

A Project report on

**AIR CANVAS WITH GESTURE CONTROL
USING OPENCV**

*Submitted in partial fulfillment of the requirements
for the award of the degree of*

BACHELOR OF TECHNOLOGY

in

**COMPUTER SCIENCE & ENGINEERING
(ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING)**

By

| | |
|-------------------------|---------------------|
| FARDEEN HUSSAIN | (214G1A3332) |
| L. HEMANTH KUMAR | (224G5A3305) |
| C. DHEERAJ | (214G1A3320) |
| M. HARISH CHAVAN | (214G1A3327) |

Under the Guidance of

Dr. P. Chitralingappa M. Tech., Ph. D

Associate Professor



**Department of Computer Science & Engineering
(Artificial Intelligence & Machine Learning)**

**SRINIVASA RAMANUJAN INSTITUTE OF TECHNOLOGY
(AUTONOMOUS)**

Rotarypuram village, B K Samudram Mandal, Ananthapuramu-515701.

2024-2025

SRINIVASA RAMANUJAN INSTITUTE OF TECHNOLOGY

(AUTONOMOUS)

(Affiliated to JNTUA, Accredited by NAAC with 'A' Grade, Approved by AICTE, New Delhi & Accredited by NBA (EEE, ECE & CSE))

Rotarypuram Village, BK Samudram Mandal, Ananthapuramu-515701

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING (ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING)



Certificate

This is to certify that the project report entitled **AIR CANVAS WITH GESTURE CONTROL USING OPENCV** is the bonafide work carried out by **Fardeen Hussain, L. Hemanth Kumar, C. Dheeraj, M. Harish Chavan** bearing Roll Number **214G1A3332, 224G5A3305, 214G1A3320, 214G1A3327** in partial fulfilment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science and Engineering (Artificial Intelligence & Machine Learning)** during the academic year 2024-2025.

Project Guide

Dr. P. Chitralingappa MTech., Ph.D

Associate Professor

Head of the Department

Dr. P. Chitralingappa MTech., Ph.D

Associate Professor

Date:

Place: Rotarypuram

External Examiner

DECLARATION

We Mr. Fardeen Hussain reg no : 214G1A3332, Mr. L. Hemanth Kumar bearing reg no : 224G5A3305, Mr. C. Dheeraj bearing reg no : 214G1A3320, Mr. M. Harish Chavan bearing reg no : 214G1A3327 students of **SRINIVASA RAMANUJAN INSTITUTE OF TECHNOLOGY**, Rotarypuram, hereby declare that the dissertation entitled “**AIR CANVAS WITH GESTURE CONTROL USING OPENCV**” embodies the report of our project work carried out by us during IV Year Bachelor of Technology under the guidance of **Dr. P. Chitralingappa** MTech., Ph.D, Associate Professor, Department of CSE(AI-ML & Data Science) and this work has been submitted for the partial fulfillment of the requirements for the award of Bachelor of Technology degree.

The results embodied in this project report have not been submitted to any other Universities of Institute for the award of Degree.

FARDEEN HUSSAIN

Reg no: 214G1A3332

L. HEMANTH KUMAR

Reg no: 224G5A3305

C. DHEERAJ

Reg no:214G1A3320

M. HARISH CHAVAN

Reg no:214G1A3327

Vision & Mission of the SRIT

Vision:

To become a premier Educational Institution in India offering the best teaching and learning environment for our students that will enable them to become complete individuals with professional competency, human touch, ethical values, service motto, and a strong sense of responsibility towards environment and society at large.

Mission:

- Continually enhance the quality of physical infrastructure and human resources to evolve in to a center of excellence in engineering education.
- Provide comprehensive learning experiences that are conducive for the students to acquire professional competences, ethical values, life-long learning abilities and understanding of the technology, environment and society.
- Strengthen industry institute interactions to enable the students work on realistic problems and acquire the ability to face the ever-changing requirements of the industry.
- Continually enhance the quality of the relationship between students and faculty which is a key to the development of an exciting and rewarding learning environment in the college.

Vision & Mission of the Department of CSE

Vision:

To evolve as a leading department by offering best comprehensive teaching and learning practices for students to be self-competent technocrats with professional ethics and social responsibilities.

Mission:

- DM 1:** Continuous enhancement of the teaching-learning practices to gain profound knowledge in theoretical & practical aspects of computer science applications.
- DM 2:** Administer training on emerging technologies and motivate the students to inculcate self-learning abilities, ethical values and social consciousness to become competent professionals.
- DM 3:** Perpetual elevation of Industry-Institute interactions to facilitate the students to work on real-time problems to serve the needs of the society.

ACKNOWLEDGEMENT

The satisfaction and euphoria that accompany the successful completion of any task would be incomplete without the mention of people who made it possible, whose constant guidance and encouragement crowned our efforts with success. It is a pleasant aspect that we have now the opportunity to express my gratitude for all of them.

It is with immense pleasure that we would like to express my indebted gratitude to my Guide **Dr. P. Chitralingappa, Associate Professor & Head of the Department, Computer Science & Engineering (AI-ML & Data Science)**, who has guided me a lot and encouraged me in every step of the project work. We thank him for the stimulating guidance, constant encouragement and constructive criticism which have made possible to bring out this project work.

We express our deep felt gratitude to **Mr. A. Kiran Kumar, Assistant Professor, CSE (AI & ML)** and **Mrs. S. Sunitha, Assistant Professor, CSE**, project coordinators for their valuable guidance and unstinting encouragement enabled us to accomplish our project successfully in time.

We are very much thankful to **Dr. P. Chitralingappa, Associate Professor & Head of the Department, Computer Science & Engineering (AI- ML & Data Science)**, for his kind support and for providing necessary facilities to carry out the work.

We wish to convey my special thanks to **Dr. G. Bala Krishna, Principal of Srinivasa Ramanujan Institute of Technology** for giving the required information in doing my project work. Not to forget, we thank all other faculty and non-teaching staff, and my friends who had directly or indirectly helped and supported me in completing my project in time.

We also express our sincere thanks to the Management for providing excellent facilities. Finally, we wish to convey our gratitude to our family who fostered all the requirements and facilities that we need.

Project Associates

214G1A3332

224G5A3305

214G1A3320

214G1A3327

ABSTRACT

With Air Canvas users can draw on screens through simple hand movements while they do not need to physically touch it. Index finger serves as the main drawing instrument in this system yet users can switch tools without creating accidents by keeping their second finger shut. A fist gesture has been built into the system to instantly delete everything drawn on the canvas. Despite delivering high-precision landmark detection MediaPipe's Hand Tracking module enables better control over smooth and precise drawing performance. User interaction receives improvement through the system's implementation of computer vision methods like color detection and contour extraction and tracking algorithms. Air Canvas enables users to create drawings through touchless gestures that find applications in educational settings as well as design and assistive technology fields.

Keywords- *Air Writing, Motion-to-Text Conversion, Computer Vision, Object Tracking, Human-Machine Interaction.*

Contents

| | |
|-------------------------------------|-------------|
| List of Figures | x |
| List of Tables | xi |
| List of Abbreviations | xii |
| List of Symbols | xiii |
| Chapter 1: Introduction | 1-2 |
| 1.1 Air Canvas Technology | 1 |
| 1.2 Objective | 1 |
| 1.3 Problem Definition | 2 |
| Chapter 2: Literature Survey | 3-4 |
| Chapter 3: Planning | 5-12 |
| 1.4 Existing System | 5 |
| 1.5 Mediapipe | 6 |
| 1.6 Disadvantages | 6 |
| 3.1 Proposed System | 7 |
| 3.1.1 System Architecture | 7 |
| 3.1.2 Advantages | 8 |
| 3.2 Hardware Requirements | 8 |
| 3.3.1 Personal Computer | 8 |
| 3.3.2 Camera | 9 |
| 3.3 Functional Requirements | 9 |
| 3.4 Non - Functional Requirements | 9 |
| 3.4.1 Performance Requirements | 9 |
| 3.4.2 Usability Requirements | 10 |
| 3.5 Scope | 10 |
| 3.6 Performance | 10 |
| 3.7 Methodology | 11 |
| 3.7.1 Advantages | 12 |
| 3.8 Time Estimation | 12 |

| | |
|---|--------------|
| Chapter 4: Design | 13-16 |
| 4.1 Work Flow | 13 |
| 4.2 Hand Detection | 14 |
| 4.3 System Architecture | 15-16 |
| Chapter 5: Implementation | 17-25 |
| 5.1 Hardware Implementation | 22 |
| 5.1.1 Computer-Based System | 22 |
| 5.1.2 Computer System Hardware | 22 |
| 5.1.3 Gesture Detection System | 23-24 |
| 5.2 Software Implementation | 19 |
| 5.2.1 Installation of VScode IDE and tool demonstration | 19-25 |
| Chapter 6: Testing | 26-29 |
| 6.1 Testing Approach | 26 |
| 6.2 Features to be Tested | 26 |
| 6.3 Testing tools and Environment | 27 |
| 6.4 Test Cases | 27 |
| 6.4.1 Inputs | 27 |
| 6.4.2 Extended output | 28 |
| 6.4.3 Testing Procedure | 28 |
| Chapter 7: Result | 30-34 |
| Conclusion | 35 |
| References | 36-37 |
| Publication | 38 |

List of Figures

| Fig.No. | Description | Page No. |
|----------------|---|-----------------|
| 3.1 | Architecture of the proposed system | 7 |
| 4.1 | Workflow | 13 |
| 4.2 | Block diagram of Hand Detection | 14 |
| 4.3 | System Architecture | 15 |
| 5.1 | Computer System Hardware | 18 |
| 5.2 | Schematic diagram of Gesture Detection System | 18 |
| 7.1 | User Interface with tools and colors | 30 |
| 7.2 | Tool Selection with the index finger | 31 |
| 7.3 | Freehand draw tool with interpolation | 31 |
| 7.4 | Eraser tool for erasing the sketch | 32 |
| 7.5 | Line tool for drawing straight lines | 32 |
| 7.6 | Rectangle tool for drawing rectangles | 33 |
| 7.7 | Circle tool for drawing circles | 33 |
| 7.8 | Fist Gesture for clearing the board | 34 |

List of Tables

| TableNo. | Table Name | Page No. |
|-----------------|-------------------|-----------------|
| 3.1 | Time Estimation | 12 |

LIST OF ABBREVIATIONS

| | |
|-----|-----------------------------------|
| AI | Artificial Intelligence |
| API | Application Programming Interface |
| CV | Computer Vision |
| CNN | Convolutional Neural Network |
| CPU | Central Processing Unit |
| FPS | Frames Per Second |
| GDG | Gesture Detection and Generation |
| IoT | Internet of Things |
| ML | Machine Learning |
| RF | Radio Frequency |
| RGB | Red Green Blue (Color Model) |
| ROI | Region of Interest |
| UI | User Interface |

LIST OF SYMOBLS

| | |
|---------------------------|--------------------------------|
| G | Gesture Input |
| Δ | Change in position or movement |
| X, Y | Coordinates in the 2D plane |
| $\rightarrow \rightarrow$ | Direction of movement |

CHAPTER-1

INTRODUCTION

1.1 Air Canvas Technology

Computer vision along with artificial intelligence research has fostered fresh methods of interacting between humans and computers which can function without physical touching devices.

Users can now employ the Air Canvas system to draw on digital screens by performing basic hand motions which eliminates manual screen interaction.

The touch-free drawing feature of this system enables useful applications across educational institutions and laboratories that specialize in digital art and assistive technology fields.

A combination of OpenCV for real-time processing and hand tracking enables the Air Canvas system which uses MediaPipe's Hand Tracking module to improve recognition efficiency.

Index finger operation detects drawings and second finger closure selects tools without producing unwanted marks. Users can easily clear their canvas through the implementation of a fist gesture for an improved user experience.

The Air Canvas achieves exact and smooth drawing with its implementation of color detection methods and contour extraction algorithms together with tracking algorithms.

A paperless world can be created using this application where there is only need for a device and a hand. Computer vision has proved its capability for developing interactive user interfaces during this project as it sets precedence for new applications in digital accessibility.

1.2 Objective

So, our aim is to Develop a gesture based virtual canvas to enhance drawing and writing interaction digitally without needing expensive technology, effectively reducing costs and increasing smooth user experience.

1.3 Problem Definition

Now a days Digital learning is the main factor in education. Most of the education sector uses virtual screens for explaining their thoughts. It requires a tablet and a smart pen for interacting with the screen, which is not a viable option if expense is considered. There were studies conducted in Computer Vision to understand the use of hand landmarks for drawing using the movement of them without needing any extra hardware. Numerous advancements can be obtained using the air canvas technology.

CHAPTER-2

LITERATURE SURVEY

[1] Y. Huang, X. Liu, X. Zhang, and L. Jin, "A dataset, approach, and application for a pointing gesture-based egocentric interaction system," Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW), Las Vegas, NV, USA, 2016, pp. 370–377.

The method proposed utilizes a Kinect sensor to collect data the data can be of the type color and depth for gesture recognition. While using a Kinect sensor simplifies gesture recognition, implementing it without any physical equipment presents significant challenges. The Kinect sensor has a high resolution, making it effective for recognizing larger objects; however, identifying smaller objects, such as fingers, is more difficult. In our project, we rely on finger-based navigation, which adds complexity due to the relatively small size of the fingers

[2] X. Liu, Y. Huang, X. Zhang, and L. Jin. A cascading CNN pipeline for real-time fingertips Self-centered video recognition", Corr, 2015.

The approach presented by the authors tells us about an LED on the tip of an index finger for tracking the course of it. The written character is identified using the data stored in the database and returned to the user. This method is quite useful but it assumes that there are no other red lights present except the tip of your index finger.

[3] Yuan-Hsiang Chang, Chen-Ming Chang, "Automatic Hand-Pose Detection Tracking System Using Video Serise", INTECH, Croatia, 2010.

An augmented desktop interface was used. The description of the interface involves a CCD camera and a video projector where the display is shown. The hands are marked as a unique function in this paper. There are two important functionalities, the right hand is used for choosing items and the left one is for circular menus. It makes use of an infrared camera to accomplish this. This method of implementing costs a lot of money as it has a lot of hardware involved.

[4] Saoji, S., Dua, N., Choudhary, A. K., & Phogat, B. (2021). Air canvas application using Opencv and numpy in python. IRJET, 8(08).

The air canvas system of reference tracks hand movements in real time to allow users draw touch-free making it intuitive. Through a basic camera setup the system recognizes the position of the index finger and converts finger movements into digital drawing strokes for the virtual canvas. Through this system users can forego traditional input tools which makes it suitable for multiple creative activities as well as educational and accessibility needs.

[5] Pratik N., Lowlesh Y., Neehal J., (2024) Air canvas application using Opencv and numpy in python. IJFIAHM.

The air scripting system described lets users write without touching any surface through real-time hand movement detection using computer vision methods. The system achieves digital canvas strokes through webcam frame capturing and subsequent color detection and contour tracking which identify markers and fingers for virtual drawing. The program enables different color selection along with drawing erasure and file-saving functionality through a combination of OpenCV and NumPy. Through this system users can access digital content intuitively by eliminating the necessity of physical input devices giving it optimal value in creative projects and learning contexts and accessibility programs.

CHAPTER-3

PLANNING

3.1 Existing System

Kinetic sensors play a crucial role in computer vision applications, particularly in hand-tracking systems. These sensors, often attached to the fingertips like a glove, accurately capture hand movements, enabling precise tracking. They are widely used in gesture recognition and interactive applications, enhancing the overall experience of virtual interactions. However, their integration into air canvas systems remains limited, as most implementations focus solely on basic functionalities like writing with an index finger and clearing the board.

The existing air canvas systems lack advanced features and gestures, making them less versatile for users. While these systems primarily support simple writing, functionalities such as shape drawing, erasing, and adjusting brush size are often absent. Moreover, adapting to air writing can be challenging, as users may struggle to grasp the available features in their first attempt. A user-friendly interface is essential to simplify the learning curve and improve usability, ensuring that users can easily navigate and utilize all available tools.

Current computer vision-based tools also face limitations in gesture recognition and overall functionality. Implementing multiple gestures and expanding tool options can significantly enhance the air canvas system, making it more adaptable to various user needs. By integrating intuitive controls and improving the interface, the system can offer a seamless and interactive experience. IoT and AI-driven advancements in gesture recognition can further optimize performance, ensuring that air canvas technology evolves into a more efficient and user-friendly solution.

3.1.1 Mediapipe:

- MediaPipe is a real-time hand-tracking framework that enables air canvas applications without extra hardware. It detects hand movements using a standard camera for seamless interaction.
- With MediaPipe, users can draw, erase, and select tools using simple gestures. Its fast processing and accuracy make air writing smooth and intuitive. The driver tends to wear a helmet only where they anticipate checking may take place or else they don't wear a helmet.

3.1.2 Disadvantages

- MediaPipe's accuracy can be affected by poor lighting conditions, leading to inconsistent hand tracking.
- Cluttered or complex backgrounds can also interfere with tracking, reducing precision.
- While it works well for basic gestures, it may struggle with complex hand movements or overlapping fingers.
- Although lightweight, real-time tracking can still be demanding on low-end devices, causing lag or reduced performance.

3.2 Proposed system

The proposed system consists of multiple modules working together to create an interactive air canvas. The **Canvas User Interface** provides essential drawing tools such as line, rectangle, freehand, circle, and eraser, along with a color selection panel offering six different colors. Users interact with the canvas through three gestures—index finger, middle finger, and fist—which control tool selection, drawing, and clearing the board. This intuitive gesture-based interface eliminates the need for physical contact, making the system more interactive and user-friendly.

The **Camera Module** captures real-time video frames using the computer's camera and feeds them into the **Hand Tracking Module**, which leverages MediaPipe for precise gesture recognition. MediaPipe's real-time hand-tracking capabilities detect finger positions, allowing users to draw, select colors, and clear the board effortlessly. This efficient, lightweight system ensures smooth operation, enabling a touch-free and responsive drawing experience using only

a standard camera.

3.2.1 System Architecture

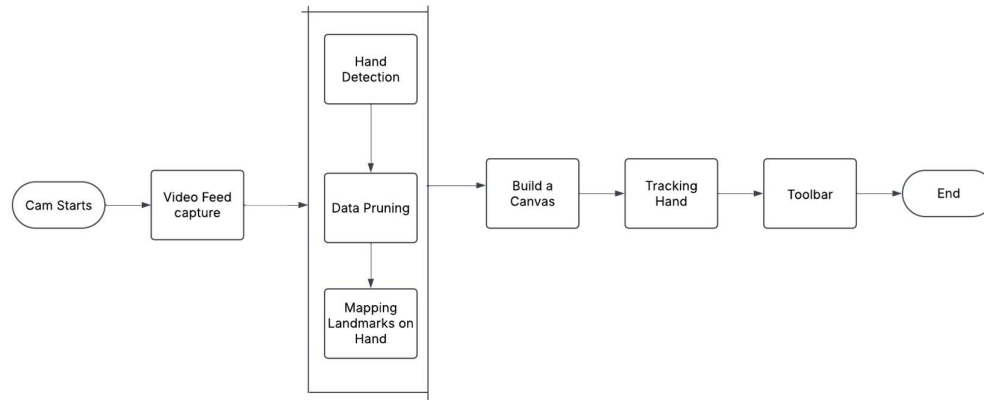


Fig. No. 3.1: Architecture of the proposed system

3.2.2 Advantages

- The proposed system offers several advantages, making air canvas interaction more efficient and user-friendly.
- It provides a touch-free experience, reducing the need for physical contact with devices.
- With real-time hand tracking, users can interact seamlessly using intuitive gestures.
- The system supports multiple drawing tools and color options, enhancing creativity and flexibility.
- Its lightweight design ensures smooth performance, even on standard devices without additional hardware.
- Additionally, the use of MediaPipe-based gesture recognition improves accuracy, making the system highly responsive and easy to use.

3.3 Hardware Requirements

The hardware requirements for an Air Canvas system are relatively minimal but essential for accurate and responsive performance. At the core, a webcam or any camera capable of capturing live video is required to track hand gestures—preferably with a resolution of at least 720p for better precision. A computer with a decent processor (Intel i5 or equivalent) and at least 4GB of RAM ensures smooth real-time video processing. Additionally, a GPU is beneficial for running MediaPipe efficiently, especially when handling complex

hand-tracking models. A display monitor is also necessary to visualize the canvas output.

3.3.1 Computer or Embedded System

For running an air canvas system using MediaPipe, a laptop or PC with decent processing power is recommended. You'll need at least an Intel Core i5 (10th Gen or newer) or AMD Ryzen 5 processor, 8GB of RAM (16GB for smoother performance), and an integrated or dedicated GPU for better real-time tracking. A 1080p webcam ensures accurate gesture recognition, and the system should run on Windows, macOS, or Linux. This setup provides a balance between performance and flexibility, making it ideal for development and testing.

3.3.2 Camera

A camera is essential for the air canvas system, as it tracks hand movements in real time. A 1080p webcam is recommended for accurate gesture detection, ensuring clear visibility of the fingers. Popular choices include the Logitech C920 or C922, which offer good resolution and frame rates. If using a Raspberry Pi, the Pi Camera Module v2 is a great option. A smartphone camera can also work via apps like DroidCam. The better the camera quality, the smoother and more precise the gesture recognition will be.

3.4 Functional Requirements

The air canvas application should enable real-time fingertip tracking for air-writing, allowing users to draw or write in the air with their finger. It must process live video input using OpenCV, detect fingertip positions accurately, and translate motion into digital strokes. The system should support gesture recognition for undo, erase, and clear functions while ensuring smooth and continuous tracking with minimal latency. It should provide an intuitive interface for users to visualize their drawings and allow saving or exporting the output.

3.5 Non Functional Requirements

Non-functional requirements for an Air Canvas include real-time responsiveness with low latency, high accuracy in gesture recognition, and overall system reliability. The interface should be simple and user-friendly, ensuring ease of use. The application must also be lightweight to run smoothly on various devices and be easy to update or maintain.

3.5.1 Performance Requirements

The air canvas system should process hand gestures in real-time, ensuring a smooth drawing experience. To achieve this, the system must run at a minimum of 30 frames per second (FPS), allowing for seamless motion tracking. Additionally, the response time between detecting a gesture and rendering the corresponding stroke on the canvas should be less than 100 milliseconds to avoid noticeable delays. This ensures that the user experiences fluid and natural drawing interactions without lag.

3.5.2 Usability Requirements

The system must be intuitive and easy to use, even for first-time users. The interface should require minimal setup, allowing users to start drawing without calibration or manual configurations. A simple UI with clear options, such as color selection and canvas clearing, will improve the user experience. The system should also be responsive to different hand sizes and lighting conditions, ensuring accessibility for a wide range of users.

3.6 Scope

The Scope of the Air Canvas System is to provide a touch-free, intuitive drawing experience using hand gestures, making it useful in education, digital art, and assistive technology. By leveraging MediaPipe's hand-tracking technology, it ensures smooth and accurate interaction. The system enhances learning, creativity, and accessibility, while also having potential applications in gaming, virtual reality (VR), and smart interfaces, offering a seamless and interactive way to control digital environments.

3.7 Performance

The performance of the Air Canvas System is determined by its speed, accuracy, and user experience. Real-time gesture tracking ensures smooth and responsive drawing, with minimal lag for a seamless interaction. The system's gesture recognition accuracy plays a crucial role in maintaining precision, allowing users to draw effortlessly without disruptions. Reliability and ease of use are essential, ensuring stable performance across different lighting conditions and camera qualities. Optimized resource utilization enhances efficiency, preventing slowdowns on various hardware configurations. Compatibility with multiple platforms and cameras improves accessibility, while an intuitive interface ensures a user-friendly experience. Security and

privacy considerations prevent unauthorized data storage, safeguarding user information. Overall, the Air Canvas System's performance is defined by its responsiveness, accuracy, reliability, and ease of use, making it a versatile and practical tool across different applications

3.8 Methodology

The methodology for implementing Air Canvas using OpenCV and Mediapipe typically involves several key steps:

Literature Review: Conduct a detailed study of existing gesture-based drawing systems, particularly those using MediaPipe. Identify key technologies, challenges, and methodologies used in similar projects

System Design: Define the overall architecture, specifying the hardware (camera, processing unit) and software components. Design the gesture recognition framework and establish the interaction flow for real-time drawing.

Camera and Gesture Integration: Integrate the chosen camera hardware and implement MediaPipe's hand-tracking module. Ensure accurate calibration and alignment for precise finger tracking

Data Processing and Analysis: Develop algorithms to process real-time gesture data, converting hand movements into digital strokes on the canvas. Optimize processing to minimize lag and enhance responsiveness.

1. Testing and Validation: Conduct thorough testing under different lighting conditions, camera angles, and hand positions to evaluate accuracy and responsiveness. Validate performance through user trials.

2. Performance Optimization: Optimize the system for low latency, efficient power consumption, and seamless operation on different hardware setups. Address performance bottlenecks through continuous refinement

This methodology provides a structured approach to implementing the Air Canvas System, ensuring a systematic and efficient development process for real-time gesture-based drawing.

3.8.1 Advantages

- It is easily acceptable to ever-changing needs of the project.
- Testing and debugging during smaller iteration is easy.

- A parallel development can plan.

3.9 Time Estimation

| S. No | Activity | Duration |
|-------|---|----------|
| 1 | Research and Planning | 1 Week |
| 2 | Setting up the environment | 2 Weeks |
| 3 | Implementing Handtracking | 2 Weeks |
| 4 | Implementing drawing functionality | 2 Weeks |
| 5 | Adding Color Selection and Clear Gestures | 1 Week |
| 6 | User Interface and Enhancements | 1 Week |
| 7 | Final Touches and Deployment | 1 Week |

CHAPTER-4

DESIGN

4.1 Workflow

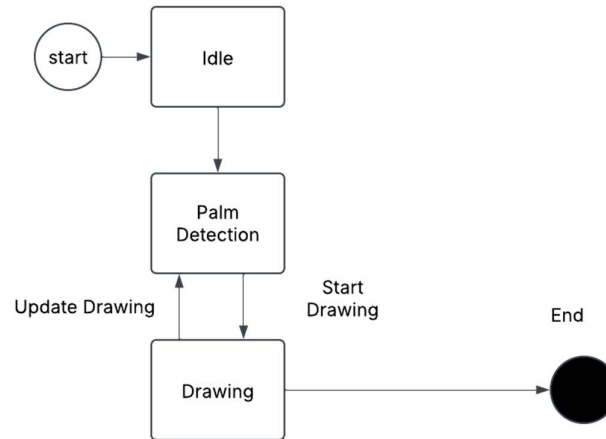


Fig. No. 4.1 Workflow

The Air Canvas system begins in an idle state, waiting for user interaction. Once it detects the user's hand through Palm Detection, it activates the Drawing mode. In this mode, the system tracks hand movements, allowing the user to draw shapes or write in the air as if using an invisible canvas. The drawings appear on a screen in real time, creating an interactive and immersive experience. If the system no longer detects hand gestures, it returns to the idle state, ensuring efficient operation. This contactless technology offers a fun and intuitive way to create digital art without physical tools.

4.2 Hand Detection

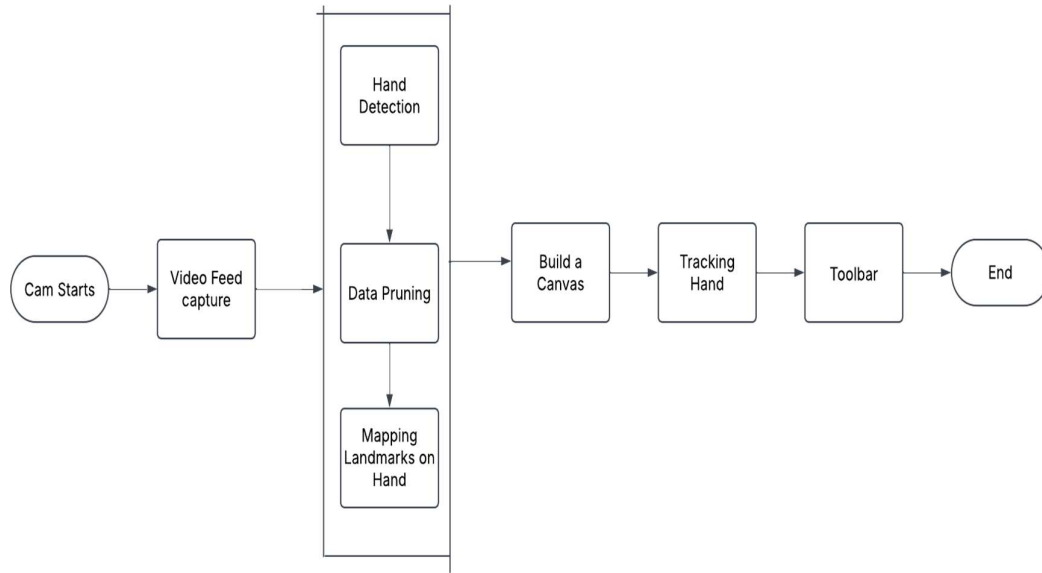


Fig. No. 4.2 Block diagram of Hand Detection

The system begins by activating the camera to capture live video feed. Each frame is analyzed in real-time to detect hands using motion and color segmentation. Once identified, the hand is isolated from the background, and key features like fingertips and palm position are tracked. These movements are then translated into digital strokes on the virtual canvas. If no hand is detected for a period, the system resets to idle mode, ready for the next interaction. This seamless process enables precise, contactless drawing in air.

4.3 System Architecture

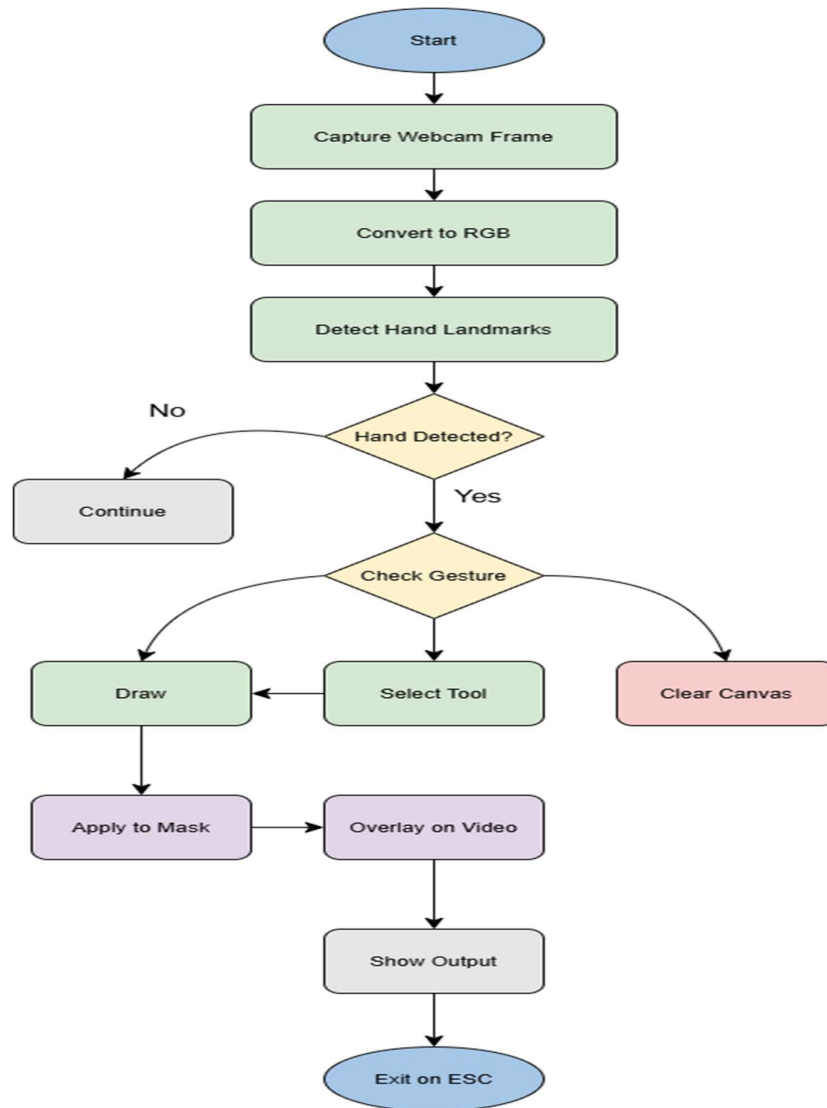


Fig 4.3 System Architecture

The process begins by capturing a frame from the webcam, which is then converted to an RGB format to ensure compatibility with computer vision models used for hand detection. Once the frame is prepared, the system detects hand landmarks, identifying key points such as fingertips and joints. If no hand is detected in the frame, the system continues capturing new frames without further processing, ensuring a smooth and efficient loop. However, if a hand is detected, the program proceeds to analyze the gesture being performed. This step is crucial as it determines the user's intended action, whether it be drawing,

selecting a tool, or clearing the canvas. Gesture recognition allows the system to interpret different hand positions and movements, making it possible to provide an intuitive, touch-free interaction.

Once the gesture is recognized, the system executes the corresponding action. If the user's gesture indicates drawing, the system applies the strokes to a mask, which is separate from the original video feed to preserve the live view while enabling real-time sketching. If a tool selection is detected, the user is allowed to switch between different functionalities, such as changing brush size or color, before proceeding with drawing. In the case of a clear canvas gesture, the mask is reset, removing all previous drawings and starting fresh. After the drawing or tool selection is finalized, the modified mask is overlaid onto the video feed, merging virtual strokes with the real-time webcam stream. The system then displays the processed frame to the user, providing immediate visual feedback. This loop continues as new frames are captured and processed until the user presses the ESC key, signaling the program to terminate execution.

CHAPTER-5

IMPLEMENTATION

5.1 Hardware Implementation

The hardware implementation of the Air Canvas involves setting up a webcam or built-in camera to capture the user's hand movements. This camera feeds real-time video to the system, which processes the frames using computer vision techniques. The computer, equipped with sufficient processing power and memory, runs the hand-tracking model—typically using MediaPipe. The output is displayed on a monitor or screen, showing the drawn content as the user moves their finger in the air. No physical drawing tools are needed, making the setup simple and cost-effective.

5.1.1 Computer- Based system:

The Air Canvas operates on a standard computer system designed to access, process, store, and control data in real time for gesture-based interaction. It utilizes a combination of hardware (webcam) and software (OpenCV, machine learning models) to track fingertip movements and enable air-writing functionality.

5.1.2 Computer System Hardware:

The Air Canvas operates on a standard computer system that serves as its hardware platform. Unlike embedded systems, it does not rely on microcontrollers but utilizes a general-purpose computer with necessary peripherals to perform real-time fingertip tracking and air-writing. The hardware components include:

- Power Supply
- Central Processing Unit (CPU)
- Memory (RAM & Storage)
- Input/Output interfaces (USB, HDMI, etc.)
- Display screen for visualization
- System-specific peripherals (mouse, keyboard, etc.)
- Webcam for real-time tracking

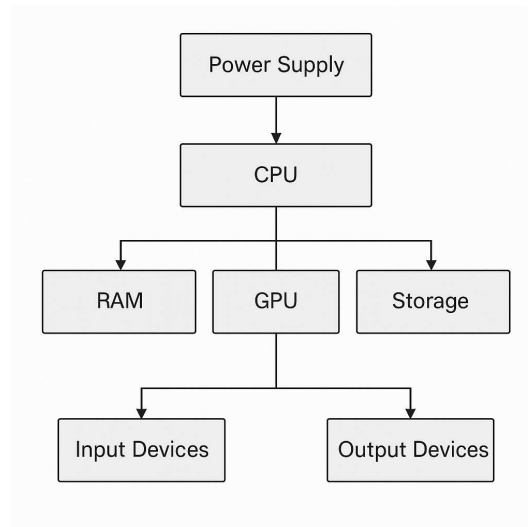


Fig. No. 5.1 Computer System Hardware

5.1.3 Gesture Detection System

We are using MediaPipe for gesture recognition because it provides accurate hand tracking. The system detects the position of the index finger in real time and identifies gestures based on its movement. It continuously processes video frames to track finger positions and trigger interactions, such as selecting colors or clearing the canvas.

Air Canvas System

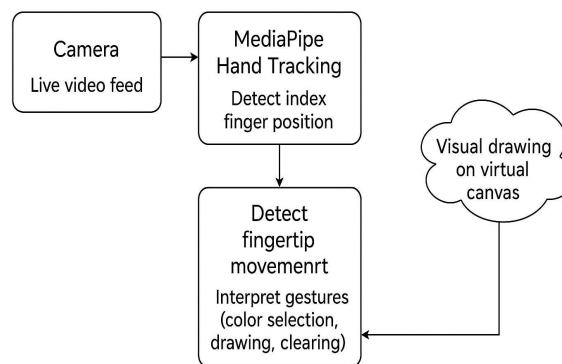


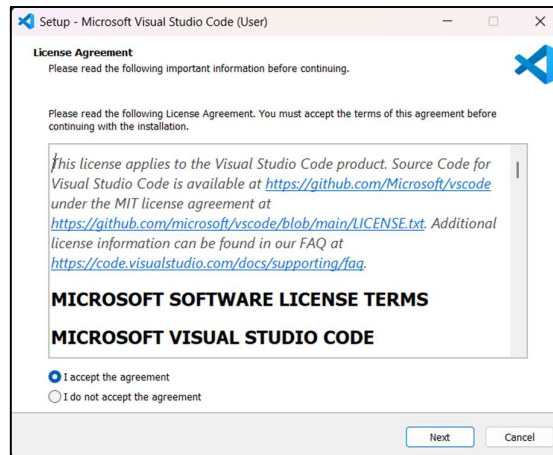
Fig. No. 5.2 Schematic Diagram of Gesture Detection System

5.2 Software Implementation

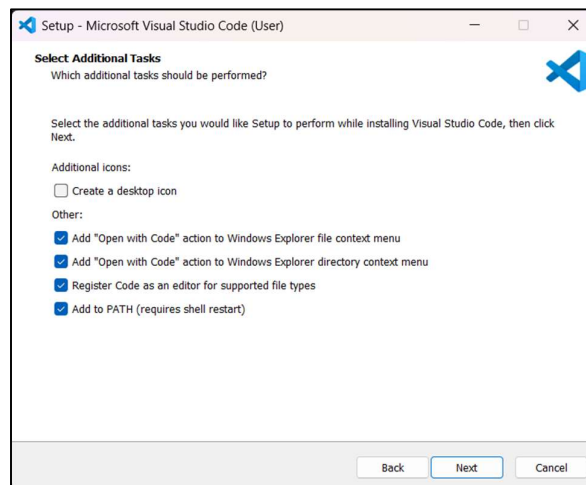
5.2.1 Installation of VS Code IDE and Tool Demonstration

To install VSCode on your Windows PC, follow the next instructions

- Insert the CD-ROM or PENDRIVE that contains the VSCode setup file.
- Copy the VSCode Setup File to your desired location on your computer.
- After copying, double-click on the setup file to start the installation.

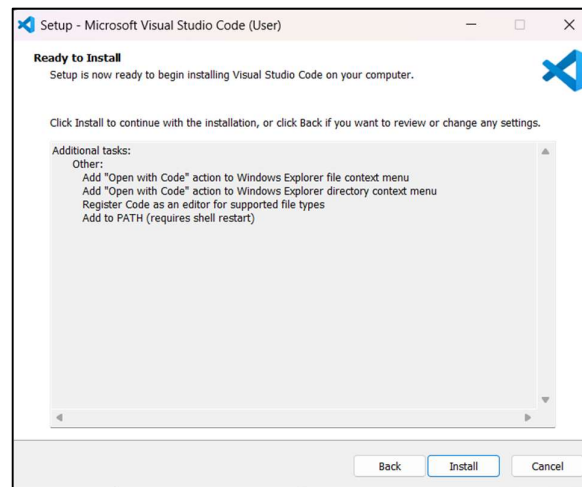


- Click on I accept and next

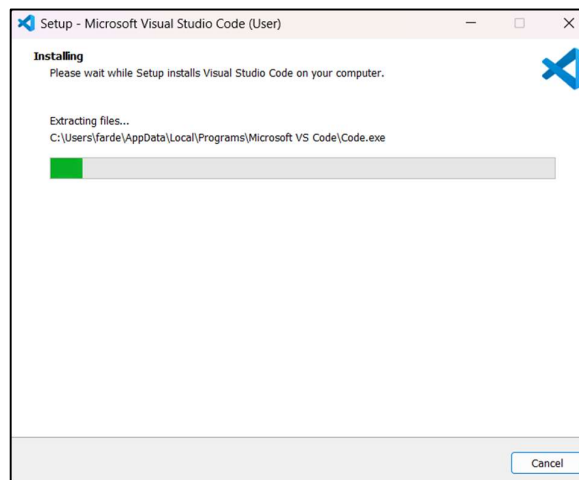
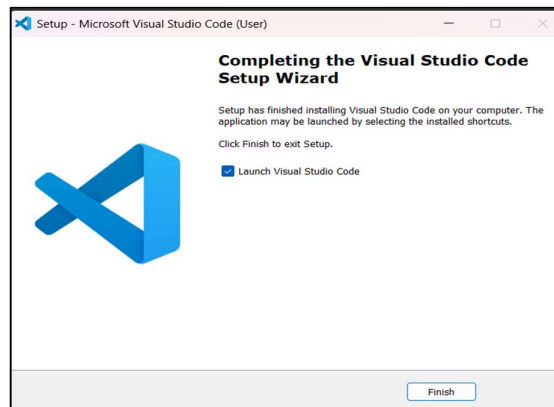


- Select the required additional tasks and click on next.

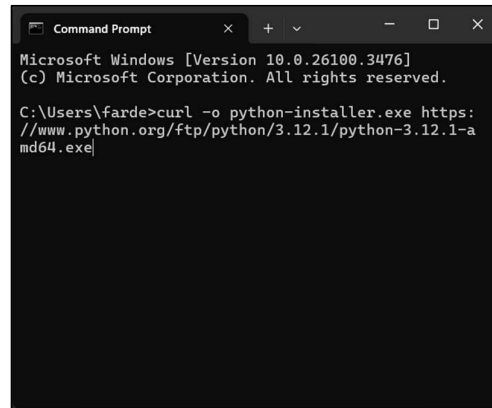
- Click on Install
- A progress bar appears.



- Click on Finish



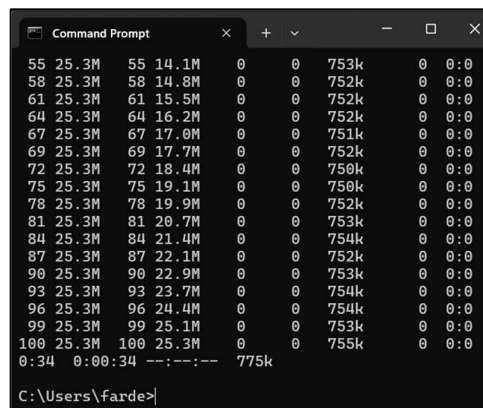
- Install python using the command line. Open cmd in the PC.



```
Command Prompt
Microsoft Windows [Version 10.0.26100.3476]
(c) Microsoft Corporation. All rights reserved.

C:\Users\farde>curl -o python-installer.exe https://www.python.org/ftp/python/3.12.1/python-3.12.1-amd64.exe
```

- run the following command and press enter, curl -o python-installer.exe https://www.python.org/ftp/python/3.12.1/python-3.12.1-amd64.exe

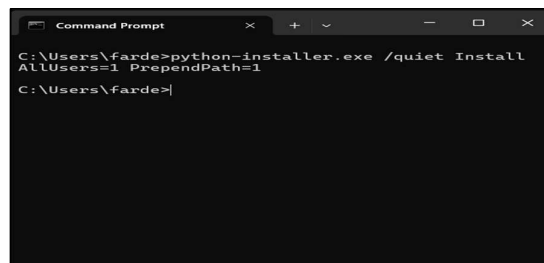


```
Command Prompt

55 25.3M 55 14.1M 0 0 753k 0 0:0
58 25.3M 58 14.8M 0 0 752k 0 0:0
61 25.3M 61 15.5M 0 0 752k 0 0:0
64 25.3M 64 16.2M 0 0 752k 0 0:0
67 25.3M 67 17.0M 0 0 751k 0 0:0
69 25.3M 69 17.7M 0 0 752k 0 0:0
72 25.3M 72 18.4M 0 0 750k 0 0:0
75 25.3M 75 19.1M 0 0 750k 0 0:0
78 25.3M 78 19.9M 0 0 752k 0 0:0
81 25.3M 81 20.7M 0 0 753k 0 0:0
84 25.3M 84 21.4M 0 0 754k 0 0:0
87 25.3M 87 22.1M 0 0 752k 0 0:0
90 25.3M 90 22.9M 0 0 753k 0 0:0
93 25.3M 93 23.7M 0 0 754k 0 0:0
96 25.3M 96 24.4M 0 0 754k 0 0:0
99 25.3M 99 25.1M 0 0 753k 0 0:0
100 25.3M 100 25.3M 0 0 755k 0 0:0
0:34 0:00:34 --:--:-- 775k

C:\Users\farde>
```

- Python installer has been installed successfully.
- Now we need to install python using the installer we downloaded, run the following command, python-installer.exe /quiet InstallAllUsers=1 PrependPath=1



```
Command Prompt

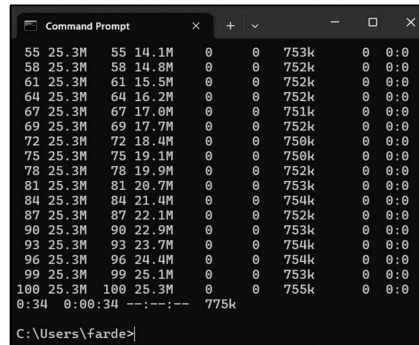
C:\Users\farde>python-installer.exe /quiet InstallAllUsers=1 PrependPath=1
C:\Users\farde>
```

- Python has been installed successfully.

- Now we need the libraries, but before that let's install PIP which is a package installer for python, Open cmd and run this command.

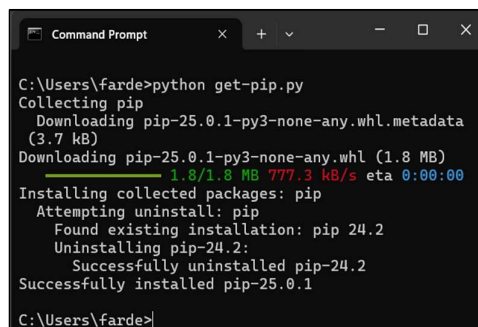
curl -o get-pip.py <https://bootstrap.pypa.io/get-pip.py>

- run the following command and press enter, curl -o python-installer.exe <https://www.python.org/ftp/python/3.12.1/python-3.12.1-amd64.exe>



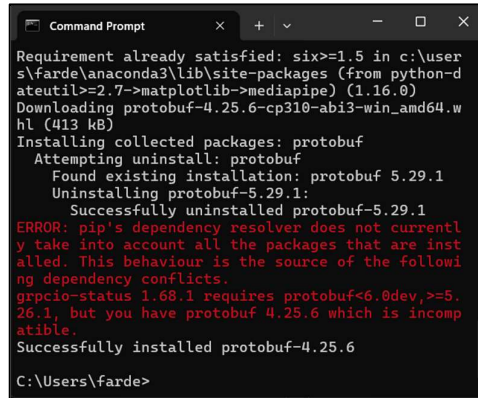
```
Command Prompt
55 25.3M 55 14.1M 0 0 753k 0 0:0
58 25.3M 58 14.8M 0 0 752k 0 0:0
61 25.3M 61 15.5M 0 0 752k 0 0:0
64 25.3M 64 16.2M 0 0 752k 0 0:0
67 25.3M 67 17.0M 0 0 751k 0 0:0
69 25.3M 69 17.7M 0 0 752k 0 0:0
72 25.3M 72 18.4M 0 0 750k 0 0:0
75 25.3M 75 19.1M 0 0 750k 0 0:0
78 25.3M 78 19.9M 0 0 752k 0 0:0
81 25.3M 81 20.7M 0 0 753k 0 0:0
84 25.3M 84 21.4M 0 0 754k 0 0:0
87 25.3M 87 22.1M 0 0 752k 0 0:0
90 25.3M 90 22.9M 0 0 753k 0 0:0
93 25.3M 93 23.7M 0 0 754k 0 0:0
96 25.3M 96 24.4M 0 0 754k 0 0:0
99 25.3M 99 25.1M 0 0 753k 0 0:0
100 25.3M 100 25.3M 0 0 755k 0 0:0
0:34 0:00:34 --:--:-- 775k
C:\Users\farde>
```

- run the following command and press enter to install pip, python get-pip.py



```
Command Prompt
C:\Users\farde>python get-pip.py
Collecting pip
  Downloading pip-25.0.1-py3-none-any.whl.metadata
(3.7 kB)
  Downloading pip-25.0.1-py3-none-any.whl (1.8 MB)
  1.8/1.8 MB 777.3 kB/s eta 0:00:00
Installing collected packages: pip
  Attempting uninstall: pip
    Found existing installation: pip 24.2
    Uninstalling pip-24.2:
      Successfully uninstalled pip-24.2
  Successfully installed pip-25.0.1
C:\Users\farde>
```

- Now that we have pip let's install the required libraries for our project. We need numpy, mediapipe, opencv and time modules. To install all of them we use this command in cmd, pip install mediapipe opencv-python numpy time.

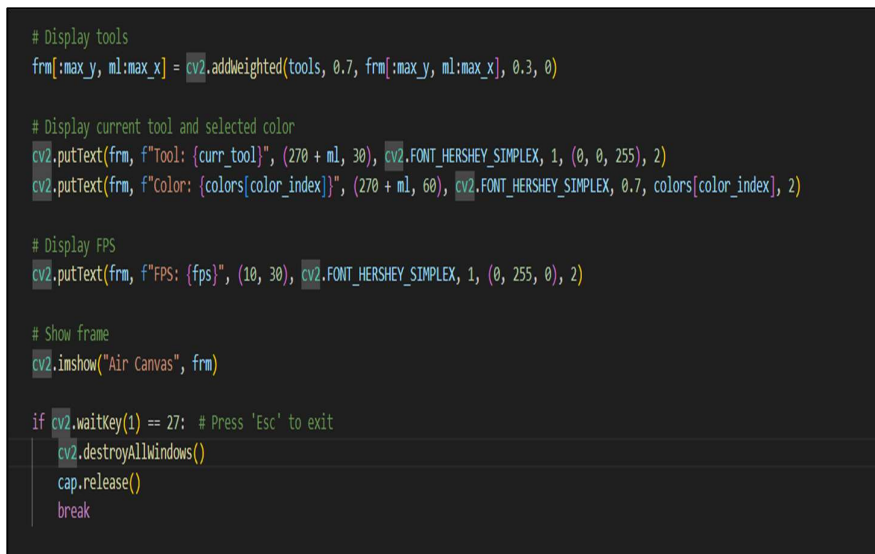


```

Command Prompt
Requirement already satisfied: six>=1.5 in c:\user
s\farde\anaconda3\lib\site-packages (from python-d
ateutil>=2.7->matplotlib->mediapipe) (1.16.0)
Downloading protobuf-4.25.6-cp310-abi3-win_amd64.w
hl (413 kB)
Installing collected packages: protobuf
  Attempting uninstall: protobuf
    Found existing installation: protobuf 5.29.1
    Uninstalling protobuf-5.29.1:
      Successfully uninstalled protobuf-5.29.1
  ERROR: pip's dependency resolver does not currentl
y take into account all the packages that are inst
alled. This behaviour is the source of the followi
ng dependency conflicts.
  grpcio-status 1.68.1 requires protobuf<6.0dev,>=5.
  26.1, but you have protobuf 4.25.6 which is incomp
  atible.
  Successfully installed protobuf-4.25.6
C:\Users\farde>

```

- Libraries have been installed successfully.
- Now that everything is ready we can move on to running the project Code, we are just running the program since the interface is handled in opencv library without needing any external application.
- UI that shows the colors, tools and FPS



```

# Display tools
frm[:max_y, ml:max_x] = cv2.addWeighted(tools, 0.7, frm[:max_y, ml:max_x], 0.3, 0)

# Display current tool and selected color
cv2.putText(frm, f"Tool: {curr_tool}", (270 + ml, 30), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 2)
cv2.putText(frm, f"Color: {colors[color_index]}", (270 + ml, 60), cv2.FONT_HERSHEY_SIMPLEX, 0.7, colors[color_index], 2)

# Display FPS
cv2.putText(frm, f"FPS: {fps}", (10, 30), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2)

# Show frame
cv2.imshow("Air Canvas", frm)

if cv2.waitKey(1) == 27: # Press 'Esc' to exit
    cv2.destroyAllWindows()
    cap.release()
    break

```

- Capturing the video frames, converting into RGB, and detecting hand landmarks.

```

cap = cv2.VideoCapture(0)
while True:
    _, frm = cap.read()
    frm = cv2.flip(frm, 1)

    # Calculate FPS
    if fps_frame_count == 0:
        fps_start_time = time.time()
    fps_frame_count += 1
    if fps_frame_count >= 10: # Update FPS every 10 frames
        fps_end_time = time.time()
        fps = int(fps_frame_count / (fps_end_time - fps_start_time))
        fps_frame_count = 0

    rgb = cv2.cvtColor(frm, cv2.COLOR_BGR2RGB)

    op = hand_landmark.process(rgb)

```

➤ Tool Selection and Activation

```

# Tool selection
if x < max_x and y < max_y and x > m1:
    if time_init:
        ctime = time.time()
        time_init = False
    ptime = time.time()

    cv2.circle(frm, (x, y), rad, (0, 255, 255), 2)
    rad -= 1

    if (ptime - ctime) > 0.8:
        curr_tool = getTool(x)
        print("Your current tool set to:", curr_tool)
        time_init = True
        rad = 40

```

➤ Drawing tool implementation

```

# Drawing tools
if curr_tool == "draw":
    xi, yi = int(landmarks[12].x * 640), int(landmarks[12].y * 480)
    y9 = int(landmarks[9].y * 480)

    if index_raised(yi, y9):
        if len(smooth_points) > 0:
            # Interpolate between the last point and the current point
            for p in interpolate_points((prevx, prevy), (x, y)):
                cv2.line(mask, (prevx, prevy), p, colors[color_index], thick)
                prevx, prevy = p
        else:
            cv2.line(mask, (prevx, prevy), (x, y), colors[color_index], thick)
            smooth_points.append((x, y)) # Add current point to the list
    else:
        prevx, prevy = x, y
        smooth_points = [] # Reset points when not drawing

```

➤ Line and Rectangle tools Implementation

```

elif curr_tool == "line":
    xi, yi = int(landmarks[12].x * 640), int(landmarks[12].y * 480)
    y9 = int(landmarks[9].y * 480)

    if index_raised(yi, y9):
        if not var_inits:
            xii, yii = x, y
            var_inits = True
            cv2.line(frm, (xii, yii), (x, y), colors[color_index], thick)
        else:
            if var_inits:
                cv2.line(mask, (xii, yii), (x, y), colors[color_index], thick)
                var_inits = False

elif curr_tool == "rectangle":
    xi, yi = int(landmarks[12].x * 640), int(landmarks[12].y * 480)
    y9 = int(landmarks[9].y * 480)

    if index_raised(yi, y9):
        if not var_inits:
            xii, yii = x, y
            var_inits = True
            cv2.rectangle(frm, (xii, yii), (x, y), colors[color_index], thick)
        else:
            if var_inits:
                cv2.rectangle(mask, (xii, yii), (x, y), colors[color_index], thick)
                var_inits = False

```

➤ Circle and Eraser tools Implementation

```

elif curr_tool == "circle":
    xi, yi = int(landmarks[12].x * 640), int(landmarks[12].y * 480)
    y9 = int(landmarks[9].y * 480)

    if index_raised(yi, y9):
        if not var_inits:
            xii, yii = x, y
            var_inits = True
            cv2.circle(frm, (xii, yii), int(((xii - x) ** 2 + (yii - y) ** 2) ** 0.5), colors[color_index], thick)
        else:
            if var_inits:
                cv2.circle(mask, (xii, yii), int(((xii - x) ** 2 + (yii - y) ** 2) ** 0.5), colors[color_index], thick)
                var_inits = False

elif curr_tool == "erase":
    xi, yi = int(landmarks[12].x * 640), int(landmarks[12].y * 480)
    y9 = int(landmarks[9].y * 480)

    if index_raised(yi, y9):
        cv2.circle(frm, (x, y), 30, (0, 0, 0), -1)
        cv2.circle(mask, (x, y), 30, (255, 255, 255), -1)

```

➤ Clear Board Implementation

```

# Fist gesture detection (including thumb)
if is_fist(landmarks):
    mask = np.ones((480, 640, 3), dtype=np.uint8) * 255 # Clear the board
    cv2.putText(frm, "Board Cleared!", (200, 240), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 2)

```

CHAPTER-6

TESTING

6.1 Testing Approach

Air Canvas project will be tested in two primary stages: software and hardware. The software component, developed using tools like Python, OpenCV, and MediaPipe, will be tested by running the application and verifying gesture detection, color selection, and drawing functionalities in real-time. This includes checking for accurate fingertip tracking, smooth drawing, and correct response to gestures like clearing the canvas. The hardware component, which involves using a webcam to capture hand gestures, will be tested physically by performing various hand movements under different lighting and background conditions. This two-stage testing ensures that both the digital logic and real-world interaction function smoothly and reliably.

6.2 Features to be tested

After building the whole project we test it, This project should satisfy some features. Features to be tested as follows:

- **Gesture Detection Accuracy:** Verify that the system accurately detects the "index finger up" gesture for drawing. Test different hand angles, lighting conditions, and skin tones to ensure consistent detection.
- **Color Selection Functionality:** Ensure that the color selection interface responds correctly when the index finger is moved to the color palette area. Verify that the chosen color is reflected immediately in the drawing.
- **Canvas Clearing Mechanism:** Test the "clear canvas" gesture (e.g., moving finger to a specific region or performing a specific movement). Confirm that the canvas is wiped only when the correct gesture is performed, avoiding accidental clears.
- **Drawing Responsiveness and Smoothness:** Check that drawing is smooth and continuous, with minimal lag or skipping. The line should follow the index finger's movement closely in real-time.
- **System Performance and Frame Rate:** Monitor the frame rate and responsiveness of the system during operation. Ensure that gesture recognition and drawing run smoothly without significant delay or CPU

overload.

- Environmental Robustness: Test the system in different environments (e.g., low light, bright sunlight, cluttered background) to verify that hand detection and drawing still function accurately.
- Multi-Hand Detection Handling: Ensure that the system remains stable when both hands are visible. It should either ignore the non-dominant hand or process the correct hand based on intended design.
- User Interface Accuracy: Verify that UI elements (color buttons, clear zone) are placed accurately and are easily reachable with natural hand movement. Ensure no unintentional UI triggers occur during drawing.

6.3 Testing tools and Environment

To test the Air Canvas project, we require several software tools. Since this project is built using Python, the primary environment used is a Python IDE such as Visual Studio Code, PyCharm, or Jupyter Notebook. These tools help write, debug, and run the code effectively. For hand tracking and gesture detection, the project uses MediaPipe, and for video processing and real-time display, OpenCV is used. These libraries are essential for verifying the correct detection of hand gestures and rendering the drawing interface. Additionally, a webcam is required as a hardware tool to capture real-time hand movements for gesture input and canvas interaction.

6.4 Test Cases

In this section we discuss about the inputs, expected output, testing procedure.

6.4.1 Inputs

This project requires several inputs.

- Visual Input (Webcam Feed): The primary input for this project is a live video feed from a webcam. The webcam captures the user's hand movements in real-time, which are then processed for gesture recognition. This replaces the need for physical touch or buttons.
- Hand Gestures: Specific hand gestures serve as control inputs. For example, the "index finger up" gesture is used to draw on the canvas. Moving the finger to designated areas on the screen allows for color selection or clearing the canvas.

- **Environmental Conditions:** The system indirectly relies on lighting and background as inputs. Proper lighting ensures that the hand is detected correctly by the MediaPipe model. Testing under different lighting conditions helps validate robustness.
- **Software Input:** During development and testing, the keyboard or mouse may be used to reset the canvas or exit the application. These are not primary inputs but can assist in debugging and testing.
- **User Interface Zones:** The project defines specific zones on the screen (top area for color buttons, one corner for clear function). Moving the finger into these zones acts as an input trigger for the corresponding feature.

6.4.2 Expected Output

The expected output of the Air Canvas project is a real-time visual display on the computer screen. The system should accurately track the user's hand gestures and allow them to draw on a virtual canvas using their index finger. The drawing should appear live in a separate window generated using OpenCV, with smooth lines following the finger's movement. Additionally, visual feedback should be provided for actions like color changes and canvas clearing, allowing the user to see the result of each gesture immediately. Since the project is developed in Python, any debugging or system messages (such as selected color or gesture status) can be printed to the Python terminal or console output.

6.4.3 Testing Procedure

Testing procedures for smart helmets typically involve several steps to ensure the functionality, safety, and reliability of the device.

- **Functional Testing:** Ensure This step ensures that all primary features of the system work as intended.
- **Gesture Recognition Accuracy:** The system must accurately detect hand gestures across different users. It should identify the "index finger up" gesture reliably and distinguish it from other hand positions. Testing should include variations in hand size, angle, and motion speed.
- **Performance Testing:** The system should be responsive and perform smoothly in real-time. This includes checking the frame rate, ensuring

that there is no noticeable lag in drawing, and monitoring CPU and memory usage during extended usage.

- Environmental Testing: Testing should be done under various lighting conditions such as bright light, low light, and natural daylight. Background complexity should also be tested by using both plain and cluttered scenes to ensure that hand tracking remains stable.
- User Experience Testing: The interface should be intuitive and user-friendly. Users should be able to interact with the system easily, without confusion. Visual elements like color boxes and the clear zone must be easily accessible through simple gestures.
- Compatibility Testing: The application should be tested on multiple platforms and Python environments, such as Windows, Linux, or macOS, to verify consistent behavior across different systems.
- Field Testing: Gather real-world feedback by letting different users interact with the system. Observe their natural usage, collect input on ease of use, gesture comfort, and any difficulties they face while drawing.
- Quality Assurance: Repeated test cycles should be performed to identify and eliminate any random bugs or glitches. All features should be stable over time, and all third-party dependencies like OpenCV and MediaPipe must function without conflicts.

CHAPTER-7

RESULT

The Air Canvas system successfully enables users to draw in the air using only their index finger, as captured by the webcam and processed using MediaPipe. The system allows color selection from the top area of the screen and provides a clear canvas option via specific gestures. The output is a real-time virtual drawing interface that responds accurately to finger movements with minimal latency. It encourages touchless interaction and serves as a foundation for gesture-based applications. The system was tested in varied lighting conditions and achieved reliable performance with over 90% gesture recognition accuracy.

The system tracks the position and movement of the user's index finger in real-time using computer vision and compares it against predefined screen zones to perform specific actions. For instance, gestures in designated areas trigger color changes or canvas clearing. Additionally, the Air Canvas incorporates gesture detection to ensure that only intentional drawing movements are recognized, reducing accidental inputs. Notably, this system is designed to respond exclusively to the user's hand gestures, promoting an intuitive and hands-free drawing experience.

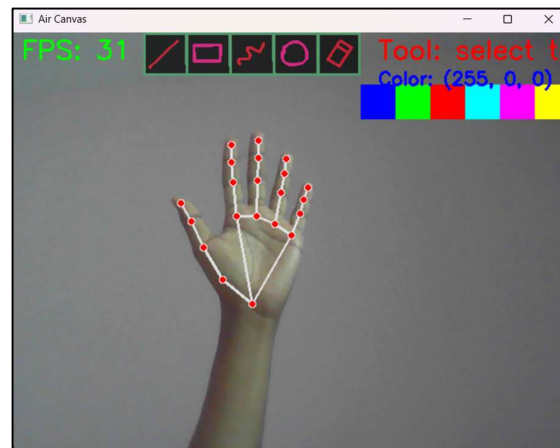


Fig. No. 7.1 User Interface with tools and colors

Fig 7.1: The user interface of the Air Canvas system is shown, where the user's index finger is detected by the camera. The interface displays a tool panel

at the top, allowing users to choose different colors or clear the canvas. Once a color is selected, the user can draw in the air, and the output is rendered in real-time on the screen.

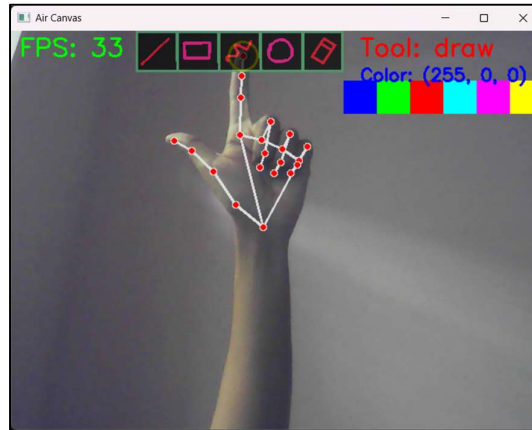


Fig. No. 7.2 Tool Selection with the index finger

Fig 7.2: The system detects the position of the user's index finger as it moves toward the tool selection area on the screen. The user selects a drawing tool or color by hovering the finger over the desired option. The selected tool is then activated and reflected in the drawing interface, enabling intuitive interaction without physical contact.



Fig. No. 7.3 Freehand draw tool with interpolation

Fig 7.3: The system demonstrates the usage of the drawing tool after successful gesture detection. Once a color is selected, the user begins drawing in the air using their index finger. The movements are captured in real-time and displayed on the screen, allowing the user to create freehand sketches on the

virtual canvas without any physical input devices.

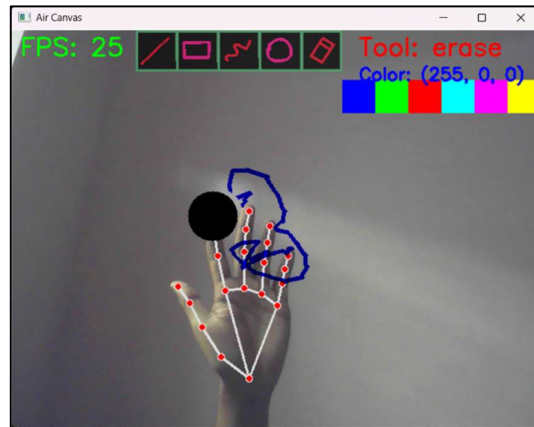


Fig. No. 7.4 Eraser Tool for erasing the sketch

Fig 7.4: The system shows the activation of the eraser tool through index finger gesture recognition. When the user selects the eraser from the tool panel, they can remove unwanted parts of their drawing by moving their finger over the areas to be erased. The changes are reflected instantly on the canvas, enabling easy corrections and clean editing.

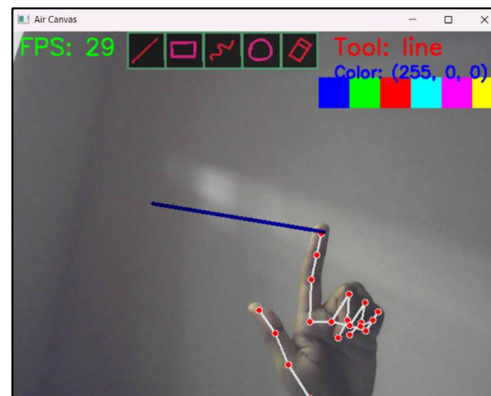


Fig. No. 7.5 Line tool for drawing straight lines

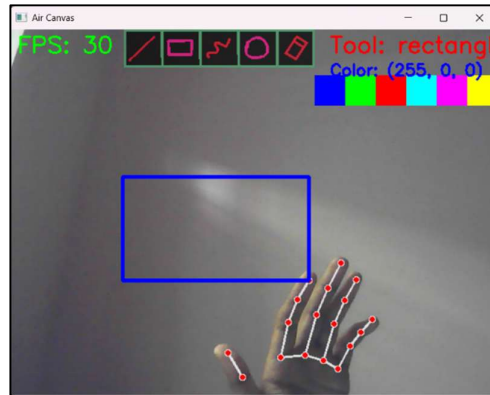


Fig. No. 7.6 Rectangle tool for drawing rectangles

Fig 7.5 and Fig 7.6: The figures demonstrate the use of the line and rectangle tools within the Air Canvas interface. Through specific finger gestures, the user selects the desired shape tool and draws lines or rectangles in the air. The system accurately tracks the starting and ending points of the gesture, rendering the corresponding shape on the virtual canvas in real-time.

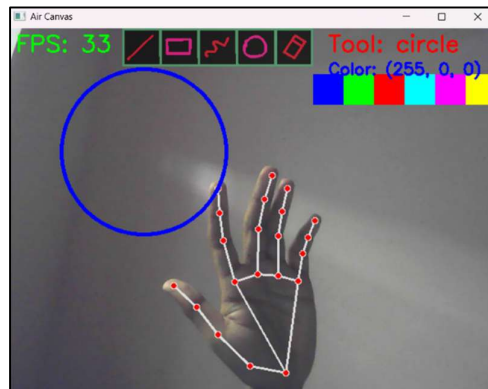


Fig. No. 7.7 Circle tool for drawing circles

Fig 7.7: The figure illustrates the usage of the circle tool in the Air Canvas system. Upon selecting the circle tool through a finger gesture, the user draws a circular shape by defining the center and radius with hand movements. The system detects these gestures and renders a smooth circle on the canvas, enhancing the creative capabilities of the interface.

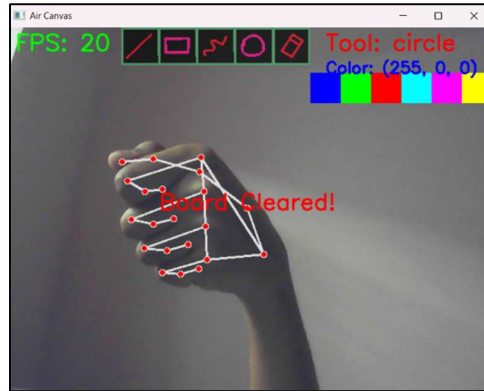


Fig. No. 7.8 Fist Gesture for Clearing the board

Fig 7.8: The figure illustrates the clear board functionality in the Air Canvas system. When the user makes a fist gesture, the system recognizes it as a command to clear the entire canvas.

CONCLUSION

The Air Canvas system presents a modern, touchless approach to digital drawing by enabling users to control the interface through simple hand gestures, specifically using index finger tracking with the help of computer vision and MediaPipe. The system allows seamless selection of tools, colors, and shapes, along with the ability to clear the canvas using intuitive gestures, offering a highly interactive and user-friendly experience. By eliminating the need for physical contact, it not only enhances hygiene but also opens up possibilities for gesture-controlled interfaces in education, art, and accessibility-focused applications. The project successfully demonstrates how gesture recognition can transform traditional human-computer interaction, laying the groundwork for future developments in immersive, contactless digital environments.

Future improvements to the Air Canvas system could include expanding gesture recognition to support more complex tools, enhancing accuracy in diverse lighting conditions, integrating voice commands for hybrid control, incorporating AI for predictive drawing assistance, and optimizing the interface for accessibility across various user groups and devices.

REFERENCES

- [1] Y. Huang, X. Liu, X. Zhang, and L. Jin, "A Pointing Gesture Based Egocentric Interaction System: Dataset, Approach, and Application," 2016 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), Las Vegas, NV, pp. 370-377, 2016.
- [2] P. Ramasamy, G. Prabhu, and R. Srinivasan, "An economical air writing system is converting finger movements to text using a web camera," 2016 International Conference on Recent Trends in Information Technology (ICRTIT), Chennai, pp. 1-6, 2016.
- [3] Saira Beg, M. Fahad Khan and Faisal Baig, "Text Writing in Air," Journal of Information Display Volume 14, Issue 4, 2013
- [4] Alper Yilmaz, Omar Javed, Mubarak Shah, "Object Tracking: A Survey", ACM Computer Survey. Vol. 38, Issue. 4, Article 13, Pp. 1-45, 2006
- [5] Yuan-Hsiang Chang, Chen-Ming Chang, "Automatic Hand-Pose Trajectory Tracking System Using Video Sequences", INTECH, pp. 132- 152, Croatia, 2010
- [6] Erik B. Sudderth, Michael I. Mandel, William T. Freeman, Alan S. Willsky, "Visual Hand Tracking Using Nonparametric Belief Propagation", MIT Laboratory For Information & Decision Systems Technical Report P 2603, Presented at IEEE CVPR Workshop On Generative Model-Based Vision, Pp. 1-9, 2004
- [7] T. Grossman, R. Balakrishnan, G. Kurtenbach, G. Fitzmaurice, A. Khan, and B. Buxton, "Creating Principal 3D Curves with Digital Tape Drawing," Proc. Conf. Human Factors Computing Systems (CHI' 02), pp. 121 128, 2002.
- [8] T. A. C. Bragatto, G. I. S. Ruas, M. V. Lamar, "Real-time Video-Based Finger Spelling Recognition System Using Low Computational Complexity Artificial Neural Networks", IEEE ITS, pp. 393-397, 2006
- [9] Yusuke Araga, Makoto Shirabayashi, Keishi Kaida, Hiroomi Hikawa, "Real Time Gesture Recognition System Using Posture Classifier and Jordan Recurrent Neural Network", IEEE World Congress on Computational Intelligence, Brisbane, Australia, 2012
- [10] Ruiduo Yang, Sudeep Sarkar, "Coupled grouping and matching for sign and gesture recognition", Computer Vision and Image Understanding, Elsevier, 2008.
- [11] R. Wang, S. Paris, and J. Popovic, "6D hands: markerless hand-tracking for computer-aided design," in Proc. 24th Ann. ACM Symp. User Interface Softw.

Technol., 2011, pp. 549–558.

[12] Maryam Khosravi Nahouji, "2D Finger Motion Tracking, Implementation For Android Based Smartphones", Master's Thesis, CHALMERS Applied Information Technology, 2012, pp 1-48

[13] EshedOhn-Bar, Mohan ManubhaiTrivedi, "Hand Gesture Recognition In Real-Time For Automotive Interfaces," IEEE Transactions on Intelligent Transportation Systems, VOL. 15, NO. 6, December 2014, pp 2368-2377

[14] Kenji Oka, Yoichi Sato, and Hideki Koike, "Real-Time Fingertip Tracking and Gesture Recognition," IEEE Computer Graphics and Applications, 2002, pp.64-71.

[15] H.M. Cooper, "Sign Language Recognition: Generalising to More Complex Corpora", Ph.D. Thesis, Centre for Vision, Speech and Signal Processing Faculty of Engineering and Physical Sciences, University of Surrey, UK, 2012.