Project Report

on

AirCanvas – AI Hand Gesture Drawing Application

Submitted by
Priyanka Chandwani
25MCI10122

Under the guidance of

Dr. Rinku Sharma

in partial fulfilment for the award of the degree of

MASTER OF COMPUTER APPLICATIONS
ARTIFICIAL INTELLIGENCE & MACHINE LEARNING

Chandigarh University

November 2025

CHANDIGARH UNIVERSITY
Discover. Learn. Empower.

NAAC GRADE A+
Accredited University

Certificate

This is to certify that **Priyanka Chandwani**, a student of **Master of Computer Applications (MCA) – Artificial Intelligence & Machine Learning**, has successfully completed the Mini Project titled "**AirCanvas – AI Hand Gesture Drawing Application**" under the esteemed guidance of **Dr. Rinku Sharma, Assistant Professor, University Institute of Computing (UIC), Chandigarh University**.

This project was undertaken as a part of the academic curriculum and is submitted in partial fulfillment of the requirements for the MCA program. The work presented in this project is a result of independent research, diligent effort, and innovation, demonstrating the student's ability to apply artificial intelligence concepts, computer vision, and Python programming to practical problem-solving.

The project, "**AirCanvas – AI Hand Gesture Drawing Application**", is an interactive computer vision-based system that enables users to draw virtually in the air using real-time hand gesture detection and tracking powered by OpenCV and MediaPipe. It effectively showcases the integration of machine learning, image processing, and user interaction design creatively and technically.

I hereby confirm that this project is an original work carried out by the student and has not been submitted elsewhere for the award of any other degree, diploma, or certification.

Project Guide :
Dr. Rinku Sharma
Assistant Professor, University Institute of Computing
Chandigarh University

Acknowledgement

I would like to express my sincere gratitude to **Chandigarh University** and the **University Institute of Computing (UIC)** for providing me with the opportunity to undertake this project, **"AirCanvas – AI Hand Gesture Drawing Application using Python."**

I extend my heartfelt appreciation to my esteemed mentor, **Ms. Rinku Sharma**, Assistant Professor, for her invaluable guidance, continuous support, and insightful feedback throughout the development of this project. Her expertise in **Artificial Intelligence, Machine Learning, and Python Programming** played a crucial role in the successful completion of this work.

I am also deeply thankful to my friends and peers for their constructive discussions and constant encouragement, which helped refine my ideas and implementation approach. Finally, I express my gratitude to my family for their unwavering support, motivation, and patience during the entire duration of this project.

This project has been an enriching learning experience, allowing me to practically explore the integration of **computer vision, artificial intelligence, and human-computer interaction**, and I hope it serves as a strong foundation for future innovation and research in this domain.

Priyanka Chandwani

MCA – Artificial Intelligence & Machine Learning

Chandigarh University

ontents

1. Introduction

In today's era of touchless technology and human–computer interaction, the **AirCanvas** project presents an innovative way to draw, write, and interact with computers using simple **hand gestures** — without the need for physical tools like a mouse or stylus. This mini project, titled **"AirCanvas – AI Hand Gesture Drawing Application using Python,"** utilizes real-time computer vision and artificial intelligence to track hand movements and allow users to draw virtually in the air.

The system leverages the **MediaPipe** framework for accurate hand landmark detection and **OpenCV** for video stream processing and visualization. Using these technologies, the application identifies fingertip coordinates and maps them onto a digital canvas, enabling users to create sketches, patterns, or write text with intuitive gestures. It also includes functionality to clear the screen, change drawing color, and save the output image, offering a fully interactive and engaging user experience.

The project aligns with the **United Nations Sustainable Development Goal (SDG) 9: Industry, Innovation, and Infrastructure**, which emphasizes the importance of technological advancement and innovation. By promoting a touchless, AI-powered interaction model, AirCanvas encourages creativity, accessibility, and digital innovation. It also supports **SDG 4: Quality Education**, as it can serve as an educational tool to demonstrate the integration of computer vision and AI in real-world applications.

In essence, **AirCanvas** transforms computer vision and AI concepts into a practical, engaging experience. It enhances technical understanding of **image processing, gesture recognition, and human-computer interaction**, while showcasing how artificial intelligence can simplify digital creativity and inspire innovative learning experiences.

2. Objective

1. **To develop a real-time virtual drawing system:**
   The project aims to create a touchless drawing platform where users can draw or write in the air using hand gestures, eliminating the need for traditional input devices like a mouse or stylus.

2. **To implement advanced hand gesture detection and tracking:**
   By integrating MediaPipe and OpenCV, the system accurately detects and tracks hand landmarks, particularly the index fingertip, to enable smooth and responsive drawing movements.

3. **To design an interactive graphical user interface (GUI):**
   Using Tkinter, the project provides an intuitive and user-friendly interface that includes options to select colors, clear the canvas, save drawings, and exit the application.

4. **To apply computer vision and artificial intelligence techniques:**
   The project demonstrates how AI and image processing can be combined to create real-time, gesture-based interactive applications.

5. **To enhance creativity through contactless interaction:**
   The system promotes innovative digital expression by allowing users to draw without physical contact, supporting hygienic and futuristic interaction methods.

3. Tools and Technologies Used

- **OpenCV (cv2):** Used for real-time computer vision tasks such as capturing video frames, image processing, and blending effects.

- **MediaPipe:** A powerful framework developed by Google for real-time hand tracking and landmark detection.

- **NumPy:** Utilized for numerical operations and handling arrays efficiently within image processing tasks.

- **Tkinter:** Used to design the graphical user interface (GUI) of the application, including buttons, frames, and display windows.

- **Pillow (PIL):** Employed to convert OpenCV images into a format compatible with Tkinter for display purposes.

4. Implementation Steps

Step 1: Setting Up the Environment

- Installed **Python 3.8** and required libraries including **OpenCV**, **MediaPipe**, **NumPy**, **Pillow**, and **Tkinter**.
- Created a **virtual environment** to isolate project dependencies.
- Verified webcam access and tested basic frame capture using OpenCV.

Step 2: Capturing Real-Time Video

- Initialized the webcam using `cv2.VideoCapture(0)` to capture live video frames.
- Applied **frame flipping** using `cv2.flip()` to provide a mirror effect for intuitive hand movement.
- Converted the frames from **BGR to RGB** for processing by MediaPipe.

Step 3: Hand Detection and Landmark Tracking

- Integrated **MediaPipe Hands** to detect and track 21 landmarks on the hand.
- Extracted coordinates of the **index fingertip** and **middle fingertip**.
- Used the distance between these points to identify gestures:
  - Fingers **apart** → Drawing mode.
  - Fingers **together** → Pause drawing.

Step 4: Drawing Mechanism

- Created a blank canvas using NumPy:

```
self.canvas = np.zeros((480, 640, 3), dtype=np.uint8)
```

- Drew lines between consecutive fingertip positions using `cv2.line()` to simulate air drawing.
- Maintained previous fingertip coordinates to ensure smooth and continuous strokes.

Step 5: User Interface Development (Tkinter)

- Designed an interactive **Tkinter window** that includes:
  - A **video frame display** for the live camera feed.
  - Control buttons — **Pick Color**, **Clear Screen**, **Save Artwork**, and **Exit**.
- Used **PIL (Pillow)** to convert OpenCV frames for display within Tkinter.

Step 6: Adding Blur and Visual Effects

- Applied **Gaussian Blur** using OpenCV to give a soft, aesthetic background effect.
- Merged the blurred background with the drawing canvas using `cv2.addWeighted()` for better visibility.

Step 7: Saving the Artwork

- Used Python's **os** and **time** modules to create a "drawings" folder automatically.
- Saved the final blended image (blurred background + drawing) using `cv2.imwrite()` at the directory:
- `D:\MCA\SEM1\project\hand_gesture\drawings`

Step 8: Testing and Refinement

- Tested the system under various lighting conditions and hand distances.
- Optimized hand detection accuracy and GUI responsiveness.
- Ensured proper release of the webcam and window closure when exiting.

5. Code

```
import cv2
import numpy as np
import mediapipe as mp
import tkinter as tk
from tkinter import colorchooser, messagebox
from PIL import Image, ImageTk
import os
import time

class AirDrawApp:
    def __init__(self, root):
        self.root = root
        self.root.title("🖐 Air Draw - Hand Gesture App")
        self.root.geometry("900x700")
        self.root.configure(bg="#1e1e2f")

        self.color = (0, 255, 0)
        self.thickness = 5
        self.prev_point = None
        self.canvas = np.zeros((480, 640, 3), dtype=np.uint8)
        self.display_frame = None

        self.cap = cv2.VideoCapture(0)
        self.mp_hands = mp.solutions.hands
        self.hands = self.mp_hands.Hands(max_num_hands=1, min_detection_confidence=0.7)
        self.mp_draw = mp.solutions.drawing_utils

        tk.Label(root, text="Air Draw using Hand Gestures 🎨",
                 fg="#00ffcc", bg="#1e1e2f", font=("Segoe UI", 18, "bold")).pack(pady=10)

        self.video_label = tk.Label(root, bg="#1e1e2f")
        self.video_label.pack()

        control_frame = tk.Frame(root, bg="#1e1e2f")
        control_frame.pack(pady=15)

        tk.Button(control_frame, text="🎨 Pick Color", command=self.choose_color,
                  bg="#00ffcc", fg="black", font=("Segoe UI", 10, "bold"),
width=12).grid(row=0, column=0, padx=10)
        tk.Button(control_frame, text="🧹 Clear Screen", command=self.clear_canvas,
                  bg="#00ffcc", fg="black", font=("Segoe UI", 10, "bold"),
width=12).grid(row=0, column=1, padx=10)
        tk.Button(control_frame, text="💾 Save Artwork", command=self.save_canvas,
```

```
                  bg="#00ffcc", fg="black", font=("Segoe UI", 10, "bold"),
width=12).grid(row=0, column=2, padx=10)
        tk.Button(control_frame, text="❌ Exit", command=self.close_app,
                  bg="red", fg="white", font=("Segoe UI", 10, "bold"),
width=12).grid(row=0, column=3, padx=10)

        self.update_frame()

    def choose_color(self):
        color_code = colorchooser.askcolor(title="Pick Drawing Color")
        if color_code[0]:
            self.color = tuple(map(int, color_code[0]))

    def clear_canvas(self):
        self.canvas = np.zeros((480, 640, 3), dtype=np.uint8)

    def save_canvas(self):
        if self.display_frame is None:
            messagebox.showwarning("Error", "No frame available to save yet!")
            return

        save_dir = r"D:\MCA\SEM1\project\hand_gesture\drawings"
        os.makedirs(save_dir, exist_ok=True)
        filename = f"art_{int(time.time())}.png"
        path = os.path.join(save_dir, filename)

        img_bgr = cv2.cvtColor(self.display_frame, cv2.COLOR_RGB2BGR)
        cv2.imwrite(path, img_bgr)
        messagebox.showinfo("Saved", f"Your artwork has been saved at:\n{path}")

    def close_app(self):
        self.cap.release()
        cv2.destroyAllWindows()
        self.root.destroy()

    def update_frame(self):
        success, img = self.cap.read()
        if not success:
            self.root.after(10, self.update_frame)
            return

        img = cv2.flip(img, 1)
        img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        results = self.hands.process(img_rgb)

        # Mild background blur
        blurred = cv2.GaussianBlur(img, (11, 11), 0)
```

```
        if results.multi_hand_landmarks:
            for hand_landmarks in results.multi_hand_landmarks:
                lm_list = []
                for id, lm in enumerate(hand_landmarks.landmark):
                    h, w, _ = img.shape
                    lm_list.append((int(lm.x * w), int(lm.y * h)))

                if lm_list:
                    x1, y1 = lm_list[8]   # Index tip
                    x2, y2 = lm_list[12]  # Middle tip
                    dist = np.hypot(x2 - x1, y2 - y1)

                    if dist > 40:
                        if self.prev_point:
                            cv2.line(self.canvas, self.prev_point, (x1, y1), self.color,
self.thickness)
                        self.prev_point = (x1, y1)
                    else:
                        self.prev_point = None

                    self.mp_draw.draw_landmarks(blurred, hand_landmarks,
self.mp_hands.HAND_CONNECTIONS)

        # Blend drawing with blurred background
        mask = self.canvas.astype(bool)
        blurred = cv2.GaussianBlur(blurred, (35, 35), 15)
        blurred[mask] = cv2.addWeighted(blurred, 0.5, self.canvas, 2.0, 0)[mask]

        self.display_frame = cv2.cvtColor(blurred, cv2.COLOR_BGR2RGB)
        img_pil = Image.fromarray(self.display_frame)
        imgtk = ImageTk.PhotoImage(image=img_pil)
        self.video_label.imgtk = imgtk
        self.video_label.configure(image=imgtk)

        self.root.after(10, self.update_frame)

if __name__ == "__main__":
    root = tk.Tk()
    app = AirDrawApp(root)
    root.mainloop()
```

6.Output

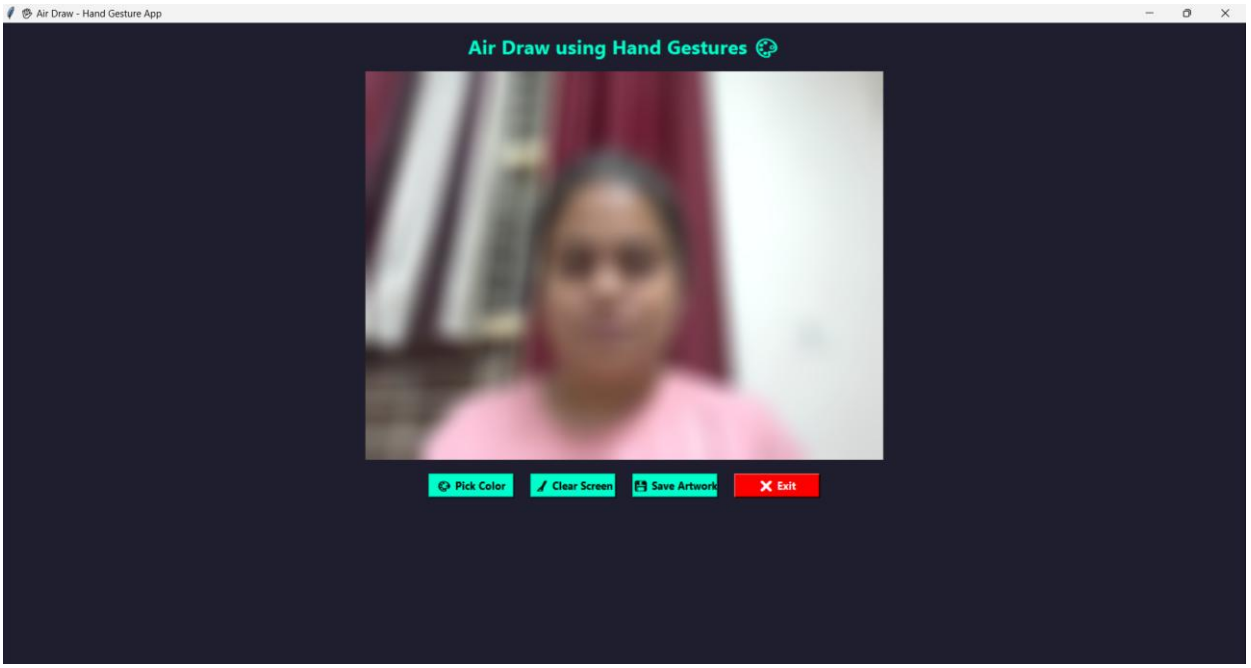

Figure 1: Main Interface of Project

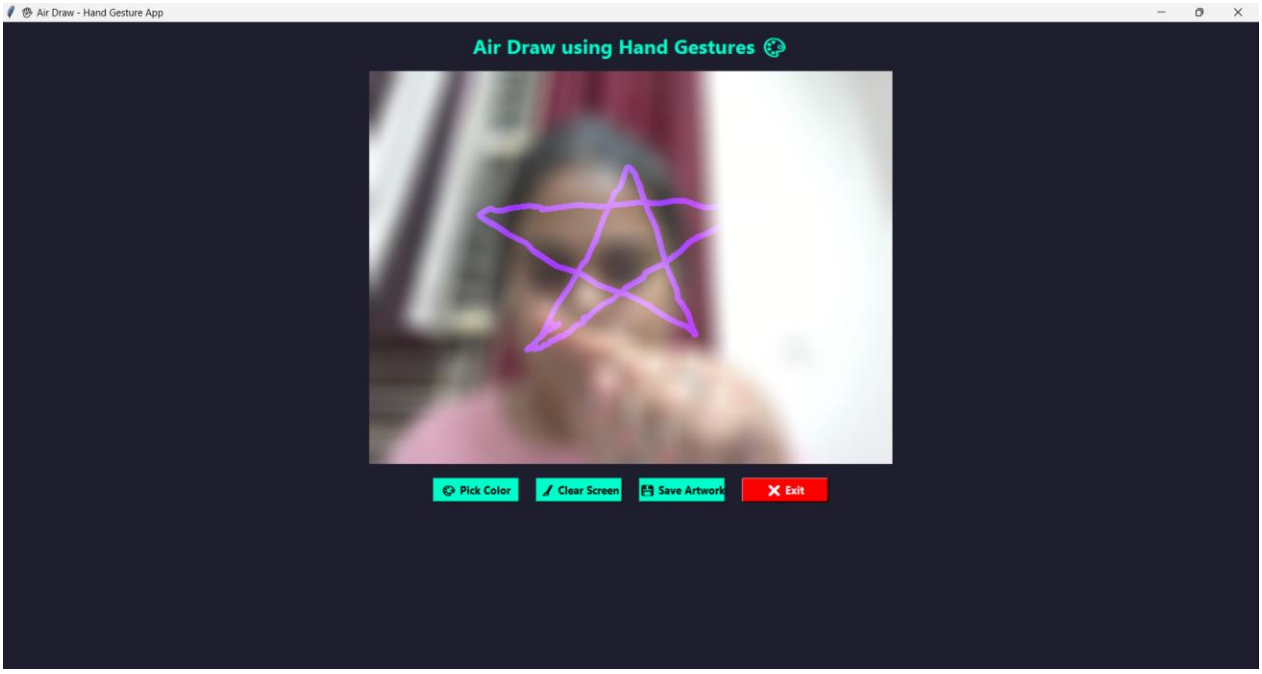

Figure 2: After Drawing using hand gesture

Conclusion

The project **"AirCanvas – AI Hand Gesture Drawing Application using Python"** successfully demonstrates how **computer vision** and **artificial intelligence** can be combined to create an intuitive, touch-free drawing experience. Using **MediaPipe** for real-time hand landmark detection and **OpenCV** for frame processing, the system enables users to draw in the air using natural hand movements without the need for any physical input device.

The integration of **Tkinter** provides an easy-to-use graphical interface that makes interaction seamless and visually appealing. Users can choose colors, clear the screen, and save their artwork conveniently. The project highlights how gesture-based human-computer interaction can enhance creativity, accessibility, and user engagement.

Through this mini project, I have gained practical knowledge in **image processing**, **real-time computer vision**, **GUI design**, and **AI-based gesture recognition**. The system lays the foundation for future improvements such as multi-hand support, gesture-based commands, and advanced shape recognition.

In conclusion, **AirCanvas** is a creative application of artificial intelligence that showcases how technology can make digital interaction more natural and engaging, bridging the gap between human gestures and machine understanding.

References

1. **GitHub link -** https://github.com/priyanka7304/AirCanvas.git
2. **Python Documentation** – https://docs.python.org
3. **NumPy Documentation** – https://numpy.org/doc/
4. **Pandas Guide** –https://pandas.pydata.org/docs/
5. **Matplotlib Documentation** – https://matplotlib.org/stable/contents.html
6. **Tkinter Documentation** – https://docs.python.org/3/library/tkinter.html
7. **OpenCV Documentation** – https://docs.opencv.org/
8. **MediaPipe by Google** – https://developers.google.com/mediapipe/
9. **Pillow (PIL) Documentation**– https://pillow.readthedocs.io/en/stable/