SYNOPSIS

ON

“AIR CANVAS”

Submitted in

Partial Fulfillment of requirements for the Award of Degree

*of*

Bachelor of Technology

*In*

Computer Science and Engineering

By

**(Project Id: )**

# Divya Laldinani (2201640310048)

# Sajal Srivastava (2201641530146)

# Tanzeelah Farid Idrisi (2201641530184)

# Yaggya Chopra (2201641530197)

Under the supervision of

**Supervisor Name**

**(Designation)**



**Pranveer Singh Institute of Technology**.

 Kanpur - Agra - Delhi National Highway - 19

Bhauti -Kanpur - 209305.

(Affiliated to Dr. A.P.J. Abdul Kalam Technical University)

1. Introduction

The rapid advancement in technology, particularly in the field of computer vision and artificial intelligence, has opened new doors for human-computer interaction. One such advancement is the ability to interact with devices without physical contact, using gestures or motion detection. The Air Canvas project capitalizes on these technologies to create a digital drawing platform that is entirely controlled by hand gestures in real-time.

The Air Canvas system aims to revolutionize how users engage with digital canvases by offering a gesture-controlled drawing experience. This application can cater to a wide range of users, from artists and educators to professionals and students, who can utilize this tool for digital drawing, note-taking, or even brainstorming in presentations without any need for a mouse, stylus, or touchscreen. The project is designed to deliver a seamless, intuitive, and hands-free interface through which users can create digital art and write notes.

2. Project Objective

The primary objective of the Air Canvas project is to develop a real-time, gesture-controlled drawing platform that allows users to interact with a virtual canvas using only their hands, captured via a standard webcam. The system should:

* Detect hand gestures and convert them into drawing instructions.
* Offer core drawing functionalities such as brush selection, erasing, and color selection, all controlled by hand gestures.
* Achieve real-time gesture recognition with minimal latency, ensuring smooth and fluid user interaction.
* Provide a user-friendly interface that is accessible and intuitive, ensuring that users can interact with the platform easily without requiring technical expertise.

The system is envisioned to be highly responsive and functional in various environments, from classrooms to home offices and even public exhibits, where interaction with physical devices may be inconvenient. By using affordable and accessible hardware (webcams), Air Canvas also aims to lower the barrier for entry into digital drawing and creative expression.

Additionally, the project explores the potential for integrating artificial intelligence (AI) and machine learning (ML) models to enhance gesture recognition accuracy and expand the range of recognizable gestures. This would allow the system to evolve beyond basic shapes and movements, opening the door to a more diverse range of commands and controls, potentially incorporating more sophisticated hand and finger movements.

This should give a clear picture of the project. Objectives should

be clearly specified. What the project ends up to and in what way

this is going to help the end user has to be mentioned.

1. Feasibility Study:

Before starting any project, it is essential to determine its feasibility across several dimensions. These include technical, operational, economic, and legal feasibility. The Air Canvas project has undergone such an analysis to ensure its practicality and success.

##### ****3.1 Technical Feasibility****

The technical feasibility of the Air Canvas project is strong due to the availability of mature, open-source technologies. Python, the programming language used, offers robust libraries like OpenCV for computer vision and image processing tasks. OpenCV’s capabilities in handling real-time video input and gesture recognition make it ideal for this project.

The system uses a standard webcam to capture hand movements, and OpenCV algorithms are employed to detect specific gestures. This approach is cost-effective, leveraging hardware that is ubiquitous and does not require specialized devices. The processing power required for the application to run is also manageable by most modern computers, including laptops and desktops with basic specifications.

In addition, should the project evolve, it could integrate more sophisticated algorithms for gesture recognition. By incorporating machine learning models, the system could learn to recognize more complex gestures, thereby enhancing its accuracy and responsiveness.

##### ****3.2 Operational Feasibility****

The Air Canvas system is operationally feasible because it requires minimal setup and maintenance. The user needs only a standard webcam and the software application, which can be easily installed and run on any modern computer system. The interface is simple, and the gestures required for controlling the canvas are intuitive, ensuring a low learning curve.

Furthermore, the project is designed to be versatile across various fields—education, arts, presentations, etc.—and thus has multiple potential applications. Its use in classrooms, for example, could revolutionize how teachers interact with students by providing a contactless means of drawing or explaining concepts. The operational feasibility is also supported by the fact that the system does not require constant internet access, as it processes everything locally.

##### ****3.3 Economic Feasibility****

Economically, the project is highly feasible. All of the core technologies used (OpenCV, Python) are open-source, meaning there are no licensing fees involved. The development costs are primarily tied to the time and effort of the developers, which makes the project affordable to implement, especially for educational institutions or individuals.

The hardware requirements are minimal, and since most modern computers come with built-in webcams, additional costs for hardware are negligible. This allows for the widespread adoption of the system without significant financial barriers.

##### ****3.4 Legal Feasibility****

From a legal perspective, there are no major concerns. All software used is open-source and adheres to permissive licensing agreements. Privacy concerns related to the use of webcams are mitigated by the fact that the application does not store or transmit any video data. All image processing is done locally, ensuring compliance with data privacy laws such as the General Data Protection Regulation (GDPR).

**Start Date: 09-Oct-2024 End Date: 20-Dec-2024.**



4. Methodology/ Planning of work

The development of the Air Canvas project follows a structured methodology to ensure systematic progress from concept to final implementation.

4.1 Requirements Gathering

The first step in the development process was to gather the core requirements of the system. This involved identifying the essential features, such as:

* Real-time hand detection.
* Canvas drawing functionality.
* Gesture-based color and brush size selection.
* Erasing and clearing the canvas.

The requirements were gathered based on use cases such as educational applications, creative tools, and brainstorming during presentations.

4.2 Design

Once the requirements were defined, the system’s architecture was designed. The design consists of two primary modules:

* Hand Gesture Detection: This module uses OpenCV to process the video input from the webcam, identifying hand gestures based on color segmentation and contour detection techniques. The hand or an object (such as a colored pen cap) acts as the input to control the canvas.
* Canvas Interaction: This module maps the detected gestures to actions on the canvas. For instance, moving the hand across the canvas draws a line, and specific gestures, such as closing the fist, switch between drawing and erasing modes.

The system is built to be modular so that additional features like machine learning-based gesture recognition can be added later.

4.3 Implementation

The system is implemented using Python and OpenCV, with the following workflow:

* Gesture Recognition: The webcam captures real-time video, and color detection algorithms isolate the hand from the background. Contour detection helps identify the exact position of the hand, which is then mapped to a coordinate on the canvas.
* Drawing Mechanism: The identified hand movements are translated into lines or brush strokes on the virtual canvas. The user’s hand effectively acts as a virtual pen or brush, depending on the chosen mode.
* Additional Features: Other gestures control actions such as switching colors, changing the brush size, or clearing the entire canvas. These are implemented using simple hand positions that are easy to detect reliably.

4.4 Testing

The testing phase involves both unit testing of individual modules (e.g., hand detection accuracy) and integration testing to ensure the system operates smoothly in real-time. User feedback is also incorporated during this phase to fine-tune the system’s performance and usability.

4.5 User Feedback and Iteration

After testing, feedback from users (targeted in educational and creative fields) is used to refine the system. This includes enhancing gesture detection, improving the user interface, and addressing any lag or performance issues.

5. Tools/Technology Used:

5.1 Minimum Hardware Requirements

To run the Air Canvas system smoothly, the following hardware specifications are recommended:

1. Processor:
   * Minimum: Intel Core i3 (or equivalent)
   * Recommended: Intel Core i5 or higher (or AMD Ryzen equivalent)
   * Reason: Gesture recognition and image processing require moderate processing power to analyze video frames in real-time.
2. RAM:
   * Minimum: 4 GB
   * Recommended: 8 GB or higher
   * Reason: Processing video streams and real-time gesture recognition tasks benefit from higher memory to prevent lags.
3. Webcam:
   * Minimum: 720p resolution
   * Recommended: 1080p resolution or higher
   * Reason: A clear video feed ensures better gesture detection and accuracy in hand movement recognition.
4. Graphics:
   * Minimum: Integrated GPU (Intel HD Graphics)
   * Recommended: Dedicated GPU (NVIDIA/AMD)
   * Reason: While not mandatory, having a dedicated GPU can help speed up image processing and ensure a smoother experience.
5. Storage:
   * Minimum: 1 GB of free space
   * Reason: The project itself does not require much storage, but additional space will be needed for Python and relevant libraries.
6. Operating System:
   * Windows 10 or higher, macOS, or Linux (Ubuntu recommended)
   * Reason: Python and OpenCV are cross-platform, making it easy to run the application on any major operating system.

5.2 Minimum Software Requirements

1. Python Environment:

* Python Version: 3.7 or higher
* Reason: OpenCV and other necessary libraries are compatible with Python 3.7+.

2. Python Libraries:

* OpenCV: For image and video processing.
* NumPy: For efficient numerical computations.
* Tkinter (or PyQt for a more advanced GUI): For building the graphical interface.
* Pip: Required to install Python packages.
* MediaPipe (Optional): For enhanced hand gesture tracking if additional precision is required.

3. IDE/Code Editor:

* Visual Studio Code, PyCharm, or Jupyter Notebook for developing and testing the application.
* Reason: These IDEs provide powerful debugging and real-time testing capabilities.

4. Operating System:

* Compatible with Windows 10 and above, macOS, or Linux (Ubuntu 18.04+ recommended for easier installation of dependencies).

5. Additional Software:

* Anaconda (Optional): A Python distribution that simplifies the management of libraries and environments, particularly useful for machine learning development.
* Git: For version control and collaboration.

6. References: [IEEE format]:

1. Bradski, G., & Kaehler, A. (2008). *Learning OpenCV: Computer Vision with the OpenCV Library*. O'Reilly Media, Inc.

1. Shapiro, L. G., & Stockman, G. C. (2001). *Computer Vision*. Prentice Hall.
2. OpenCV. (n.d.). OpenCV documentation. Retrieved from <https://opencv.org>
3. Python Software Foundation. (n.d.). *Python Language Reference, version 3.8*. Retrieved from <https://www.python.org>
4. NumPy. (n.d.). NumPy Documentation. Retrieved from <https://numpy.org>