International Journal of Science and Research Archive
eISSN: 2582-8185
Cross Ref DOI: 10.30574/ijsra
Journal homepage: https://ijsra.net/

(RESEARCH ARTICLE)

Check for updates

Air canvas: A real -time touchless drawing system using computer vision

Aarya Sunil Paradkar *, Kundan Laxman Pednekar, Sai Vignesh Yenugudhati, Atharva Deepak Jadhav, Kaustubh Ravindra Parihar and Megha Patil

Department of Mechatronics, Faculty- Megha Patil, Symbiosis Skill and Professional University, Pune. India.

International Journal of Science and Research Archive, 2025, 15(02), 549-558

Publication history: Received on 23 March 2025; revised on 09 May 2025; accepted on 11 May 2025

Article DOI: https://doi.org/10.30574/ijsra.2025.15.2.1331

Abstract

This project presents a novel and interactive approach to virtual drawing through an Air Canvas system that enables users to draw in mid-air using a colored pen tip. By leveraging computer vision techniques in Python with the OpenCV library, the system tracks the pen's movement in real-time without requiring physical contact or expensive hardware like depth sensors or hand tracking modules. The pen's color is isolated using HSV color space thresholding and morphological operations, allowing for precise tracking. Users can perform drawing actions, switch colors, or clear the screen using simple gestures with the pen. The system is implemented with an adjustable color detection interface for real-time tuning and provides a low-cost, efficient, and intuitive solution for applications in education, art, and human-computer interaction. The Air Canvas demonstrates how accessible technology and basic image processing can be used to develop immersive and touchless interactive systems.

Keywords: CV2; NumPy; Python; Mask; Contour; HSV(Hue Saturation Valve); Tracking; Media Pipe hands; Blue Green Red; Canvas

1. Introduction

In recent years, Human-Computer Interaction (HCI) has been evolving toward more natural, intuitive, and immersive methods of communication. Traditional tools such as keyboards, mice, and touchscreens, while effective, are limited in enabling truly fluid interaction, especially in creative or educational contexts. In contrast, humans naturally communicate using gestures, hand motions, and spatial awareness. Bridging this gap between natural communication and digital interaction forms the basis of innovations like the Air Canvas project.

The Air Canvas project demonstrates a novel, contactless method for drawing in a virtual space using a simple, colored pen. Unlike conventional gesture-based systems that require complex hand tracking algorithms, additional hardware sensors, or libraries like MediaPipe, this system utilizes a minimalist and efficient approach. It depends solely on color detection techniques implemented using the cv2 library (OpenCV in Python). The core idea is to isolate and track a single-colored object — the tip of a pen — and use its movement to generate visual strokes on a digital canvas.

The system architecture is designed to capture live video input through a webcam and process each frame to identify a specific color in the HSV (Hue, Saturation, Value) color space. HSV is preferred over the standard BGR format because it allows for more robust color filtering under varying lighting conditions. In the provided implementation, a user-adjustable set of HSV thresholds is managed via trackbars, allowing for dynamic tuning of the color to be detected — ideal for adapting to different pen colors or lighting environments.

Once the pen tip is isolated via a binary mask, morphological operations such as erosion, dilation, and opening are applied to refine the mask and eliminate noise. The result is a clean detection of the pen tip, from which contours are

* Corresponding author: Aarya Paradkar

extracted. The largest contour is assumed to represent the pen, and its center is computed using image moments. This center acts as the current drawing point.

The application includes a simple GUI with selectable color options and a "CLEAR ALL" button embedded directly into the video frame. The user can change colors or clear the screen by hovering the pen over the corresponding buttons in the top area of the frame. The system checks if the detected pen tip's coordinates fall within predefined regions for these buttons, and based on the location, it either switches the drawing color or clears the canvas.

The actual drawing process is handled by maintaining multiple deques (double-ended queues) for each color. These queues store the sequence of points where the pen tip has been detected. By connecting these points using lines, the system simulates freehand drawing. A separate digital canvas is maintained in parallel with the video feed to render persistent strokes. The modular structure ensures that strokes of different colors are stored and rendered separately, providing clear and consistent drawing behavior.

What makes this approach particularly compelling is its simplicity and effectiveness. There is no need for sophisticated machine learning models or heavy libraries. Everything is managed through basic computer vision principles — color segmentation, contour detection, and coordinate mapping. This design choice makes the system lightweight, easy to run on basic hardware, and highly customizable. Users can adapt it to their needs simply by changing the pen color or adjusting HSV thresholds using the built-in trackbars.

Furthermore, the system is robust in terms of performance. When the pen tip is not detected (e.g., it leaves the frame), the code adds new deques to maintain clean stroke separation. This prevents errant lines from connecting unrelated points, a common issue in real-time tracking applications. This thoughtful implementation ensures that the drawing remains as natural and uninterrupted as possible.

The application also demonstrates good user experience design by providing visual feedback — the pen tip is highlighted with a circle, and drawn lines appear both on the live frame and the persistent canvas. By integrating the control interface directly into the camera feed, the user does not need additional GUI elements, keeping interaction simple and focused.

The Air Canvas project highlights a creative use of fundamental computer vision tools to deliver a functional and user-friendly drawing experience. By focusing on tracking a colored pen tip instead of the full hand, the system simplifies the technical challenge without compromising usability. Built entirely with Python and the cv2 library, it offers a

cost-effective and accessible way to explore gesture-based interaction and real-time image processing. This framework also lays the foundation for future enhancements such as shape recognition, gesture-based shortcuts, or even collaborative virtual drawing.

2. Literature Review

Air Canvas application using opencv and numpy in Python by Saurabh Uday Saoji and others it stated that Writing in air has been one of the most fascinating and challenging research areas in field of image processing and pattern recognition in the recent years. [1]

Air Canvas by Aniket Sandbhor, Prasad Rane, Prathamesh Shirole, Pawan Phapale stated Drawing is fundamental to all other arts. It is how artists structure, plan and negotiate space. Many years back the natural artist used to draw on stone by using charcoal or branches of trees. [2]

Air Canvas: Drawing in Air using by AI Prof. Hemlata A. Shinde, Shravani M. Jagtap, Anushka A.Kalpund, Pranita B. More, Ayushi A. Parkale they in their research paper stated that in today's generation, where technology has emerged to such a greater level that Human Computer Interaction has become increasingly important part of our daily life. [3]

Air Canvas application by B.VenuGopal et.al. they stated that Writing in Air is a Fascinating thing. This can be achieved by Computer Vision Techniques. [4]

Virtual Air Canvas by Akhil Mohan, Sreema E R, Leshma Mohandas, Prabath P U, Saeedh Mohammed stated that an important development in human-computer interaction is this creative use of open-source frameworks to create gesture- controlled whiteboard technology. [5]

Air Canvas with Python - opencv and mediapipe by Mr. Pradeep1, Mr. Ashwani Shivam2, Mr. Tushar Sharma3, Mr. Rajat Singh4 said that Writing in air has recently emerged as a captivating and complex research area within image processing and pattern recognition, significantly enhancing humanmachine interactions.[6]

Air Canvas by Gaurav Boraste1, Ajay Vispute2, Sahil Gaikwad3, Prof. Puspendu Biswas4 stated that The development of the Air Canvas application, involved training on various hand images to accurately identify finger positions. [7]

Air Canvas using artificial intelligence by Ms. Sowndhariya K, Ms. Jeevajanani R, stated that English text can be written and sketched over theair in front of the camera using a real-time video-based pointing technology known as air-canvas. [8]

The Virtual Air Canvas Using Image Processing by K Pavithra; A Geetha; R Chinnaiyan said in there paper that One of the most interesting and challenging research focuses on pattern recognition and image processing has emerged in recent days is writing in the air. [9]

3. Theory and Related Works

In recent years, the integration of computer vision techniques in Human-Computer Interaction (HCI) has opened new avenues for touchless interface systems. One such practical application is an Air Canvas, which enables users to draw in the air with a colored object (usually a pen) while the system detects and renders the motion on a digital canvas. This system operates primarily on color detection and object tracking principles using the cv2 (OpenCV) library in Python. The webcam captures real-time video frames, which are then converted from the BGR color space to HSV (Hue, Saturation, Value). HSV is preferred over RGB for color tracking because it separates chromatic content (hue) from intensity (value), making it more robust under varying lighting conditions.

Key components of the system include:

- Color Masking: A specific color range is defined using trackbars to isolate the pen tip from the background.
- Contour Detection: The masked binary image is processed to detect contours, and the largest one is assumed to be the pen.
- Center Calculation: Using image moments, the centroid of the detected contour is calculated, which represents the pen tip's position.
- Drawing Logic: Based on the detected coordinates, lines are drawn between consecutive points usingcv2.line(), simulating freehand drawing.
- Interactive UI: The top area of the frame is reserved for control buttons (e.g., color selection and clear), which respond to the pen's position.

This approach avoids complex models such as hand pose estimation or skeletal tracking, making it lightweight and responsive. It also reduces computational load and works effectively on low- end systems using only a basic webcam. Several works in literature and industry focus on gesture- based and touchless interaction systems. Some of the most relevant ones are:

- MediaPipe Hands (Google): A state-of- the-art real-time hand tracking model that detects 21 landmarks on the hand. While highly accurate, it requires GPU support and higher computational power. In contrast, the Air Canvas uses a simpler, color-based approach suitable for real-time tasks on low- end systems.
- Virtual Drawing Boards using Leap Motion: Leap Motion controllers allow precise hand and finger tracking. However, they are expensive and require specialized hardware, whereas the Air Canvas only uses a webcam.
- Gesture Recognition using CNNs and Deep Learning: Many academic projects use deep learning to classify hand gestures. While accurate, training and deploying these models require datasets, pre- processing, and considerable computing resources, which are not needed for color masking approaches.
- Augmented Reality (AR) Drawing Apps: Applications on platforms like ARKit or ARCore enable drawing in 3D space. These systems are advanced but rely on mobile sensors and frameworks, while the Air Canvas is simple, desktop-based, and easy to implement.
- "Drawing with Webcam" Projects on OpenCV: Similar projects exist where users draw using a colored object. However, many of them lack dynamic color selection, a UI overlay, or clear separation of multiple drawing colors. Your project adds value by implementing interactive buttons and managing multiple drawing queues. cooled air blower to release the excess heat into the atmosphere. It has a cooling vent which transfers the air to a fan which is connected to a pump which releases the air to the atmosphere.

4. Work Flow

4.1. Flow Chart of Air Canvas

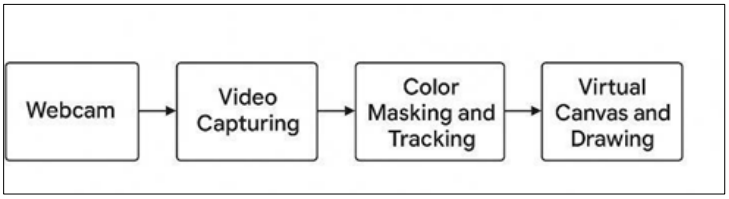

Figure 1 Working Diagram

4.2. Block Diagram

- Start: This marks the beginning of the program. It signifies the execution point where the Python interpreter starts running the script. From here, all system components begin initializing.

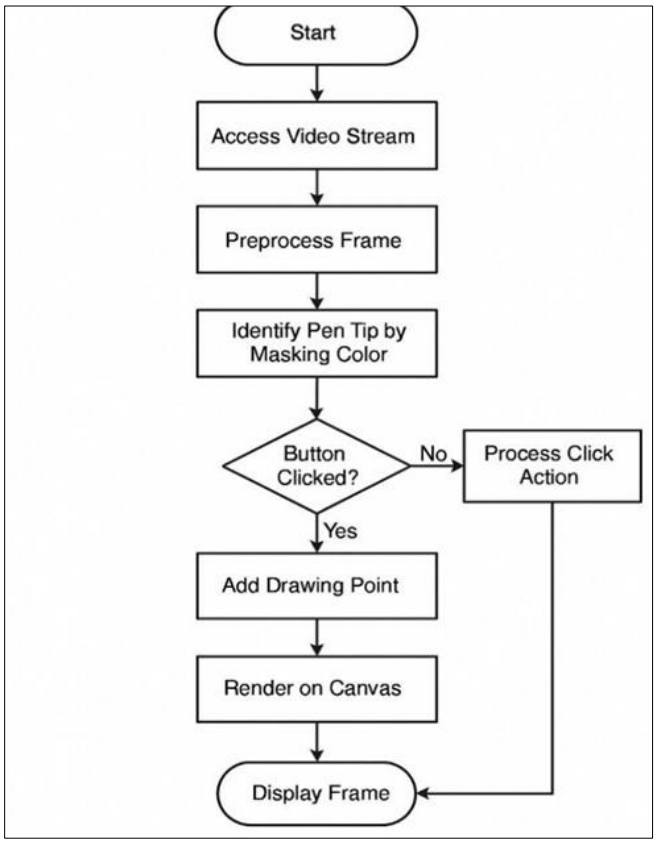

Figure 2 Work Flow Diagram

- Initialize Variables and Trackbars; In this step, various initial setups are done:
  - Importing libraries like cv2, numpy, and deque.
  - Creating trackbars using cv2.createTrackbar() to allow dynamic tuning of HSV values, which helps detect the colored pen tip.
  - Initializing deque structures for storing drawing points for each color (blue, green, red, yellow). These deques help maintain a history of points used to draw continuous lines.
  - Setting up the paint window with buttons like CLEAR, BLUE, GREEN, RED, and YELLOW, which are used as tool selection areas.
- Open Webcam: The system then opens the computer's webcam using cv2.VideoCapture(0). The webcam acts as the primary input source and streams real-time video, which is later processed frame by frame to detect the pen's movement.

- Capture Frame and Flip; Each frame captured from the webcam is:
  - Flipped horizontally using cv2.flip(frame, 1) so that the user's movement appears more natural, like looking into a mirror.
  - This ensures better hand-eye coordination when interacting with the virtual canvas.
- Convert BGR to HSV: The default color format for images in OpenCV is BGR. However, BGR isn't suitable for color segmentation because it's sensitive to lighting.
  - The frame is converted to HSV (Hue, Saturation, Value) using cv2.cvtColor().
  - HSV makes it easier to isolate specific colors (like a colored pen tip) under varying lighting conditions.
- Get HSV Threshold Values from Trackbars; The system reads the upper and lower HSV bounds set by the user using the trackbars:
  - cv2.getTrackbarPos() retrieves real- time values.
  - These bounds determine which color the system should detect.
- Apply Color Mask; With HSV bounds ready, the program:
  - Applies color thresholding using cv2.inRange() to create a binary mask.
  - This mask shows only the pixels that fall within the specified color range (i.e., the pen tip).
  - Morphological operations (like erosion, dilation, and opening) help remove noise and improve accuracy in detecting the colored object.
- Find Contours: Contours are outlines or boundaries of shapes detected in the mask.
  - The program uses cv2.findContours() to locate all shapes formed by the white pixels in the mask.
  - These contours are potential candidates for identifying the pen.
- Contour Found: The system checks if any contour was found:
  - If no contours are found, it implies that the colored pen tip is not visible.
  - If a contour is found, the system proceeds to process it further.
- Calculate Center of Largest Contour:
  - The system sorts all contours and picks the largest one (most likely to be the pen tip).
  - It then calculates the center point (centroid) of that contour using image moments.
  - This center is treated as the current position of the pen tip.
- Center Within Toolbar Area; Now the system checks whether this center lies within the top toolbar area of the screen ($y \leq 65$):
  - If it does, then it's treated as a click/gesture on one of the tools like color selection or clear.
  - If it does not, then it's considered a drawing movement, and the system prepares to draw based on the selected color.
- Toolbar Action (Color Selection / Clear All); If the pen tip is in the toolbar area:
  - CLEAR button clicked: The system clears all color point lists and resets the canvas.
  - Color box clicked: The color Index is updated based on where the pen was placed
    - enabling the user to switch between blue, green, red, and yellow pens.
- Append Drawing Points; If the pen tip isn't in the toolbar area:
  - The detected center point is appended to the deque list of the currently selected color.
  - This allows the system to track the pen's motion and render it as a line on the screen.

Each frame can append hundreds of points over time, which is why the deque has a fixed size (maxlen=512).

- Draw on Canvas; This is where the actual virtual drawing happens:
  - It connects the recorded points using
  - cv2.line() to draw lines on both:
    - The live webcam frame, and
    - The static white canvas (paintWindow), which stores the full artwork.

This process continues for every frame, producing a smooth, continuous line.

- Display Frame, Canvas, and Mask; Three display windows are shown simultaneously:
  - "Tracking" – live webcam feed with buttons and current pen stroke.
  - "Paint" – the white canvas where the artwork is created.
  - "Mask" – binary output showing only the pixels in the color range.

This real-time feedback helps the user understand what the system is detecting and how it responds.

- 'q' Key Pressed; The loop continues until the user presses the 'q' key:
  - If pressed, the loop is broken, and the application prepares to shut down.
  - If not, it goes back to Step 4 to capture the next frame and repeat the process.
- End / Exit; Upon exiting:

- The webcam is released using
- cap.release().
- All OpenCV windows are closed with cv2.destroyAllWindows() to clean up system resources.

This ensures a proper shutdown and avoids webcam lock or memory issues.

5. Algorithms

5.1. Color Detection using HSV Masking: To isolate and detect the colored pen tip in real-time from webcam video frames.

- Algorithm Steps:
  - Convert each frame from BGR to HSV color space using cv2.cvtColor().
  - Retrieve user-defined HSV thresholds through trackbars for dynamic tuning.
  - Generate a binary mask using cv2.inRange() where the pixels within the specified HSV range are turned white (255), and others black (0).
  - Apply morphological operations (erode, morphologyEx, and dilate) to reduce noise and enhance the detection of the pen tip.
- Result: A clean binary mask that isolates the colored pen tip from the background, allowing accurate tracking.

5.2. Contour Detection and Center Extraction: To locate the exact position of the pen tip from the mask.

- Algorithm Steps:
  - Use cv2.findContours() to extract the contours from the binary mask.
  - Sort the contours by area and pick the largest one, assuming it's the pen tip.
  - Use cv2.minEnclosingCircle() to draw a circle around the detected contour for visual feedback.
  - Calculate the centroid (center point) of the contour using image moments via cv2.moments().
- Result: A coordinate point representing the pen tip's position in the frame.

5.3. Button Press Detection (Virtual UI): To allow tool selection (like color change or canvas clear) using the pen tip as a pointer.

- Algorithm Steps:
  - Check if the pen's y-coordinate (center[1]) lies within the top region (<= 65 pixels) where the virtual buttons are drawn.
  - If so, check which x-coordinate range the pen lies in to determine:
    - Clear the canvas
    - Change color (Blue, Green, Red, Yellow)
  - Update color selection state using a colorIndex variable.
- Result: Contactless interaction with the UI using gestures instead of physical clicks.

5.4. Drawing using Deques: To store and draw the pen path as the user moves it in the air.

- Algorithm Steps:
  - Use four lists of deques (for each color) to store the detected points (x, y) over time.
  - Append the current center coordinates to the correct deque depending on colorIndex.
  - When the pen is lifted (not detected), a new deque is added for the current color to separate strokes.
  - Use cv2.line() to draw lines between consecutive points on both:
    - The live webcam feed (feedback to the user)
    - The static canvas (paintWindow) that acts like a digital whiteboard
- Result: Real-time drawing with color switching and stroke separation handled automatically.

5.5. Frame Display and System Control: To render the results and manage exit conditions.

- Algorithm Steps:
  - Continuously show the following frames:
    - "Tracking" – The live webcam feed with virtual buttons and tracking visuals.
    - "Paint" – The persistent canvas with all the drawn strokes.
    - "mask" – The binary mask of the pen tip for debugging and calibration.
  - Allow the user to press 'q' to quit the application and release the webcam.

6. Result and Discussion

6.1. Tracking Frame: This is the main live feed window that shows:

- The real-time video stream from the webcam (flipped horizontally for intuitive interaction).
- The interactive toolbar at the top (with color buttons and a “Clear” button).
- A circle around the detected pen tip, indicating the system is actively tracking the marker.
- The drawing trail as the user moves the colored marker across the screen.
- Helps users visualize both their movements and immediate drawing feedback.
- Color strokes drawn here are temporary — they depend on the pen's live movement.

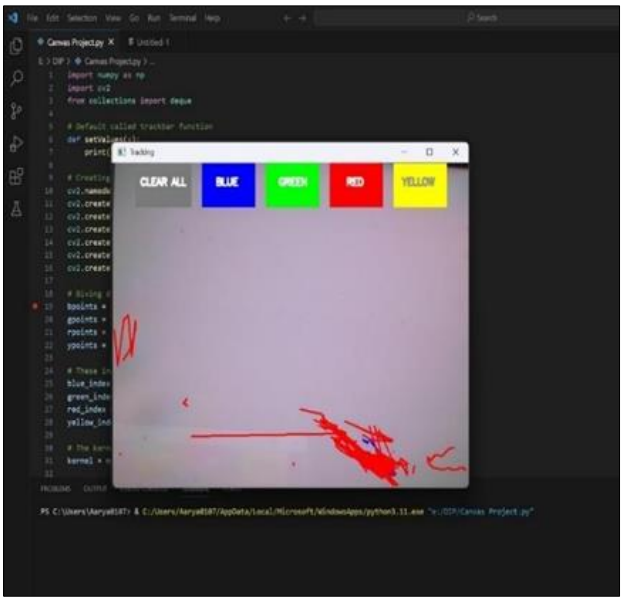

Figure 3 Tracking Frame

6.2. Paint Frame: This frame acts as the final canvas — it stores and continuously updates the actual drawing done by the user.

- Unlike the Tracking frame, this doesn’t reset between frames.
- It shows a clean version of the drawing with only the user's brush strokes.
- Toolbar is not interactive here — it’s just a static setup drawn once during initialization.

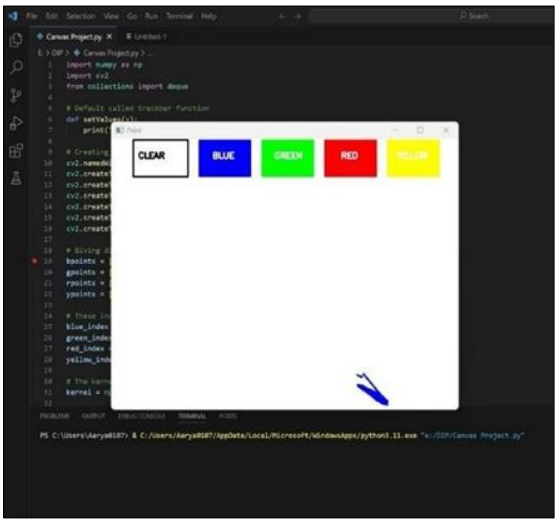

Figure 4 Paint Frame

- All strokes of selected colors (blue, green, red, yellow) are persistently stored here using the deque lists.
- This frame is ideal for saving the final drawing if needed.
- Cleaner than the live feed because it doesn't include the pen tip circle or video background

6.3. Mask Frame: This window displays the binary image used internally to detect the pen tip based on color.

- The mask is generated by applying the inRange() function on the HSV-converted frame.
- Pixels within the specified HSV range appear white (255), while all others are black (0).
- After processing (erosion, opening, dilation), this frame helps clearly visualize which parts of the image are considered valid pen locations.
- It's essential for debugging the color tracking system.
- Helps users tune the HSV trackbars correctly.
- If the pen is not detected, the user can check the mask frame to understand if the color is out of range.

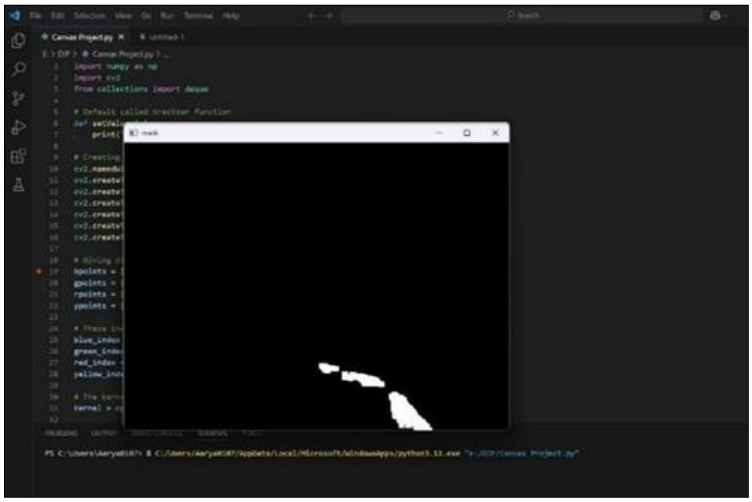

Figure 5 Mask Frame

6.4. Color Detectors Frame

The "Color detectors" window seen in the image plays a crucial role in the initial calibration and customization of the Air Canvas system. Its main purpose is to let users fine- tune the HSV (Hue, Saturation, and Value) ranges that define the color of the pen or marker being used for drawing. Since lighting conditions, webcam quality, and the color of the environment can significantly affect color detection, having a real- time adjustable interface ensures that the system can adapt to these variations without modifying the core code.

This window contains sliders (trackbars) for both Upper and Lower limits of Hue, Saturation, and Value, allowing the user to define a specific range in which the desired color lies. When the values are adjusted, they immediately affect the color masking process in the background, making it easier to find the correct threshold for detecting the pen tip accurately. This feature is especially useful when testing different pen colors or when there is more than one user using different markers.

This interactive window makes the system more user-friendly, versatile, and robust by allowing real- time configuration of color detection parameters. It is essential for ensuring that the virtual drawing system works accurately and consistently across different scenarios.

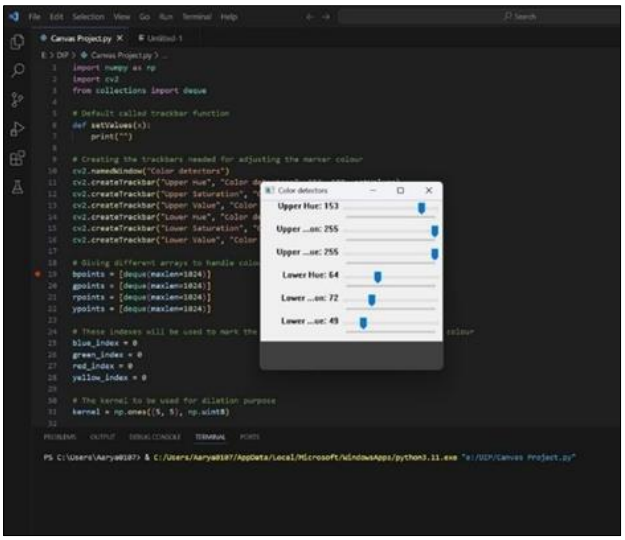

Figure 6 Colour Detectors Frame

The Air Canvas system successfully demonstrates a contactless and intuitive way of interacting with a computer using simple coloured object tracking, providing an accessible alternative to traditional input devices. The system's core functionality is built upon real-time image processing using OpenCV, allowing users to draw or write in the air by moving a colored pen in front of a webcam.

One of the major strengths of this implementation lies in its simplicity and cost-effectiveness. Unlike gesture recognition systems that rely on advanced machine learning models or depth cameras (e.g., MediaPipe, Kinect), this system uses only basic color filtering in the HSV color space, which can run efficiently on most low-spec computers. This makes the system highly portable and deployable, especially in educational and creative environments.

However, while the color-based detection technique is effective, it also introduces certain limitations. The system's performance is heavily dependent on the lighting conditions and background. If the environment has objects or surfaces with similar colors to the pen, false detections may occur, reducing accuracy. Furthermore, the need for manual calibration of HSV thresholds via trackbars might be unintuitive for general users and can vary across different webcams or room settings.

The virtual button mechanism—triggered by placing the pen tip in the top region of the screen— is a clever yet minimalistic approach to tool selection. It avoids the need for physical buttons or hand gestures but could be made more robust and user-friendly with audio feedback or tool highlights.

In terms of drawing experience, the use of deques for stroke tracking ensures smooth lines and stroke separation, enabling multi-color drawing. However, since the pen must remain visible to the camera at all times, momentary occlusions or fast hand movements may lead to missed frames or broken strokes.

This system has strong potential in fields such as remote education, digital art, presentations, and accessibility tools, where touchless interfaces are becoming increasingly valuable. Future work may involve improving detection robustness using machine learning-based color segmentation or integrating gesture-based commands while still preserving the lightweight, offline nature of the project.

7. Conclusion

The *Air Canvas* project presents a novel and efficient approach to virtual drawing by allowing users to paint in mid-air using a colored pen, with the output reflected live on the screen. Built using Python and the cv2 (OpenCV) library, the system utilizes real-time video feed from a standard webcam to detect and track the pen tip based on its color in the HSV color space. By applying a combination of image processing techniques like color masking, morphological transformations, and contour detection, the system accurately follows the movement of the colored pen and visualizes it as strokes on a digital canvas.

One of the core achievements of this project is the elimination of physical interaction with input devices. Unlike conventional drawing tools that require touchscreens, styluses, or dedicated hardware like drawing tablets, this system requires nothing more than a webcam and a colored object. This makes it highly cost-effective and accessible, particularly for educational, creative, or assistive technology applications.

The GUI includes a color selection and clear button region at the top of the screen, offering basic interactivity through simple upward movements of the pen. The integration of trackbars to fine-tune the HSV values ensures robustness under varying lighting conditions and enhances the accuracy of pen tip detection.

Furthermore, the use of deques (double-ended queues) for storing stroke points ensures smooth drawing performance and easy handling of multiple strokes or pen lifts. The canvas window keeps track of all drawing data persistently until explicitly cleared, while other output frames such as the “Tracking” and “Mask” windows help visualize the real-time process of detection and rendering.

The project demonstrates how effective and interactive systems can be developed using fundamental computer vision techniques without relying on complex machine learning models like hand tracking, pose estimation, or external libraries like MediaPipe. It provides a strong foundation for future enhancements such as adding gesture-based tool switching, multi-color drawing, saving the canvas, or even extending the concept into Augmented Reality environments.

In conclusion, this project not only showcases the potential of contactless HCI through computer vision but also proves that simple and accessible tools can be used to build innovative applications. It stands as a testament to the idea that impactful digital interaction does not always require expensive or complicated infrastructure

Compliance with ethical standards

Disclosure of conflict of interest

No conflict of interest to be disclosed.

References

[1] RamachandraH, V., Balaraju, G., Deepika, K., & Sebastian, S. R. (2022, November). Virtual Air Canvas Using OpenCV and Mediapipe. In 2022 International Conference on Futuristic Technologies (INCOFT) (pp. 1-4). IEEE.

[2] Air Canvas application B. VenuGopal1, Ch.Seshagiri Rao2, B.Ajay Kumar3, D. Prakash4, Md. ShakeelAhmed5 1,2,3,4Department of Information Technology,VVIT , AP,India 5AssociateProfessor, Dept. of Information Technology, VVIT, AP, India.

[3] Saoji, S. U., Dua, N., Choudhary, A. K., & Phogat, B. (2021). Air canvas application using Opencv and numpy in python. IRJET, 8(08), 2395.

[4] Mohan, A., Mohandas, L., & Mohammed, S. (2023). Virtual Air Canvas. In International Journal on Emerging Research Areas (IJERA). Air Canvas with python - opencv and mediapipe Mr. Pradeep1 , Mr. Ashwani Shivam2 , Mr. Tushar Sharma3 , Mr. Rajat Singh4 1 Assistant Professor 2,3,4 Student CSE Department HMR Institute Of Technology And Management, GGSIPU, Delhi- 110036

[5] Air Canvas Gaurav Boraste1 , Ajay Vispute2, Sahil Gaikwad3, Prof. Puspendu Biswas4 1,2,3Computer Engineering Student, S.M.E.S. Sanghavi College of Engineering, Nashik, India. 4H.O.D, Computer Engineering Department, S.M.E.S. Sanghavi College of Engineering, Nashik, India.

[6] Air Canvas using Artificial Intelligence by Ms. Sowndhariya K, Ms. Jeevajanani R, Pavithra, K., Geetha, A., & Chinnaiyan, R. (2023, October). The Virtual Air Canvas Using Image Processing. In 2023 International Conference on New Frontiers in Communication, Automation, Management and Security (ICCAMS) (Vol. 1, pp. 1-6). IEEE.

[7] Venkatakrishnan, R., Venkatakrishnan, R., Chung, C. H., Wang, Y. S., & Babu, S. (2022). Investigating a combination of input modalities, canvas geometries, and inking triggers on on-air handwriting in virtual reality. ACM Transactions on Applied Perception, 19(4),1-19.

[8] Saoji, S. U., Dua, N., Choudhary, A. K., & Phogat, B. (2021). Air canvas application using Opencv and numpy in python. IRJET, 8(08), 2395.