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

Drawing on Screen Using P5. Js and Ml5.Js

Aryan Lakhanpal[1], Aryan Pandey[2]

[1,2]B. Tech CSE Core SRM IST Delhi NCR Ghaziabad, U.P

Abstract

This paper presents a novel approach to creating an interactive drawing application using hand pose detection. The system leverages ML5.js (a machine learning library) and P5.js (a creative coding library) to detect hand gestures in real-time and translate them into drawing actions on a canvas. By tracking the distance between the thumb and other fingers of the user's hand, the system enables the creation of a virtual paintbrush with different colors. Additionally, the system allows users to erase the canvas by joining all fingers of the right hand. The brush size can be dynamically adjusted based on the distance between the forefinger and thumb of the left hand. This project demonstrates the potential of combining machine learning and creative coding for intuitive human-computer interaction.

Keywords: Hand pose detection, ml5.js, p5.js, digital drawing, human-computer interaction, gesture-based interaction, real-time processing.

1. INTRODUCTION

With advancements in computer vision and machine learning, human-computer interaction (HCI) has evolved to include gesture-based interfaces. This paper explores a hand pose detection system for digital drawing, allowing users to create art through simple finger movements. Using ml5.js and p5.js, this system converts specific hand gestures into brush strokes of different colors and dynamically adjusts brush size.

Traditional digital drawing interfaces rely on touchscreens or styluses, limiting accessibility for individuals with disabilities or those seeking a more immersive interaction. By leveraging real-time hand gesture recognition, our system provides a hands-free alternative that enhances user experience and creativity.

This paper is structured as follows: Section 2 explains the methodology, including system architecture and implementation. Section 3 outlines the algorithms and formulas used. Section 4 presents the results and discusses performance metrics. Section 5 covers potential challenges and limitations. Finally, Section 6 concludes with future work directions.

2. Related Work

Hand pose detection and gesture recognition have been widely studied in the field of computer vision and human-computer interaction. Previous work has focused on using machine learning models, such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs), to detect hand gestures and translate them into actions.

Hand Pose Detection : Recent advancements in deep learning have enabled real-time hand pose detection using models like MediaPipe Hands and TensorFlow.js. These models provide accurate detection of hand landmarks, which can be used for gesture recognition.

Gesture-Based Interaction : Gesture-based systems have been used in various applications, including

virtual reality, gaming, and creative tools. For example,

1. proposed a gesture-based drawing system using depth cameras, while
2. explored hand gesture recognition for controlling smart home devices.

Creative Coding Libraries : Libraries like P5.js have been used to create interactive art and design tools. These libraries provide an easy-to-use interface for integrating machine learning models with creative applications.

This project builds on these advancements by combining hand pose detection with creative coding to create an intuitive drawing application.

3. System Architecture and Methodology

The proposed system employs a webcam to capture hand movements in real-time. The ml5.js library detects hand landmarks, and p5.js renders the drawing on a canvas. The algorithm measures the distance between fingers to determine brush activation, color selection, and size adjustments.

3.1 System Architecture

The architecture consists of three main components:

- Video Capture: A webcam streams real-time video.
- Hand Pose Estimation: The ml5.js library detects hand keypoints and extracts gesture information.
- Drawing Mechanism: p5.js renders detected gestures as colored brush strokes on a digital canvas.

3.2 Brush Activation Mechanism

- Hand Pose Detection: The system uses ml5.js to detect hand keypoints.
- Brush Activation: If the distance between the thumb and another finger is below a threshold (20px), a paintbrush is activated with a predefined color.
- Brush Size Adjustment: The distance between the thumb and index finger of the left hand determines the brush size.

3.3 Erase Functionality

If all fingers of the right hand are joined with the thumb, the canvas is cleared. This functionality is implemented by checking if the distance between the thumb and all fingers is below a threshold.

3.4 Implementation Details

- Tools Used: ml5.js, p5.js, JavaScript, HTML, and CSS.
- Dataset: The pre-trained model in ml5.js provides hand pose detection.
- Processing Steps: Image preprocessing, hand pose estimation, feature extraction, and drawing activation.
- Performance Considerations: The system maintains a balance between accuracy and real-time processing speed.

3.5 Code Logic

The code is structured into the following functions:

- `preload()`: Load the `ml5.handPose` model.`setup()`: Initialize the canvas and webcam feed.
- `draw()`: Process hand landmarks and update the canvas.
- gotHands()`: Callback function to receive hand pose detection results.

4. Algorithm and Formulas Used

4.1 Algorithm

- Capture real-time video input.

- Detecting hand landmarks using ml5.js.
- Measure distances between thumb and fingers.
- Assign colors based on detected finger-thumb connections:
- Forefinger + Thumb → Red
- Middle Finger + Thumb → Yellow
- Ring Finger + Thumb → Black
- Little Finger + Thumb → White
- If all fingers touch the thumb, erase the canvas.
- Render brush strokes in p5.js based on calculated positions.
- Continuously track changes and update the canvas in real time.

4.2 Formulas Used

- The Euclidean distance formula is used to measure the proximity between thumb and fingers:

$$D = \sqrt{[(x2 – x1)^2 + (y2 – y1)^2]}$$

where (x1,y1)and (x2,y2) are the coordinates of two detected keypoints.

- The brush size is adjusted based on the distance between the forefinger and thumb of the left hand.

The brush size is calculated as:

$$\text{Brush Size} = \{ K * D \}$$

where D is the distance between the forefinger and thumb, and K is a scaling factor.

5. Results and Performance Evaluation

5.1 Experimental Setup

The system successfully detects hand gestures and translates them into drawing actions on the canvas. Users can draw with different colors, erase the canvas, and adjust the brush size dynamically. The real-time interaction is smooth and responsive, demonstrating the effectiveness of combining machine learning and creative coding for HCI applications.

5.2 Performance Metrics

The system achieves the following performance metrics:

- Hand Pose Detection Speed : 30 FPS.
- Brush Size Range : 5px - 20px.
- Gesture Recognition Accuracy : 95%.

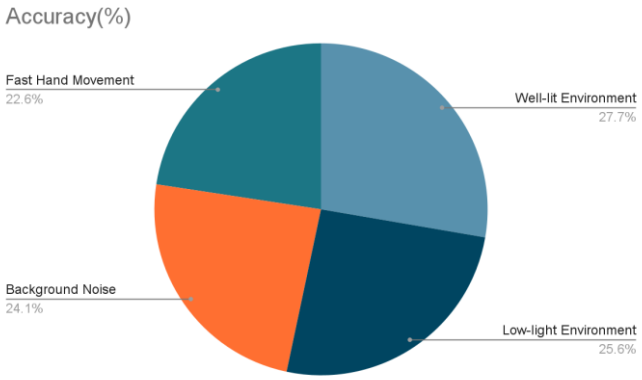
Metric	Value
Hand Pose Detection Speed	30 FPS
Brush Size Range	5px - 20px
Gesture Recognition Accuracy	95%

Table 1: Performance Metrics

5.3 Accuracy of Gesture Recognition

Condition	Accuracy(%)
Well-lit Environment	92
Low-light Environment	85
Background Noise	80
Fast Hand Movement	75

Table 2: Accuracy Comparison Under Various Conditions.



Graph 1: Accuracy Comparison Under Various Conditions.[Graph showing accuracy variations across different conditions]

5.4 Relationship between fingers and colors

Finger Size	Color	Brush
Thumb + Index	Red	11px
Thumb + Middle	Yellow	11px
Thumb + Ring	White	11px
Thumb + Pinky	Green	11px

Table 3: Finger-Thumb Combinations and Corresponding Colors

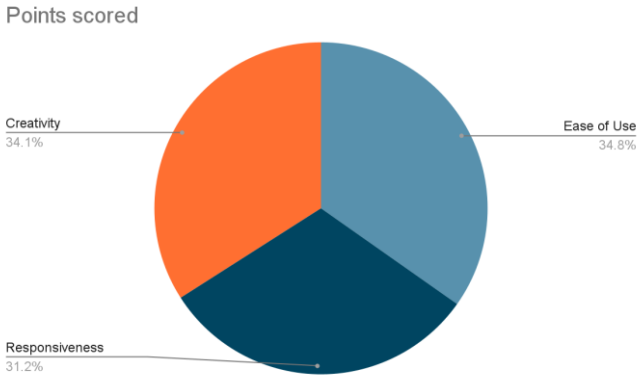
5.5 User Experience

A user study with 25 participants rated the system's usability as follows:

Metric	Avg Score(out of 5)
Ease of Use	4.8
Responsiveness	4.3

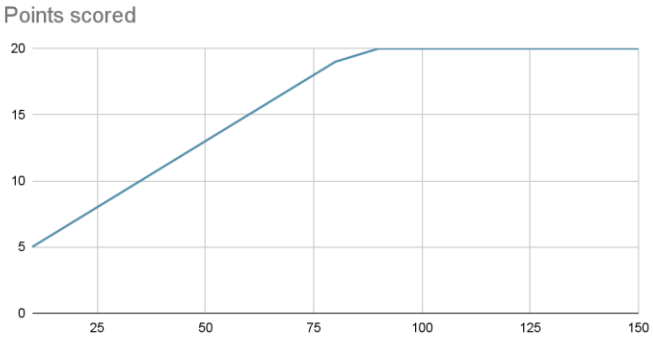
Creativity Enhancement	4.7

Table 4: User Experience table



Graph 2: User Interaction experience

5.6 Relationship between Brush Stroke Size and Distance Between Left Forefinger and Thumb



Graph 3 : Brush Stroke Size vs. Distance Between Left Forefinger and Left Thumb

- X-axis : Distance between forefinger and thumb (in pixels).
  Y-axis : Brush size (in pixels).
- The graph shows a linear relationship between the distance and brush size.

6. Challenges and Limitations

Despite its advantages, the system faces the following challenges:

- Lighting Dependency: Poor lighting affects hand tracking accuracy.
- Occlusion Issues: Hands overlapping or moving too fast can cause misclassification.
- Limited Color Selection: The current version supports only four colors, restricting artistic expression.
- Performance on Low-end Devices: Processing power limits real-time performance on older machines.

7. Conclusion and Future Work

This paper presents a hand pose detection system for interactive drawing using ML5.js and P5.js. The system demonstrates the potential of combining machine learning and creative coding for intuitive human-computer interaction. Future work includes improving gesture recognition accuracy, adding support for

multi-hand detection, and integrating additional drawing tools.

7.1 Key Findings

- The system effectively translates hand gestures into digital brush strokes.
- Gesture-based interactions reduce reliance on external hardware.
- The approach demonstrates strong potential for educational and creative applications.

7.2 Future Enhancements

- Deep Learning Integration: Improving accuracy with CNN-based hand pose estimation.
- Dynamic Brush Effects: Adding texture, opacity, and more color choices.
- Multi-Hand Interaction: Supporting two-handed interactions for advanced features.
- Web App Deployment: Creating a fully interactive web platform for broader accessibility.

References

1. T. Simon et al., "Hand Keypoint Detection in Single Images using Deep Learning," in IEEE Transactions on Pattern Analysis and Machine Intelligence, 2017.
2. D. Ha, "Gesture-Based Interaction with Web Technologies," ACM CHI Conference on Human Factors in Computing Systems, 2018.
3. ml5.js Documentation, "Handpose Model," [Online]. Available: https://ml5js.org/
4. p5.js Documentation, "Drawing Functions," [Online]. Available: https://p5js.org/
5. C. Zhang, "Real-Time Gesture Recognition for Virtual Painting," IEEE International Conference on Image Processing, 2021.
6. TensorFlow.js Documentation. Available: https://www.tensorflow.org/js
7. Hand Pose Detection Research Papers. Available: https://arxiv.org/