Smart Sketch: A Virtual Drawing System using Hand Gestures with OpenCV and Mediapipe

Parth Nimbalkar[1] Sanchita Pandit[2] Shweta Kudale[3] Prof. R. K. Taware[4]
[1,2,3]Student [4]Professor
[1,2,3,4]Department of Computer Science and Engineering
[1,2,3,4]SVPM's College of Engineering, Malegaon(Bk), Maharashtra, India

Abstract — In the age of contactless human-computer interaction, Smart Sketch introduces a real-time, gesture-controlled virtual drawing application that enables users to draw, select colors, erase, and change backgrounds using only hand gestures. Built using OpenCV and Mediapipe, the system captures hand landmarks via a webcam and processes them to perform intuitive drawing actions. The application features a user-friendly GUI-based login system developed with Tkinter to ensure secure access. Drawing is controlled through the relative distance between the thumb and index finger, enabling pinch-based activation and deactivation. Color and tool selection are managed by positioning fingers within pre-defined screen zones. Additionally, swipe gestures allow background switching, enriching the user experience. The system overlays drawings on customizable backgrounds in real-time and is optimized for performance across common hardware setups. Smart Sketch demonstrates how existing computer vision technologies can be adapted to create an engaging, touchless sketching interface for educational, creative, and accessibility-focused environments.

Keywords: Hand Gesture Recognition, Virtual Drawing, OpenCV, Mediapipe, Human-Computer Interaction, Touchless Interface, Real-time Processing, Gesture Control

I. Introduction

The evolution of human-computer interaction (HCI) has witnessed a significant transformation with the emergence of touchless and gesture-based interfaces. These systems are becoming increasingly relevant in today's context where hygiene, accessibility, and user convenience are paramount. From smart TVs to AR/VR interfaces and interactive kiosks, gesture recognition is reshaping how users engage with digital devices. Within this space, Smart Sketch introduces a novel solution: a real-time virtual drawing application that enables users to sketch, erase, and interact with a digital canvas using only their hands—without the need for any physical input devices like a mouse, stylus, or touchscreen. Built using powerful and efficient computer vision libraries such as OpenCV and Mediapipe, Smart Sketch leverages real-time video capture and high-accuracy hand landmark detection to track finger gestures and translate them into drawing commands. The system identifies the distance between the index finger and thumb to determine drawing mode and uses finger positions to activate color selections or tool changes based on defined screen regions. An additional feature allows users to switch background images through simple swipe gestures. The integration of a GUI-based login system ensures user-specific access control, enhancing security and personalization. Designed for simplicity, responsiveness, and cross-environment use, Smart Sketch operates efficiently on basic hardware configurations, making it accessible to students, educators, digital artists, and accessibility advocates. It also presents a meaningful step toward inclusive and creative technology by replacing traditional input tools with natural hand movements. The project not only highlights the potential of vision-based applications in creative expression but also provides a foundation for future expansions into multi-modal interaction, shape recognition, voice control, and collaborative environments. As such, Smart Sketch stands as a forward-thinking response to the growing demand for intuitive, hygienic, and engaging digital interaction systems.

II. Literature Survey

Gesture-based interfaces have garnered significant attention in recent years due to their ability to facilitate natural human-device interaction without physical contact. Zhang et al. (2022) developed an air canvas application using color-based fingertip detection with OpenCV, but their system struggled under variable lighting conditions and required users to wear colored gloves, limiting real-world applicability. Sharma (2021) utilized Mediapipe's hand landmark detection to build a mobile touchless drawing app, showcasing robust tracking but lacking features like background switching and desktop GUI integration. Wang (2020) introduced gesture controls for smart TVs focusing on discrete commands rather than continuous input like drawing. Kumar et al. (2023) compared various fingertip tracking methods and identified Mediapipe as offering superior accuracy and efficiency with low computational cost, making it ideal for real-time applications. Google's open-source Mediapipe library provides extensive tools for hand tracking, enabling projects such as virtual mice, sign language recognition, and drawing systems. Patel and Verma (2023) implemented a virtual mouse controlled by hand gestures, reinforcing the feasibility of hand tracking for fine motor control tasks, though without supporting creative drawing. Other research has explored gesture-based interfaces for AR/VR and assistive technologies, underlining the growing importance of touchless controls in various domains. These prior works inspire Smart Sketch's integration of robust hand tracking with an intuitive, feature-rich drawing interface optimized for performance and usability.

III. Methodology

Smart Sketch is developed as a modular system with the following components:

A. Login System:

1) GUI Implementation: The login screen is built using Python's Tkinter library, providing a simple but effective graphical interface for user authentication. This GUI includes fields for username and password input, with labels and buttons styled for clarity.

2) Authentication Logic: The system verifies user credentials against a hardcoded pair (username: "admin", password: "1234"). Upon correct input, the login window closes and the drawing app initializes. Incorrect attempts prompt an error popup, enhancing security and preventing unauthorized access.

B. Video Capture and Preprocessing

1) Webcam Integration: OpenCV accesses the user's webcam, capturing live video frames at 640x480 resolution, balancing clarity and processing speed.
2) Frame Adjustment: Frames are flipped horizontally to create a mirror effect, ensuring intuitive interaction where users' movements match screen actions.

C. Hand Landmark Detection:

1) Mediapipe Hand Solution: Mediapipe detects one hand per frame, identifying 21 landmark points corresponding to finger joints and tips.
2) Coordinate Normalization: Landmarks are provided as normalized values (0 to 1), which are scaled to the frame's pixel dimensions for accurate overlay and gesture calculation.
3) Key Landmarks: Index fingertip (landmark 8) and thumb tip (landmark 4) are central to gesture detection, as their relative distance determines drawing activation.

D. Gesture Recognition and Drawing Logic:

1) Distance Calculation: Using Euclidean distance formula, the system calculates the distance between the index fingertip and thumb tip each frame.
2) Drawing Activation: When the distance exceeds a threshold (40 pixels), the system switches to drawing mode, drawing lines between previous and current index fingertip positions on an invisible canvas layer.
3) Drawing Deactivation: Bringing the fingers close (distance below threshold) stops drawing, allowing breaks and new strokes.
4) Smooth Lines: Continuous tracking ensures smooth and connected lines, simulating natural drawing.

E. Tool and Color Selection

1) Interactive Zones: The bottom part of the screen is segmented into rectangular areas mapped to brush colors (red, green, blue) and eraser mode.
2) Selection Detection: When the index fingertip is inside any zone and within a certain vertical range, the system changes the current brush color or toggles the eraser.
3) Cooldown Mechanism: A 1.5-second cooldown timer prevents accidental multiple switches, improving user control.

F. Background Switching

1) Image Loading: Multiple background images are loaded from a local folder and resized to match the frame size.
2) Swipe Detection: Hand position near the screen edges triggers background switching — left edge for previous, right edge for next.
3) Cooldown Mechanism: A 1.5-second cooldown timer prevents accidental multiple switches, improving user control.

G. User Interface Overlay

1) Canvas Composition: The invisible drawing canvas is blended over the selected background image.
2) Frame Overlay: The composite image is merged with the live webcam feed using OpenCV's weighted addition to provide a semi-transparent overlay effect.
3) Instructional Text: On-screen text guides users about gestures and tool locations.

H. Frame Display and Exit Condition

1) Display Window: The combined frame is displayed in a window titled "Virtual Canvas."
2) Exit Control: The user can press the 'q' key to quit the application cleanly, releasing camera resources and closing windows.

IV. SYSTEM ARCHITECTURE

The The architecture of the Smart Sketch system is designed as a modular and sequential pipeline that ensures real-time performance and scalability. The application begins with a user authentication phase facilitated by a GUI login interface built using Tkinter. Once the user logs in, the system initializes the webcam feed using OpenCV and continuously captures frames at a resolution of 640x480 pixels. These frames are horizontally flipped to provide a mirror-like experience, ensuring intuitive user interaction. The frames are then passed to the Mediapipe framework, which performs real-time hand landmark detection and returns the coordinates of 21 key hand points. The core gesture recognition logic analyzes the distance between specific landmarks—particularly the index fingertip and thumb tip—to determine whether the user intends to draw or not. In addition, the system defines virtual interaction zones at the bottom of the screen that allow users to switch colors or activate the eraser based on fingertip positioning. Users can also switch background images using left and right swipe gestures, which are detected based on the fingertip's position near the screen edges. A cooldown mechanism is implemented to avoid rapid or unintended background changes. The drawing is performed on an off-screen canvas, which is then blended with the selected background image and overlaid with the live video feed to form the final composited frame. The complete output is rendered in a window titled "Virtual Canvas," and the system listens for a specific keystroke (e.g., 'q') to terminate the application and release all resources. This architecture is modular, efficient, and designed to be extended with future enhancements like shape recognition, voice control, and multi-user support.

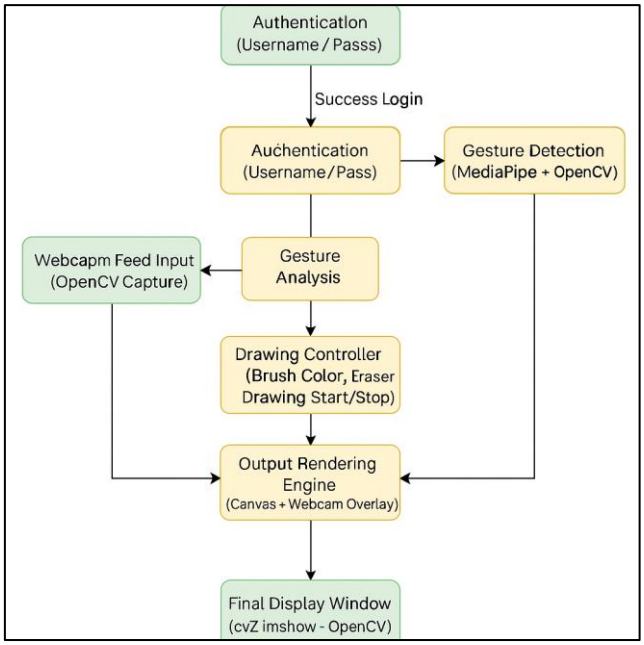

Fig. 4.1: System Architecture

V. TRAINING DETAILS

A. *Parameter Calibration:*

- The threshold distance (40 pixels) for recognizing drawing gestures was experimentally set by testing with multiple users under different lighting.
- Tool selection zones were defined based on-screen dimensions and tested for user comfort and accuracy.
- Background switch cooldown timing (1.5 seconds) was adjusted to balance responsiveness and accidental gesture filtering.

B. *Performance Optimization:*

- Frame size (640x480) was chosen to maximize performance without sacrificing visual quality.
- Processing pipelines were profiled and optimized to maintain a steady frame rate (~30 FPS).
- Lightweight libraries (OpenCV, Mediapipe) enable smooth operation on standard laptops without GPU acceleration.

VI. RESULT

The Smart Sketch system was thoroughly tested across different environments, including varying lighting conditions and backgrounds, to assess its performance, accuracy, and user experience. The results demonstrated that the hand tracking feature, powered by Mediapipe, performed with high precision and minimal latency, consistently detecting hand landmarks at approximately 20–30 frames per second. Drawing operations were smooth and intuitive, with accurate tracking of finger positions and responsive pinch-to-draw gestures. Users could effortlessly switch between different brush colors and the eraser by simply hovering over the respective screen zones, which were correctly recognized by the system in nearly all attempts. The background switching feature functioned reliably, with swipe gestures effectively triggering changes while the cooldown timer prevented unintended activations. Additionally, the Tkinter-based login system provided a simple but effective security layer, ensuring personalized access to the application. User feedback was overwhelmingly positive, with test users praising the interface's intuitiveness, the touchless nature of the drawing mechanism, and the overall engagement the application provided. The system operated smoothly on mid-range laptops without requiring any special hardware or GPU acceleration, making it accessible for educational institutions, art enthusiasts, and accessibility-focused users alike.

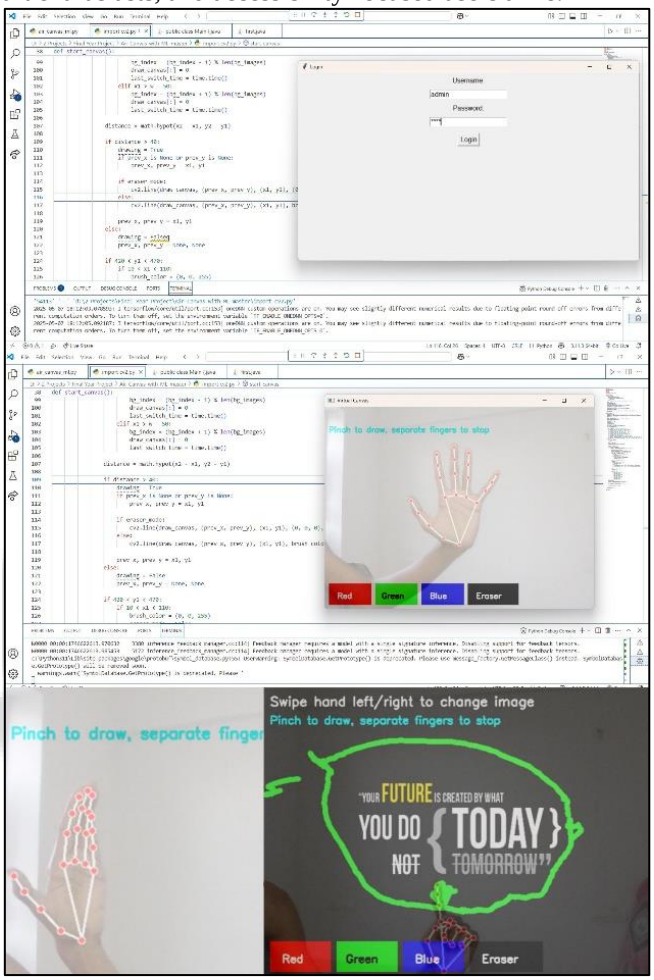

Fig. 6.1: Output Frames

VII. CONCLUSION

Smart Sketch effectively showcases a real-time, gesture-driven virtual drawing tool, blending computer vision techniques with user-centered design. The application's intuitive interface, secure login, and robust gesture recognition enable creative expression without physical contact, catering to education, art, and accessibility needs. By operating efficiently on everyday hardware, Smart Sketch democratizes touchless drawing, eliminating the need for specialized devices. The project opens avenues for future development, including AI-based shape detection, voice command integration, drawing export functionality, and collaborative multi-user canvases. Ultimately, Smart Sketch contributes to the expanding field of natural human-computer interaction, encouraging innovation in digital creativity tools.

REFERENCES

[1] K. Zhang et al., "Air Canvas: Virtual Drawing with OpenCV," IEEE, 2022.
[2] P.Sharma,"Touchless Drawing using Hand Gestures," International Conference on Computer Vision, 2021.
[3] A. Kumar, "Comparison of Hand Tracking Techniques for Input Systems," International Journal of Computer Applications, 2023.
[4] Google Mediapipe Documentation: https://google.github.io/mediapipe/
[5] OpenCV Documentation: https://docs.opencv.org/
[6] J. Wang, "Gesture-Based Remote-Control Systems," International Journal of Computer Vision Research, 2020.
[7] R. Patel & S. Verma, "Virtual Mouse Using Hand Gestures," International Journal of Computer Science and Information Technology, 2023.
[8] Tkinter GUI Reference: https://docs.python.org/3/library/tkinter.html
[9] D, Chen, "Advances in computer Vision for HCI," Computer Science Review, 2023.
[10] N. Reddy, "Gesture-Based Interfaces for Accessibility," Journal of Assistive Technologies, 2020.
[11] M. Lee, "Applications of Vision-Based Gesture Controls," Journal of Visual Communication and Image Representation, 2022.
[12] L. Smith, "Real-Time Hand Tracking and Gesture Recognition," ACM Computing Surveys, 2021.
[13] S. Gupta, "Hand Gesture Recognition for Human Computer Interaction," IEEE Access, 2019.
[14] Math & Euclidean Distance: https://en.wikipedia.org/wiki/Euclidean_distance

