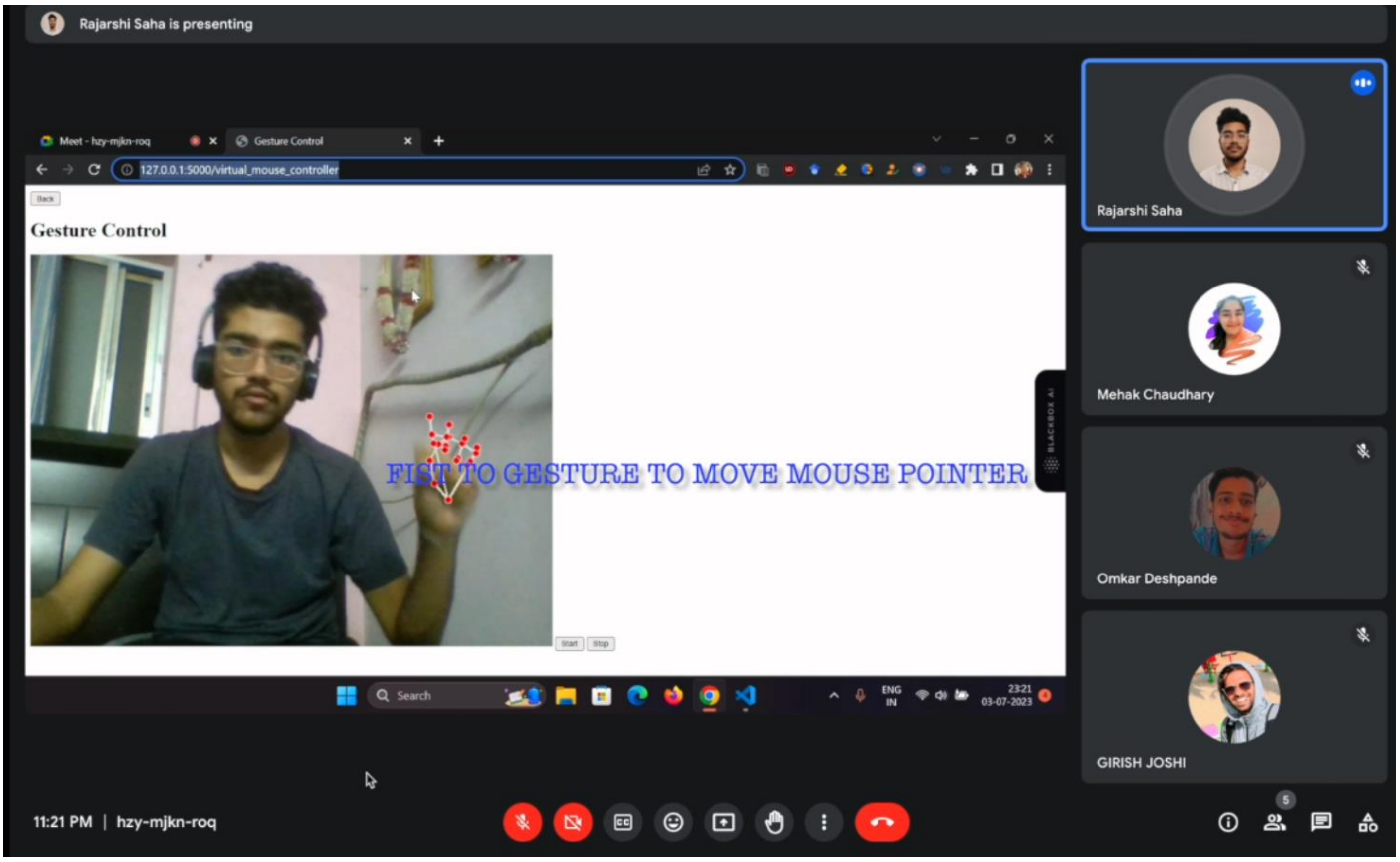
Settings

Back

Move mouse: FIST Left Click: INDEX - default Right Click: PINKY Drag: PALM Double Click: FIRST 2 fingers Save

Back to Homepage

Fist Gesture for moving the mouse pointer

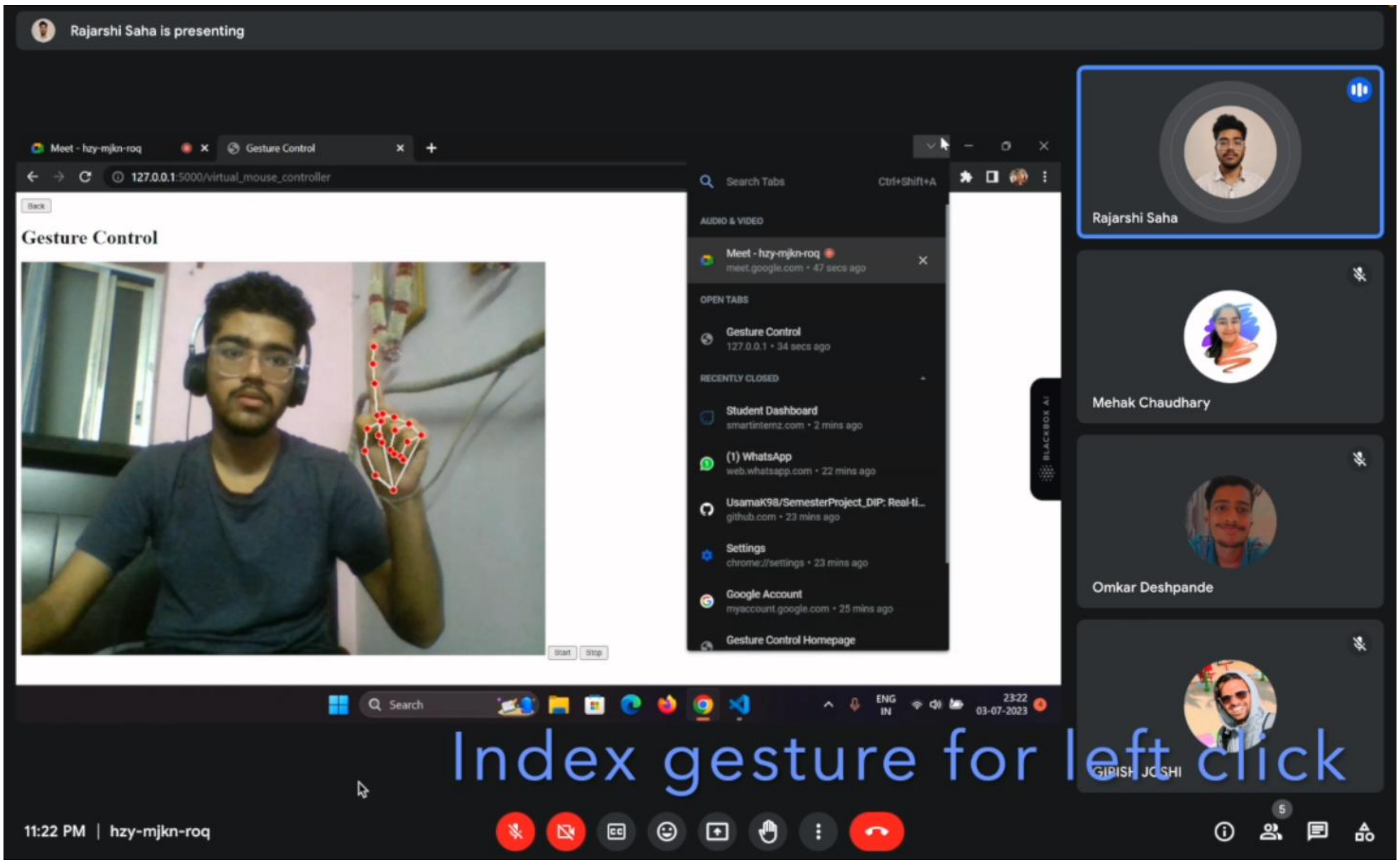

Rajarshi Saha is presenting

127.0.0.1:5000/virtual_mouse_controller

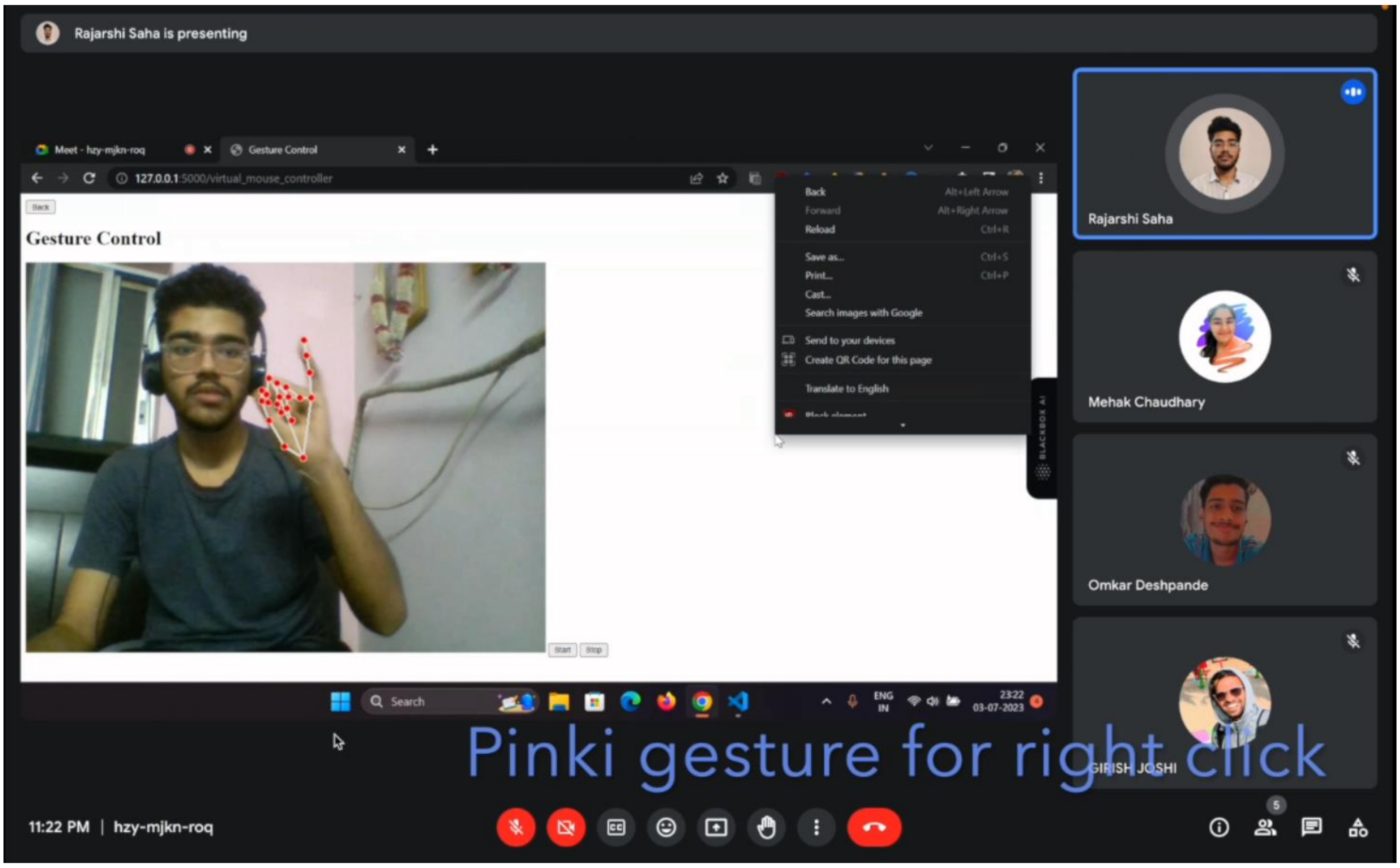
Gesture Control

FIST TO GESTURE TO MOVE MOUSE POINTER

Index gesture for left click



Pinki finger gesture for right click



7. ADVANTAGES/DISADVANTAGES:

ADVANTAGES

Intuitive and Natural Interaction: Gesture control offers a more intuitive and natural manner of communicating with a computer or other device. By just moving their hands, users may control the mouse cursor and carry out tasks that imitate real-world movements. This reduces the learning curve and increases user comfort by doing away with the requirement for physical input devices like a mouse or trackpad.

Hands-Free Operation: Gesture control for a virtual mouse allows for hands-free operation, which is especially advantageous for people with physical limitations or mobility issues. A more accessible computing experience is made possible by the ability of users to explore and interact with a computer or device without the need to physically operate input devices.

Reduced Physical Strain: Using a mouse for an extended period of time puts physical strain on your hands. Repetitive hand movements and holding of physical input devices are no longer necessary, potentially lowering the risk of repetitive strain injuries (RSIs) or other musculoskeletal illnesses.

Novel Interaction Paradigm: Gesture control for a virtual mouse presents a fresh interaction paradigm that can inspire creativity and innovation in application design. In fields like gaming, virtual reality, augmented reality, and immersive simulations, developers can use hand gestures to create distinctive and compelling user experiences.

Future-Proof Technology: Gesture control is a technique that is future-proof and is in line with the expanding trend towards touchless and organic user interfaces. Gesture control and recognition are projected to become more important as technology advances in human-computer interaction.

DISADVANTAGES

Limited Gesture Set: The system may only recognise a small number of the predefined motions. Although unique gestures can be defined, the system's capacity to recognise a variety of gestures may be influenced by their complexity and computational power.
Physical exhaustion: Controlling gestures for extended periods of time, in particular, can cause physical exhaustion or muscle strain in the user's arm or hand. Repeated motions or holding the hand in an elevated position for a long time may make you feel uncomfortable or exhausted.
Environmental Constraints: Background clutter, fluctuating lighting conditions, and interference from nearby objects can all have an impact on how well the hand tracking system performs. In some contexts, ensuring ideal circumstances for precise tracking may be difficult.
Limited Sensing Range: Hand tracking systems often have a restricted sensing range, which could limit the user's range of motion. For accurate tracking, users might need to stay inside a given area or keep a certain distance from the camera or sensor, which might restrict the system's use in some circumstances.
Privacy Issues: Hand tracking systems record users' hand motions in visual form, which in some situations may give rise to privacy issues. User data must be handled and safeguarded with care in compliance with privacy laws and user consent.

8. APPLICATIONS

1. **Human-Computer Interaction:** Hand tracking technology makes it possible for users to interact with computers and other devices in a simple and straightforward way. It may be used to navigate menus, move the pointer, and provide gesture-based commands without touching the user interface.
2. **Virtual Reality (VR) and Augmented Reality (AR):** The hand tracking system allows users to have immersive experiences in virtual reality (VR) and augmented reality (AR) apps by precisely tracking and simulating their hand movements in the virtual or augmented environment. It improves these technologies' realism and interactivity.
3. **Gesture-based gaming:** By integrating the hand tracking system with gaming programmes, users are able to manipulate and communicate with

virtual items and characters using hand gestures. It creates opportunities for more realistic and captivating gameplay.

4. **Sign Language Recognition:** Hand tracking and gesture recognition are essential for sign language interpreting systems, according to sign language recognition. This project can be used to create software that can convert sign language motions into text or spoken language and recognise them, making it easier for deaf or hard-of-hearing people and the general public to communicate.
5. **Robotics and humanoid systems:** Hand tracking allows robots and humanoid systems to recognise and comprehend human hand motions, enhancing their ability to communicate and work together with people. It can be applied to tasks like manipulating objects with the help of a robot, interacting with humans and robots, and controlling prosthetic hands.
6. **Rehabilitation and healthcare:** Hand tracking can be used to support hand therapy exercises and track progress in rehabilitation and healthcare applications. It gives therapists the ability to monitor and assess patients' hand movements for therapeutic purposes.
7. **Education and Training:** The hand tracking technology can be utilised to improve interactive learning environments in educational settings. Students can operate virtual items and complete tasks using their motions in hands-on virtual presentations and simulations.

9. CONCLUSION

The vision-based hand tracking system demonstrated in this project can run in real-time on a standard PC with inexpensive cameras without the use of special markers or gloves. With the assumption that a calibrated pair of cameras is monitoring the hands from above with the palms facing downward, the system is specifically able to track the tip positions of the thumb and index finger for each hand. This hand tracker was developed as part of a desktop-based two-handed interaction system that allows users to choose and alter 3D geometry in real-time using their natural hand gestures. After presenting the hand tracker's algorithmic details, the system's performance and accuracy were discussed, along with some suggestions on how they could be improved.

10. FUTURE SCOPE

- **Multi-Modal Input Integration:**
  The project can be expanded to include multi-modal input, which combines hand motions with other input modalities like voice commands or eye tracking. Through the use of gestures, voice commands, and eye movements, users may be able to operate the virtual mouse and carry out actions for a more thorough and natural engagement.
- **Three-Dimensional interactivity:**
  Adding support for three-dimensional (3D) interactivity to the project can open up new avenues. This might entail giving users the ability to manage virtual things in three dimensions via hand gestures, creating a more intuitive and immersive user experience.
- **Mobile and Wearable technologies:**
  The project's usefulness can be increased by making it compatible with mobile and wearable technologies. In order to provide a mobile and on-the-go engagement experience, this would allow users to manipulate the virtual mouse using hand gestures on smartphones, tablets, or smartwatches.
- **Assistive technology and accessibility:**
  The Gesture Control Virtual Mouse can be improved to accommodate people with disabilities. Accessibility and inclusivity can be considerably improved by adapting the system to assistive technology applications, for as by offering several input ways for people with motor impairments.