

# Gesture-Based AI Virtual Painter for Contactless Drawing and Erasing

1<sup>st</sup> Abhishek Biswas

*Department of Artificial Intelligence*

*IIT Kharagpur*

Kharagpur, India

biswasabhishek521@gmail.com

**Abstract**—This project introduces a contactless virtual drawing tool controlled by hand gestures using the MediaPipe framework for hand tracking. The tool enables intuitive drawing and erasing functionalities without requiring a physical input device, enhancing accessibility for users in various fields. Our experiments demonstrate reliable gesture recognition and responsive performance across varied environments, with potential applications in digital art, education, and virtual reality.

**Index Terms**—Virtual Painter, Gesture Recognition, MediaPipe, Hand Tracking, Digital Art

## I. INTRODUCTION

In recent years, gesture-based interfaces have gained significant attention due to their potential to enhance user interaction in a wide variety of applications, from gaming to artistic creation. These interfaces allow users to interact with digital systems through natural hand gestures, eliminating the need for traditional input devices like keyboards and mice. One area where gesture-based interaction has shown great promise is in virtual drawing applications, where users can create digital art using hand movements as a control mechanism.

The development of hand tracking systems has played a pivotal role in enabling such interaction. Advances in computer vision and machine learning, particularly with the use of deep learning models like MediaPipe, have made it possible to track hand movements in real-time with high accuracy. These systems can detect and recognize hand landmarks, such as the positions of the fingers and palm, allowing for precise interpretation of gestures.

This paper presents a real-time hand tracking and gesture recognition system designed to facilitate a virtual drawing experience. The system enables users to draw on a digital canvas using hand movements, with the ability to switch between different drawing modes based on the orientation and positioning of their hands. Specifically, the system detects the user's hand gestures to toggle between "drawing" mode and "selection" mode, where the user can choose tools or colors. The proposed system leverages the MediaPipe hand tracking library to detect hand landmarks, processes these landmarks to determine the orientation of the hand, and interprets the user's intent based on the positions of the fingers.

The main objectives of this system are to provide an intuitive and responsive method for creating digital art, allowing for seamless interaction through gestures. By combining

hand detection, gesture recognition, and real-time drawing capabilities, this approach aims to provide an immersive and interactive painting experience. In this paper, we explore the technical aspects of the system, including the implementation of the hand detection class, the gesture recognition logic, and the integration of the hand tracking system with a virtual drawing canvas. The effectiveness of the system is demonstrated through various use cases, highlighting the potential of gesture-based interaction in creative applications.

## II. RELATED WORK

Gesture recognition systems have been a focal point in human-computer interaction research for decades. Early methods relied on handcrafted features such as contour analysis and edge detection [1], [2]. Recent advancements in deep learning, including convolutional neural networks (CNNs), have significantly improved the accuracy of gesture recognition systems [3].

Frameworks like MediaPipe [4] and OpenPose [5] have revolutionized hand tracking by enabling real-time detection of hand landmarks. These tools have been applied in diverse fields, including virtual reality [6], gaming [7], and healthcare [8]. However, their application in virtual drawing tools remains underexplored.

While gesture recognition has been used in some digital art applications, such as Kinect-based painting systems [9], these often require specialized hardware. In contrast, MediaPipe offers a hardware-agnostic solution that uses only a webcam, making it more accessible.

Digital art software like Adobe Photoshop [10] and CorelDRAW [11] rely on traditional input devices. Gesture-based tools like Google's Quick, Draw! [12] have limited functionalities, focusing on simple sketches rather than full-fledged artistic creation.

Key challenges in gesture-based systems include handling occlusions, varying lighting conditions, and ensuring real-time responsiveness [13]. By leveraging MediaPipe's robust hand-tracking capabilities, this project addresses these challenges and introduces a novel approach to contactless drawing.

### III. METHODOLOGY

#### A. Data Collection and Preprocessing

We utilized MediaPipe's pretrained hand-tracking model, which allows for real-time detection of hand landmarks. MediaPipe's robustness across lighting conditions and angles negates the need for additional data collection, making it ideal for our gesture-based application.

#### B. Hand Tracking and Gesture Recognition

In this subsection, we present the **Hand Detector Class**, which forms the core of our hand tracking system. This class allows for customizable hand detection settings, including the ability to adjust the number of hands to track, detection confidence, and tracking consistency. The hand tracking process begins by converting each video frame into the RGB format, as required by the MediaPipe hand detection model. The model then processes the frame to detect key hand landmarks, such as the wrist, fingers, and joints. Based on these detected landmarks, the system can differentiate between "draw" and "select" modes. In "draw" mode, the system recognizes when the index finger is raised, enabling the user to draw on the screen. In "select" mode, both the index and middle fingers must be raised to allow the user to select colors or tools for drawing. This approach enables seamless interaction, providing users with an intuitive, gesture-based control mechanism.

#### C. Canvas Management

The management of the drawing canvas is a crucial aspect of the virtual painting system. The canvas, denoted as  $C(x, y)$ , is a separate overlay maintained in RGB format to store user drawings independently of the live video feed. This separation ensures that the user's drawings persist across frames while the video feed, represented as  $V(x, y)$ , updates in real time.

Initially, the canvas is a blank image of the same dimensions as the video feed, initialized with zero values:

$$C(x, y) = 0, \quad \forall (x, y) \in \text{Image Dimensions.}$$

The drawing process on the canvas is facilitated by tracking finger gestures and dynamically updating  $C(x, y)$ . For instance, when the system detects a drawing gesture, a line of the selected color  $\mathbf{c}$  and thickness  $t$  is added to the canvas:

$$C(x, y) = \begin{cases} \mathbf{c}, & \text{if } (x, y) \text{ lies on the gesture path,} \\ C(x, y), & \text{otherwise.} \end{cases}$$

Similarly, when the system detects an eraser gesture, the corresponding area on the canvas is cleared by setting the pixel values to zero:

$$C(x, y) = \begin{cases} 0, & \text{if } (x, y) \text{ lies on the eraser path,} \\ C(x, y), & \text{otherwise.} \end{cases}$$

By maintaining the canvas as a separate layer, the system can efficiently store and manipulate user inputs without interfering with the video feed. This approach also enables features like undo functionality or exporting the drawing as a standalone image. The canvas management ensures a smooth

and responsive user experience, where all drawing operations are seamlessly captured and preserved for further blending with the video feed.

#### D. Dynamic Color and Tool Selection

The dynamic color and tool selection process enables seamless interaction between the user and the virtual painting interface. This is achieved by defining specific regions on the screen, referred to as "color zones," that correspond to distinct colors or tools. The user's index finger position, detected in real-time using hand-tracking landmarks,  $\mathbf{P}_{\text{index}}(x, y)$ , serves as the primary selector for these zones. Let the screen be represented as a 2D coordinate space,  $\mathbf{S}(x, y)$ , divided into  $N$  discrete zones,  $\mathbf{Z}_i(x, y)$ , where  $i = 1, 2, \dots, N$ . Each zone is associated with a unique color  $\mathbf{C}_i$  or tool  $\mathbf{T}_i$ .

When  $\mathbf{P}_{\text{index}}(x, y) \in \mathbf{Z}_i(x, y)$ , the corresponding color  $\mathbf{C}_i$  or tool  $\mathbf{T}_i$  is selected. This selection is visually represented by updating the interface overlay with an icon or swatch corresponding to the active choice. Mathematically, the selection process can be expressed as:

$$\text{Selected Item} = \begin{cases} \mathbf{C}_i, & \text{if } \mathbf{P}_{\text{index}}(x, y) \in \mathbf{Z}_i \text{ (color zones),} \\ \mathbf{T}_i, & \text{if } \mathbf{P}_{\text{index}}(x, y) \in \mathbf{Z}_i \text{ (tool zones).} \end{cases}$$

To enhance the user experience, the interface continuously updates the screen with a feedback mechanism, ensuring responsiveness and clarity. The process also supports mode switching, where different gestures, such as raising one or two fingers ( $\mathbf{G}_{\text{gesture}}$ ), toggle between drawing and erasing modes. This is defined as:

$$\text{Mode} = \begin{cases} \text{Draw,} & \text{if } \mathbf{G}_{\text{gesture}} = 1 \text{ finger,} \\ \text{Erase,} & \text{if } \mathbf{G}_{\text{gesture}} = 2 \text{ fingers.} \end{cases}$$

The integration of dynamic color and tool selection with these mathematical foundations ensures a fluid and intuitive interaction, empowering users to switch tools and colors effortlessly while maintaining focus on the creative process. This modular design lays the groundwork for potential future enhancements, such as adding more complex gestures or expanding the number of selectable options dynamically.

#### E. Overlay Blending Process

The overlay blending process involves multiple transformations of the canvas and live video frame. The canvas image  $\mathbf{C}(x, y)$ , in RGB format, is first converted to its grayscale version,  $\mathbf{G}(x, y)$ . This grayscale representation is then used to derive a binary inverse mask,  $\mathbf{I}(x, y)$ , through a thresholding operation where  $T$  is the threshold value. The live video frame,  $\mathbf{V}(x, y)$ , is masked using the inverse mask, producing a masked live video frame  $\mathbf{V}'(x, y)$ . Finally, the updated canvas image is added to the masked video frame, resulting in the final blended output image,  $\mathbf{O}(x, y)$ .

This sequence ensures that new drawings from the canvas seamlessly overlay the live video feed, maintaining visual consistency.

The integration of the drawing canvas with the live video feed involves several mathematical operations:

- 1) **Grayscale Conversion:** The canvas image  $\mathbf{C}$  is converted to grayscale:

$$\mathbf{G}(x, y) = 0.2989 \cdot \mathbf{C}_R(x, y) + 0.5870 \cdot \mathbf{C}_G(x, y) + 0.1140 \cdot \mathbf{C}_B(x, y)$$

- 2) **Binary Thresholding:** A binary inverse mask  $\mathbf{I}(x, y)$  is created:

$$\mathbf{I}(x, y) = \begin{cases} 0, & \text{if } \mathbf{G}(x, y) > T \\ 1, & \text{if } \mathbf{G}(x, y) \leq T \end{cases}$$

- 3) **Masking Live Frame:** The live video frame  $\mathbf{V}(x, y)$  is updated:

$$\mathbf{V}'(x, y) = \mathbf{V}(x, y) \cdot \mathbf{I}(x, y)$$

- 4) **Adding Updated Canvas:** The canvas is overlaid onto the video frame:

$$\mathbf{O}(x, y) = \mathbf{V}'(x, y) + \mathbf{C}(x, y)$$

#### IV. EXPERIMENTAL RESULTS

Our tool was tested in varied environments to evaluate gesture detection and responsiveness. Results indicate successful recognition of drawing and erasing gestures, even under changing lighting and hand positions. The delay in switching modes was minimal, contributing to a smooth and immersive experience for the user. The system maintained a consistent frame rate of 27 frames per second (fps), which ensured real-time performance and smoothness during interaction. The average latency for gesture recognition and mode switching was measured at 30 milliseconds (ms), ensuring that the user's gestures were reflected promptly on the screen.

Visual verification of the tool's smoothness and responsiveness was conducted during live interactions. The system accurately tracked and responded to rapid hand movements without noticeable lag. Transitions between drawing and erasing modes occurred fluidly, with the on-screen action consistently matching the user's gestures. Even with changes in lighting and hand orientation, the tool maintained high accuracy and responsiveness, offering an intuitive and seamless user experience.

#### V. CONCLUSION AND FUTURE WORK

In this project, we developed a contactless virtual painter that uses hand gestures for intuitive control, enhancing interaction possibilities for digital creators. Future improvements could include additional gesture-based effects, image-saving options, and exploring applications in augmented and virtual reality.

#### REFERENCES

- [1] W. T. Freeman and M. Roth, "Orientation histograms for hand gesture recognition," in *Proceedings of IEEE International Workshop on Automatic Face and Gesture Recognition*, 1995, pp. 296–301.
- [2] V. I. Pavlovic, R. Sharma, and T. S. Huang, "Visual interpretation of hand gestures for human-computer interaction: A review," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 7, pp. 677–695, 1997.
- [3] K. Simonyan and A. Zisserman, "Two-stream convolutional networks for action recognition in videos," *Advances in Neural Information Processing Systems*, pp. 568–576, 2014.
- [4] F. Zhang, V. Bazarevsky, A. Vakunov, and G. Sung, "Mediapipe hands: On-device real-time hand tracking with machine learning," *arXiv preprint arXiv:2006.10214*, 2020.
- [5] Z. Cao, T. Hidalgo, T. Simon, S.-E. Wei, and Y. Sheikh, "Openpose: Realtime multi-person 2d pose estimation using part affinity fields," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 1, pp. 172–186, 2019.
- [6] L. Feng and W. Chen, "Mediapipe applications in virtual reality: Hand-tracking case studies," *Virtual Reality and Intelligent Hardware*, vol. 3, no. 4, pp. 334–345, 2021.
- [7] J.-E. Kim and M.-J. Park, "Gesture-based interaction in virtual gaming environments using kinect," *Entertainment Computing*, vol. 30, p. 100296, 2019.
- [8] A. Karthikeyan and R. Kumar, "Gesture recognition in healthcare applications using mediapipe: A review," *Journal of Health Informatics*, vol. 25, pp. 50–60, 2020.
- [9] W. Liao, C. Cao, and H. Sun, "Kinect-based painting and virtual sculpting: Artistic perspectives on gesture interaction," *ACM Transactions on Graphics*, vol. 35, no. 5, p. 150, 2016.
- [10] (2024) Adobe photoshop. Adobe. [Online]. Available: <https://www.adobe.com/products/photoshop.html>
- [11] (2024) Coreldraw graphics suite 2024. Corel. [Online]. Available: <https://www.coreldraw.com>
- [12] (2024) Quick, draw! Google AI. [Online]. Available: <https://quickdraw.withgoogle.com/>
- [13] X. Yuan and J. Wu, "Hybrid models for overcoming occlusion in hand gesture recognition systems," *Pattern Recognition*, vol. 123, p. 108325, 2021.

#### APPENDIX

##### APPENDIX - CONTRIBUTION

- **Abhishek Biswas:** Conceptualization, Methodology, Gesture Recognition, Canvas Management, UI Design and overlay blending, Software, Validation, Visualization, Documentation.