International Journal for Multidisciplinary Research (IJFMR)
E-ISSN: 2582-2160 ● Website: www.ijfmr.com ● Email: editor@ijfmr.com

E-Learning for Fingertip Canvas

Ms. Sanika Prasad Naik[1], Ms. Anuja Navnath Shinde[2], Ms. Shruti Satish Sonakul[3], Mr. Sahil Sachin Savale[4], Mr. Atharav Dynaneshwar Ithape[5], Prof. Pradnya More[6]

[1,2,3,4,5]Student, Artificial Intelligence and Data Science, Anantrao Pawar College of Engineering and Research
[6]Student, Artificial Intelligence and Data Science, Anantrao Pawar College of Engineering and Research

Abstract:
Develop a finger-mounted bead system with gesture recognition for creating digital art. The bead will track movement and pressure, translating gestures into brushstrokes, sculpting actions, and color selection on a digital canvas. The project will focus on creating an intuitive user experience for artists, with features like real-time rendering and customizable brush settings. Challenges to address include accurate gesture recognition, pressure sensitivity calibration, and user interface design for ease of use. This project has the potential to revolutionize digital art creation by offering a more natural and expressive way for artists to interact with their tools[2] . This project aims to develop a novel system for creating digital art using a finger-mounted bead and intuitive gesture recognition. The bead will be equipped with sensors to track its movement and pressure in 3D space. These inputs will be translated into various artistic actions within a dedicated software application.

Introduction:
Develop a finger-mounted bead system with gesture recognition for creating digital art. The bead will track movement and pressure, translating gestures into brushstrokes, sculpting actions, and color selection on a digital canvas. The project will focus on creating an intuitive user experience for artists, with features like real-time rendering and customizable brush settings. Challenges to address include accurate gesture recognition, pressure sensitivity calibration, and user interface design for ease of use. This project has the potential to revolutionize digital art creation by offering a more natural and expressive way for artists to interact with their tools. This project aims to develop a novel system for creating digital art using a finger-mounted bead and intuitive gesture recognition. The bead will be equipped with sensors to track its movement and pressure in 3D space. These inputs will be translated into various artistic actions within a dedicated software application

Proposed System: *I.*
Overview of the System:
1. System Overview
The proposed system is a **real-time virtual painting application** that enables users to draw on a digital canvas using hand gestures, eliminating the need for physical input devices such as a mouse or stylus. It utilizes a standard webcam for input, **MediaPipe** for hand detection, and **OpenCV** for real-time drawing and interface rendering.

2. System Components

Component	Description
Webcam	Captures real-time video feed of the user's hand.
MediaPipe Hands	Detects hand landmarks (e.g., index fingertip, thumb tip).
OpenCV	Handles video display, drawing canvas, and UI elements (buttons).
Python Program	Processes input frames, recognizes gestures, and updates the drawing canvas.

3. System Workflow

- **Input Stage**
- Capture video frame from webcam.
- Flip the frame horizontally (mirror effect).

Hand Detection

- Use **MediaPipe Hands** to detect and track 21 key landmarks on a single hand.
- Identify index fingertip (landmark 8) as drawing tool.
- Use thumb tip (landmark 4) for gesture control.

Gesture Interpretation

- **Drawing Mode**: When index finger is extended and thumb is away.
- **Pen Up**: When thumb and index finger are close.
- **UI Interaction**: When index finger touches the top bar (used to change color or clear canvas).

Drawing and Canvas Update

- Draw lines on both live camera feed and a persistent canvas.
- Update drawing color based on selection.

Output Stage

- Display two windows:
  - Output: Camera feed with drawing overlay.
  - Paint: Clean drawing canvas.

4. Key Features

- Draw using natural hand gestures.
- Choose between multiple colors.
- Clear the canvas with a gesture.
- Smoothing using exponential moving average for better pointer control.
- No need for touchscreen or physical tools.

5. Applications

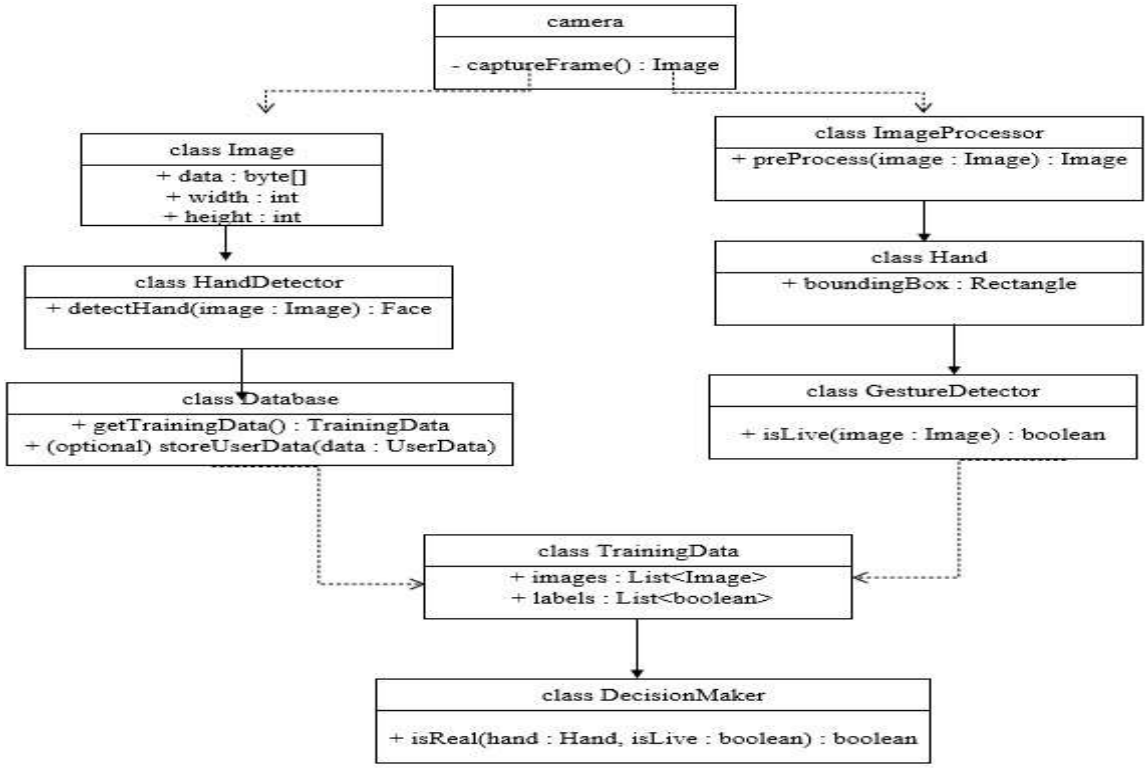
- **Educational Tools**: Virtual whiteboards for teaching.
- **Creative Apps**: Freehand digital art without stylus.
- **Rehabilitation**: Fine motor skills training.
- **Touchless UI**: In public systems or COVID-safe interfaces.

6. Advantages

- **Touchless Interaction** – No hardware other than a webcam.
- **Intuitive Use** – Natural gestures control the system.
- **Low Cost** – No need for special sensors or controllers.
- **Platform Independent** – Can run on any OS with Python and camera.

7. Future Enhancements

- Add **undo** and **save** options.
- Add **multi-finger gestures** for advanced tools.
- Integrate **voice commands** or **gesture menus**.
- Support **multi-user interaction**.
- Port to **mobile or web platforms** using OpenCV.js or Kivy.



III. Implementation Details:

1. Programming Language and Libraries

Tool	Purpose
Python	Core programming language.
OpenCV (cv2)	Handles webcam input, image processing, GUI, and drawing.
MediaPipe	Provides hand tracking and gesture recognition.
NumPy	Supports array and matrix operations for image/frame handling.

Tool	Purpose
Deque (from collections)	An efficient data structure for storing drawing paths.

2. System Architecture

a. Input Module

- Captures video from webcam using cv2.VideoCapture(0).
- Flips the frame horizontally for a mirror view.

b. Hand Detection

- Uses mediapipe.solutions.hands.Hands() to detect one hand.
- Tracks 21 landmarks per frame (focus on index finger tip landmark and thumb landmark

c. Gesture Recognition Logic

- **Drawing Gesture**: When the index finger is up and the thumb is far apart.
- **Stop Drawing (Pen Up)**: When thumb comes close to index finger (distance < 30 px).
- **Button Press Detection**: If the index finger is within the y-coordinate of the button area (y <= 65), then check the x-position for:
  - Clear
  - Color selection (Blue, Green, Red, Yellow)

d. Drawing Logic

- Each color has a separate deque list.
- When drawing is active, append current index finger position to the selected color deque.
- When stopped or hand is lost, start a new deque (new stroke).

e. Smoothing Algorithm

- Applies **Exponential Moving Average** (EMA) to index finger positions:

```
python
CopyEdit
filtered_center = (
int(alpha * prev_x + (1 - alpha) * curr_x),
int(alpha * prev_y + (1 - alpha) * curr_y)
)
```

- Prevents jittery or shaky lines.

f. Canvas and UI

- A blank white image (paintWindow) is initialized as the canvas.
- Top part is a UI panel with buttons (cv2.rectangle) and labels (cv2.putText).
- Drawings appear on both the live feed and the canvas window.

g. Display & Exit

- Displays two windows:
  - "Output": Webcam feed with drawing overlay.
  - "Paint": Clean drawing canvas only.
- Press 'q' to quit.

3. Data Structures Used

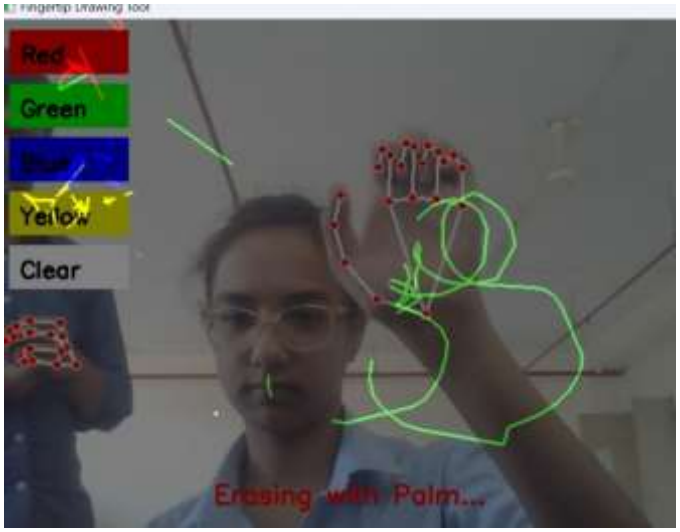
Name	Type	Purpose
bpoints, gpoints, rpoints, ypoints	List of deques	Stores the strokes for each color
filtered_center	Tuple	Smoothed coordinates of fingertip
colorIndex	Integer	Tracks current color selected

4. Control Flow (Main Loop)

1. Read and flip webcam frame.
2. Convert to RGB and pass through MediaPipe.
3. If hand detected:
   - Extract fingertip and thumb positions.
   - Smooth the fingertip location.
   - If in button area:
     - Change color or clear screen.
   - Else:
     - Append fingertip to active color list.
4. Else (no hand detected):
   - Start new deques to separate strokes.
5. Draw all strokes on both canvas and webcam feed.
6. Show the frames.

5. Code Efficiency

- **Real-time performance**: Efficient due to lightweight models in MediaPipe and minimal frame processing.
- **Optimized drawing**: Only draws non-null consecutive points.
- **Memory usage**: Limited history using deque(maxlen=512).



Future Scope

Accessibility and Integration

- **Universal Design**
  - Adapt finger-mounted bead system for various hand sizes and physical disabilities.
  - Make digital art creation more inclusive and accessible.
- **Virtual Reality (VR) Integration**
  - Use the bead system in VR environments for immersive art creation.
  - Enable users to paint or sculpt in 3D on virtual canvases or objects.
- **Cross-Platform Compatibility**
  - Ensure compatibility with various digital art tools (e.g., Photoshop, Blender).
  - Allow integration across operating systems and hardware platforms.

Beyond Art: New Applications

- **3D Sculpting and Design**
  - Track 3D finger movements to sculpt and manipulate virtual objects intuitively.
  - Replace traditional tools in CAD or modeling software with natural gestures.
- **AR Design and Prototyping**
  - Enable painting or annotation directly onto real-world surfaces in AR.
  - Useful for rapid prototyping, interior design, and interactive installations.

Performance Optimization

- **Real-time Processing**
  - Improve hand and gesture detection algorithms for faster frame processing.
  - Ensure lag-free interaction for better user experience.
- **Resource Efficiency**
  - Reduce CPU/GPU usage to run efficiently on low-end or mobile devices.Use lightweight models or quantization techniques.

Scalability and Deployment

- **Cloud-based Deployment**
  - Deploy gesture recognition backend on cloud infrastructure (e.g., AWS, Azure).

- Handle large user traffic and enable collaboration or shared canvases.
- **Mobile Integration**
- Integrate the system into mobile "air canvas" apps.
- Enable gesture-based drawing and verification features on-the-go.

Conclusion

This project illustrates the practical feasibility of integrating **gesture-based detection** with an **Air Canvas** system to enable intuitive and secure human-computer interaction. The implementation, built using **Python**, **OpenCV**, and development environments like **VS Code** and **Spyder**, showcases the fundamental capabilities of **real-time hand tracking** and **basic gesture recognition** using MediaPipe's hand landmark detection.

At its core, the system allows users to interact with a virtual canvas through simple hand movements, enabling actions such as drawing, changing colors, and clearing the canvas—without the need for any physical tools. The use of fingertip and thumb landmarks provides a gesture-based interface that responds dynamically to the user's hand motions. This real-time processing enhances the user experience, making it both engaging and efficient.

From a security and authentication standpoint, integrating gesture recognition offers potential applications beyond digital art. For example, specific hand gestures could be used for secure login or to authorize tasks in environments where traditional input devices are impractical.

Furthermore, this project lays a solid foundation for future enhancements:

- **Advanced Gesture Recognition**: Incorporating machine learning models to detect a wider range of gestures.
- **Canvas Enhancements**: Adding layers, undo functionality, or shape detection for a more complete drawing experience.
- **User Interface Improvements**: Designing a visually appealing and accessible interface to accommodate users of different ages and abilities.
- **Platform Expansion**: Making the system cross-platform and mobile-friendly for broader reach.

In conclusion, this system demonstrates not only a creative application of computer vision and gesture recognition but also highlights the potential for such interfaces to contribute to **educational technologies**, **art-based learning**, and **interactive design tools**. With further development, it could evolve into a scalable and inclusive solution for both creative and functional digital interaction.

Literature Survey:

1. Mr. Prabakaran R et al. (JETIR, Feb 2024) Focuses on object tracking for air-based writing using OpenCV. Highlights the importance of object identification, tracking, and behavior analysis in improving human-machine interaction through air gestures.
2. Mahati Khandekar & Vandana Yadav (IJCRT, May 2024) Describes a gesture-recognition-based Air Canvas for digital art. Features include real-time drawing, brush customization, and webcam compatibility. Emphasizes accessibility and creative freedom.
3. Aniket Sandbhor et al. (IJCRT, Apr 2023) Explores the evolution of drawing to modern air-drawing using Python, OpenCV, and MediaPipe. Focuses on touch-free interaction using hand motion for drawing on a virtual canvas.

4. Harshit Rajput et al. (IJCRT, May 2023) Presents a stylus-based Air Canvas system using OpenCV for object detection and real-time tracking. Discusses hardware/software design and demonstrates efficient stylus tracking for virtual drawing.
5. Aadesh et al. (IJRTI, July 2022) Implements alphabet detection via Air Canvas using Raspberry Pi, PiCamera, and OpenCV. Allows users to control brush size/color and draw letters in air with hand-tracking on PiTFT screen.
6. Piyush Garg et al. (IJIRT, July 2022) Proposes a gesture-to-text system for writing in air using Python and computer vision. Supports wearable devices and assists in communication, especially for the deaf, by translating finger motion into text.