International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 09 Issue: 05 | May - 2025 SJIF Rating: 8.586 ISSN: 2582-3930

Virtual Painting with OpenCV Using Python

V.Sadhwika[1], A. Satyasree[2], K. Yasaswini[3], D. Madhavi Latha[4]

[1,2,3] Students, Dept of Electronics and Communication Engineering, Geethanjali College of Engineering and Technology, Telangana, India

[4]Assistant Professor, Dept of Electronics and Communication Engineering, Geethanjali College of Engineering and Technology, Telangana, India

---

Abstract - The Virtual Painting system introduces an exciting new way to create digital art by combining artificial intelligence with real-time hand gesture recognition. Developed in Python, the system uses a webcam to capture live video, translating simple hand movements into drawing commands on a virtual canvas. By eliminating the need for traditional input tools like a mouse, stylus, or touchscreen, it allows users to draw freely with just their hands. This touch-free interaction not only makes digital art more accessible to a broader audience but also opens up new possibilities for creative expression.

At the core of the system are advanced computer vision technologies and widely-used open- source libraries such as OpenCV, Mediapipe, and NumPy. Mediapipe is particularly important, offering a highly accurate hand detection model that can track 21 distinct landmarks on the hand in real time. This precise tracking enables users to control the application through intuitive gestures—selecting drawing tools, switching modes, erasing, clearing the canvas, and more.

Whether drawing freehand or creating geometric shapes like circles, rectangles, and ellipses, the system provides a flexible and responsive platform for various artistic needs. Ultimately, this project showcases how the fusion of AI and gesture control can reshape the way we interact with digital spaces, making technology not only more functional but also more human-centered and creatively empowering.

Key Words: Python, Artificial Intelligence, Virtual Painting, Hand Gesture Recognition, OpenCV, Media pipe, NumPy, Canvas, Hand Tracking, Gesture Recognition.

1. INTRODUCTION

In today's rapidly evolving field of human-computer interaction, gesture recognition has emerged as a powerful tool for enabling intuitive and touchless control. The project titled *"Virtual Painting Using Hand Gestures"* offers a creative and modern way to interact with digital canvases, removing the need for traditional input devices like a mouse, keyboard, or stylus. Instead, it allows users to draw, paint, and erase using simple hand movements captured through a standard webcam.

Developed in Python, this system makes use of key libraries such as **MediaPipe**, **OpenCV**, and **NumPy**. MediaPipe accurately detects 21 hand landmarks, which are crucial for recognizing user gestures. These gestures are then linked to specific drawing actions like selecting colors, drawing shapes (e.g., circles, rectangles), erasing, clearing the canvas, or saving the artwork. OpenCV manages real-time image processing and visual output, while NumPy supports efficient numerical operations.

What makes this system stand out is its user-friendly and immersive interface. Features like adjustable eraser size, canvas clearing, and saving options enhance usability, making it suitable for digital art, education, and even creative therapy. By combining real-time gesture tracking with computer vision, this project highlights the potential of AI to create accessible, engaging, and hands-free creative tools.

2. EXISTING SYSTEMS

Traditional virtual painting systems that incorporate gesture recognition generally offer only basic drawing functions. Most of these platforms allow users to draw simple lines or shapes on a digital canvas through hand tracking, but they fall short when it comes to practical features needed for real- world use. For instance, many lack the ability to save artwork, clear the canvas easily, or customize tools such as adjusting the eraser size. This limits users to a very basic and often frustrating drawing

experience, with little control over managing or preserving their creations.

Additionally, many existing systems have rigid interaction designs that make switching between tools or modes cumbersome and unintuitive. This lack of flexibility reduces their effectiveness for users who want to engage in detailed or prolonged digital art projects. As a result, these solutions are less suitable for artists or educators looking for a versatile and efficient virtual painting experience.

3. PROPOSED SYSTEM

The proposed system builds significantly on the basic virtual painting framework by incorporating several advanced and user-friendly features. In addition to gesture-based freehand drawing, it introduces essential utilities like:

- Save functionality: Users can save their artwork at any point with a simple gesture, preserving their creations without requiring manual file operations.
- Clear canvas option: A dedicated gesture allows users to instantly clear the canvas, making it easier to restart or remove mistakes.
- Variable eraser: The system provides a dynamic eraser whose size can be adjusted, offering greater control and flexibility during corrections or detailed editing.

These enhancements are made possible through precise hand gesture tracking using Mediapipe, along with real-time rendering and event handling through OpenCV.

By combining these features, the system transitions from a basic virtual painter to a more complete digital art tool that is intuitive, efficient, and practical for users of all skill levels.

It not only improves usability but also expands the creative possibilities, making it ideal for educational, artistic, and accessibility-focused applications.

4. METHODOLOGY

The virtual painting system is developed using Python and integrates three key libraries — **Mediapipe**, **OpenCV**, and **NumPy** — to recognize hand gestures and enable drawing actions in real- time. The webcam captures the video feed which is processed using OpenCV.

Mediapipe's Hand module detects hand landmarks, specifically identifying the position of the index and middle fingers.Based on finger gestures, the user can select tools like line, rectangle, circle, freehand draw, or erase.

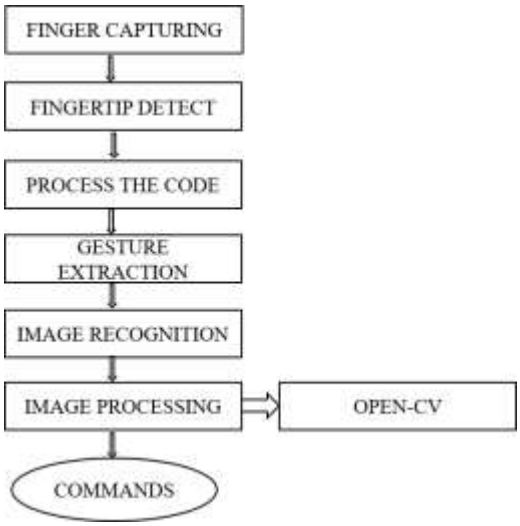

Fig:Flow Chart

A custom tool panel is either loaded as an image or generated programmatically using OpenCV drawing functions. The drawing is performed on a NumPy-based canvas, which allows for real-time, responsive updates. Different gestures determine actions:

- One finger raised for tool selection.
- Two fingers raised for drawing.
- Three fingers raised for erasing.

The system supports live updates, saving, reloading previous work, and clearing the canvas, providing a complete interactive drawing experience.

4.1 **Setting Up Dependencies:** The first step involves importing the necessary libraries for the project. We use cv2 to manage the webcam feed and create a graphical interface for the tools and canvas. The numpy library is used to create and manipulate the drawing canvas, which is essentially an array. For hand tracking, we utilize Mediapipe's solutions.hands

module, which is highly efficient for detecting and tracking hand landmarks in real-time. Additionally, time helps in calculating the frames per second (FPS) for performance monitoring, and os assists with any file operations, like saving the drawn output.

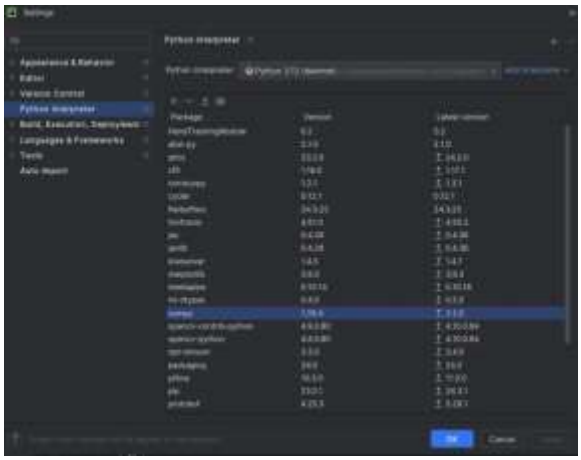

Fig 4.1 : Installing individual modules

4.2 **Hand Detection with Mediapipe:**

To track the hand movements, Mediapipe's Hands module is initialized with custom parameters. We configure it to detect a maximum of one hand at a time and set confidence thresholds for both detection and tracking.

The hand detection process outputs 21 landmarks for each hand, which serve as key points for identifying gestures and their locations in the frame. These landmarks are critical for determining the position of fingers and selecting tools or drawing.

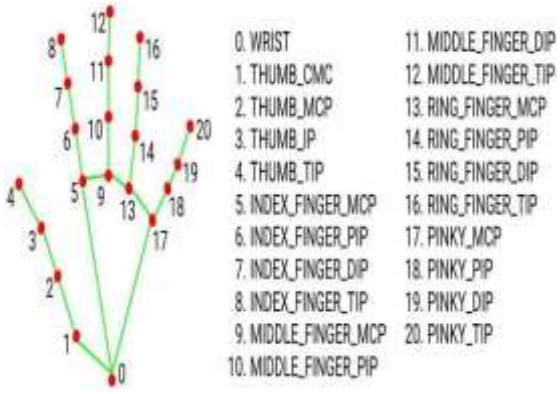

Fig 4.2 Landmarks Of The Hand

4.3. Designing the Tools Panel: The tools panel in this project is designed to provide a seamless user experience for selecting various drawing functionalities. The design integrates two approaches for flexibility:

1.Pre-Designed Image: A visually appealing tools panel image (tools.png) is uploaded and used to display tool options such as LINE, RECTANGLE, CIRCLE, DRAW, and ERASE. This approach allows for high customization, where colors, fonts, and icons can be styled in an external image editor to create a professional look.

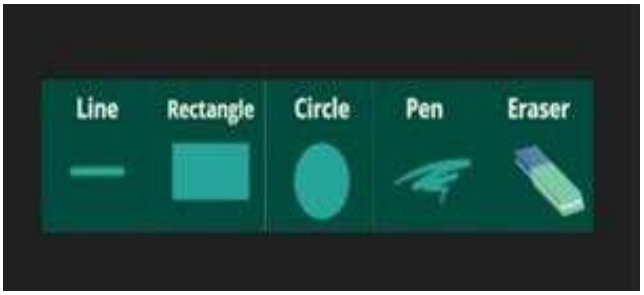

Fig 4.3.1 Uploaded Tools Panel Design

2.Programmatic Design:

When the tools panel image is unavailable, a dynamic panel is generated programmatically using OpenCV.A blank canvas (tools) is created.Tool labels are added with cv2.putText to display the names of the tools.Positioning and font settings ensure clarity and usability.

Fig4.3.2: Programmatically Generated Tools Panel.

4.4. Creating the Drawing Canvas: The drawing canvas is created as a blank image using a NumPy array initialized with zeros, matching the webcam feed's dimensions. This canvas allows independent drawing operations that persist across frames and

can be blended with the live video feed using techniques like cv2.addWeighted.

Fig 4.4.1 Drawing Canvas with Tools Panel

4.5. Webcam Setup and Processing Frames:

The webcam feed is initialized using cv2.VideoCapture, which continuously captures frames in real-time. Each frame is flipped horizontally using cv2.flip to provide a mirrorlike effect, making interactions more intuitive for the user. The frames are then converted to RGB format using cv2.cvtColor, as Mediapipe's hand detection requires RGB input. These processed frames are subsequently passed to the hands.process function to detect hand landmarks.

4.6 Detecting Hand Landmarks:

Once the frames are processed by Mediapipe,the results object provides the hand landmarks if a hand is detected.These landmarks are accessible through results.multi_hand_landmarks and include 21 key points for the detected hand. Each landmark's x, y coordinates are normalized relative to the frame's dimensions. Using these coordinates, we track specific fingers (e.g., the index and middle fingers) to implement tool selection and drawing functionalities.

Fig 4.6 Hand Detection

4.7. Tool Selection Based on Finger Position:

Tool selection is determined by detecting the position of the index finger in the tools panel. The x and y coordinates of the index finger's tip (landmark 8) are compared against predefined regions representing tool buttons. For instance, if the finger is within the bounding box for the "Rectangle" button, the current tool is set to "rectangle".

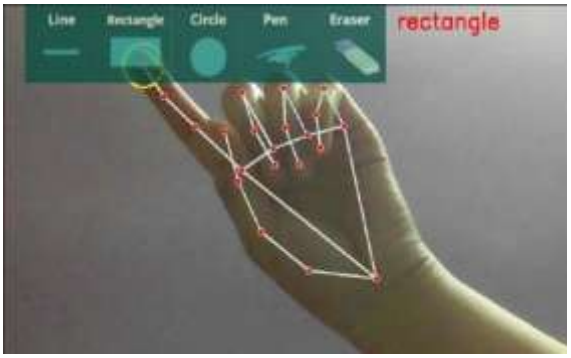

Fig 4.7 Tool Selection

4.8. Drawing on the Canvas

The selected tool determines the drawing functionality. For freehand drawing, continuous lines are drawn between successive positions of the index finger using cv2.line.

For shapes like rectangles and circles, the coordinates of the finger's initial and final positions are used to define the shape's dimensions. The erase tool is implemented by overlaying white circles on the canvas at the finger's position, effectively clearing parts of the drawing. Each functionality is implemented using OpenCV drawing functions like cv2.rectangle, cv2.circle, and cv2.line.

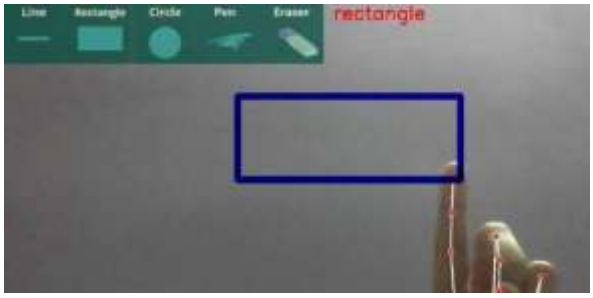

Fig 4.8.1 Drawing on Canvas Using Rectangle

4.9. Combining Canvas with Webcam Feed

To merge the drawings with the live webcam feed, we use cv2.addWeighted, which blends the canvas and the feed based on specified weights. This gives the appearance of drawing directly on the video stream. Additionally, logical operations like cv2.bitwise_or can be used to handle overlays. The combined output is then displayed to the user in realtime, maintaining a smooth and interactive experience.

4.10 Save and Clear Functionality

Keypress events are handled to add additional functionality. When the ESC key is pressed, the current state of the canvas is saved as an image file using cv2.imwrite. This allows the user to preserve their work. To reset the canvas, pressing the c key clears the canvas by setting all its pixel values to zero. This ensures a fresh workspace without restarting the application.

4.11. Exit Program

Finally, the program exits gracefully when the user presses a predefined key (ESC). The webcam feed is released using cap.release(), and all OpenCV windows are closed using cv2.destroyAllWindows. This ensures that the program terminates cleanly and releases system.

5.RESULTS:

Python, OpenCV, and other python modules are the foundation of this project. We learn how to choose an element and use that choice to initiate an action. fresh ideas and techniques for solving problems. When developing the hand tracking module, we learn how to use OpenCV concepts like limiting the active hand detection and assigning the action to the fingers. We acquire the project deployment techniques. We use modern applications like PyCharm, Visual Studio Code, and Figma. We could deal with several platforms, languages, and technologies by working on this project.

Fig 5.1 Drawing Circle on the Canvas

Fig 5.2 Drawing Rectangle on the Canvas

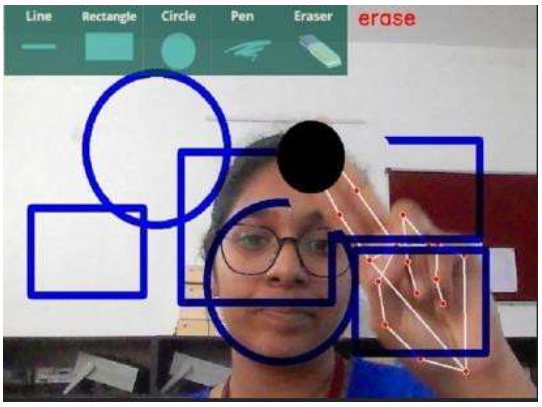

Fig 5.3 Erasing on the Canvas

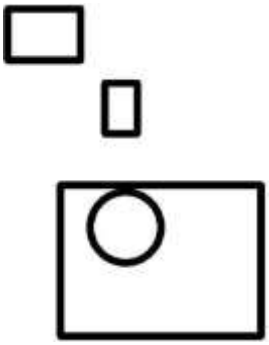

Fig 5.4 Final drawing with saved work

5. APPLICATIONS

- ***Digital Art Creation:*** The system provides an innovative interface for artists and hobbyists to create digital artwork using hand gestures, eliminating the need for traditional input tools like styluses or graphic tablets.
- ***Interactive Educational Tools:*** This technology can be integrated into learning platforms for young students or in special education to facilitate interactive drawing, enhancing engagement and learning outcomes through kinesthetic interaction.
- ***Therapeutic and Rehabilitation Applications:*** Virtual painting via gesture tracking can serve as a therapeutic tool to assist individuals undergoing motor skill rehabilitation by providing visual feedback for guided hand movements.
- ***Virtual Collaboration and Remote Whiteboarding:*** Enables participants in remote meetings to draw or annotate in real-time, enhancing communication in digital collaboration platforms.

6. BENEFITS

- ***Intuitive and Natural Interaction:*** By utilizing hand gestures, the system provides a more natural user experience compared to traditional input devices, requiring minimal learning curve.
- ***Hardware Independence:*** The use of a standard webcam makes the system accessible and cost-effective, removing the dependency on specialized or expensive hardware.
- ***Real-Time Performance:*** Leveraging efficient computer vision libraries, the system operates in real-time, providing immediate feedback to user input and enhancing the responsiveness of the interaction.
- ***Scalability and Customization:*** *The modular nature of the system allows it to be extended or adapted for various use cases, including integration with VR/AR platforms or AI-assisted drawing.*

7. CONCLUSION

The Virtual Paint Application offers a fresh take on how we interact with technology, showing how tools like Mediapipe and OpenCV can turn simple hand movements into powerful drawing commands. By tracking hand positions with precision, the system allows users to sketch, edit, and save their artwork—all without ever touching a traditional input device. A live webcam feed, combined with a customizable tools panel and responsive canvas, makes the experience smooth and engaging, even for first-time users.

This project goes beyond just demonstrating what's possible with AI and computer vision—it sets the stage for future growth. Features like recognizing multiple hands, adding more drawing tools, or interpreting complex gestures could make the system even more flexible and user-friendly. At its core, this work explores how natural movements can make digital interactions feel more human, creative, and accessible, moving us closer to a world where technology responds to us more like a collaborator than a machine.

8. REFERENCES

[1] Volume 5, Issue 1, January 2015 ISSN: 2277 128X, International Journal of Advanced Research in Computer Science and Software Engineering: Research Paper -- Gesture Controlled Computer.

[2] P. Viola, M. Jones, "Robust Real-Time Face Detection", International Journal of Computer Vision 57(2), 137154, 2004.

[3] Nidhi, "Image Processing and Object Detection", Dept. of Computer Applications, NIT, Kurukshetra, Haryana, 1(9): 396-399, 2015.

[4] Study on Object Detection using Open CV – Python, International Journal of Computer Applications (0975 – 8887) Volume 162 – No 8, March 2017.

[5] Real Time Object Detection and Tracking Using Deep Learning and OpenCV Proceedings of the International Conference on Inventive Research in Computing Applications (ICIRCA 2018) IEEE Xplore Compliant Part Number: CFP18N67- ART; ISBN:978-1-5386-2456-2.

[6] OpenCV for Computer Vision Applications, Proceedings of National Conference on Big Data and Cloud Computing (NCBDC'15), March 20, 2015.

[7] S Guennouni, A Ahaitouf and A Mansouriss , "Multiple object detection using Open CV on an embedded platform", 2014 Third IEEE International Colloquium in Information Science and Technology (CIST), 2014, pp. 374-377.

[8] G. Chandan, Mohana A.H Jain "The Real Time Object Detection and Tracking Using Deep Learning and OpenCV", 2018 International Conference on Inventive Research in Computing Applications (ICIRCA), 2018, pp. 1305-1308.

[9] Study on Object Detection using Open CV -Python, International Journal of Computer Applications (0975 - 8887) Volume 162 -No 8, March 201.

[10] 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.