A Major Project Report   
on  **AIR CANVAS USING COMPUTER VISION AND MEDIAPIPE**

Submitted in partial fulfilment of the requirements for the award of the degree   
of

**BACHELOR OF TECHNOLOGY**

in **COMPUTER SCIENC AND ENGINEERING**By

**TAKKADA ABHINAV 20EG105149**

**THUMMALA MOUNIKA 20EG105152**

**VANNEKALA CHARAN TEJA 20EG105154**

**KOLTHURU ADITYA 20EG105158**

Under the guidance of

**Mr. D. RAMANA KUMAR**

Assistant Professor

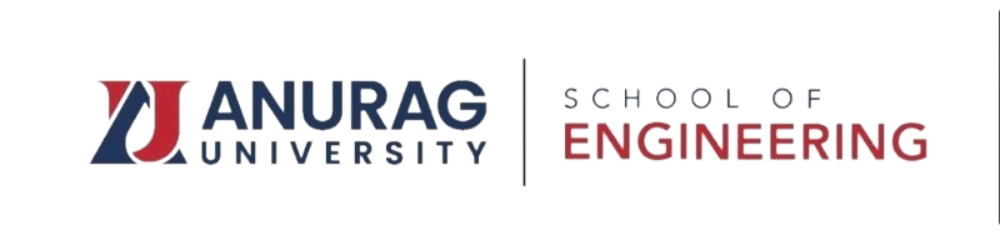
Department of CSE



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**Venkatapur(V), Ghatkeshar(M), Medchal(D)-500088**

**Academic Year 2023-24**



**CERTIFICATE**

This is to certify that the Report entitled **“AIR CANVAS USING COMPUTER VISION AND MEDIAPIPE”** that is being submitted by Takkada Abhinav (20EG105149), Thummala Mounika (20EG105152), Vannekala Charan Teja (20EG105154), Kolthuru Aditya (20EG105158) in partial fulfillment for the award of B.Tech. in **COMPUTER SCIENCE AND ENGINEERING** to the Anurag University is a record of bonafide work carried out by them under my guidance and supervision.

The results presented in this report have been verified and found to be  
 satisfactory. This result embodied in this Report have not been submitted to any other University or Institute for the award of any degree or diploma.

**Mr. D. Ramana Kumar Dr. G. Vishnu Murthy**

**Assistant Professor, Dept. of CSE Professor & Dean, Dept. of CSE**

**External Examiner**

**ACKNOWLEDGMENT**

We would like to express our sincere thanks and deep sense of gratitude to  
project supervisor **Mr. D. Ramana Kumar** for his constant encouragement and inspiring guidance without which this project could not have been completed. His critical reviews and constructive comments improved our grasp of the subject and steered to the fruitful completion of the work. His patience, guidance and encouragement made this project possible.

We would like to express our special thanks to **Dr. V. Vijaya Kumar**, Dean School of Engineering, Anurag University, for his encouragement and timely support in our B. Tech program.

We would like acknowledge our sincere gratitude for the support extended by

**Dr. G. Vishnu Murthy**, Dean, Dept. of CSE, Anurag University. We also express our deep sense of gratitude to **Dr. V V S S S Balaram** , Academic coordinator, **Dr. Pallam Ravi**, Project in-Charge, **Dr. G. Prabhakar Raju**. Project Co-Ordinator and Project review committee members, whose research expertise and commitment to the highest standards continuously motivated us during the crucial stage of our project work.

**TAKKADA ABHINAV**

**(20EG105149)**

**THUMMALA MOUNIKA  
 (20EG105152)**

**VANNEKALA CHARAN TEJA  
 (20EG105154)  
  
 KOLTHURU ADITYA  
 (20EG105158)**

**DECLARATION**

We here by declare that the Report entitled “**AIR CANVAS USING COMPUTER VISION AND MEDIAPIPE**” submitted for the award of Bachelor of technology Degree is our original work and the Report has not formed the basis for the award of any degree, diploma, associate ship or fellowship of similar other titles. It has not been submitted to any other University or Institution for the award of any degree or diploma.

**TAKKADA ABHINAV  
 (20EG105149)**

**THUMMALA MOUNIKA  
 (20EG105152)**

**VANNEKALA CHARAN TEJA  
 (20EG105154)**

**KOLTHURU ADITYA  
(20EG105158)**

Place: Hyderabad

**ABSTRACT**

Digital art creation **has become increasingly popular, yet traditional input** like mice and styluses limit the natural expressiveness of traditional mediums. The Air Canvas, a novel system powered by computer vision, bridges this gap. By translating hand gestures into strokes on a virtual canvas, the Air Canvas fosters a more intuitive and immersive art experience. This work explores the functionalities of the Air Canvas, including hand tracking, gesture recognition, and canvas interaction techniques. It details the integration of powerful libraries like OpenCV and MediaPipe to achieve real-time, accurate hand tracking. Beyond artistic creation, the Air Canvas offers potential applications in interactive presentations, allowing users to control elements with hand gestures. This intuitive approach opens doors for future research in gesture-based interaction design and its impact on digital art. The technology has the potential to influence artistic styles, user experience, and the overall creative process. In essence, the Air Canvas represents a step toward a more user-friendly and expressive digital art experience, fostering creativity and innovation.

Top of Form

**Table of Contents**

[**1. INTRODUCTION** 9](#_Toc164420848)

[**2 . LITERATURE SURVEY** 11](#_Toc164420849)

[**3. PROPOSED METHOD** 21](#_Toc164420850)

[3.1 PROBLEM IDENTIFICATION 21](#_Toc164420851)

[3.1.1 Existing Model 21](#_Toc164420852)

[3.2.1 Proposed System Architecture 22](#_Toc164420853)

[3.2 PROPOSED SYSTEM 22](#_Toc164420854)

[3.2.2 Proposed System Illustration 23](#_Toc164420855)

[**3.2.2.1 Hand Tracking and Gesture Recognition:** 23](#_Toc164420856)

[**3.2.2.2 Drawing Interface and Tools:** 23](#_Toc164420857)

[**3.2.2.3 User Interaction and Control:** 24](#_Toc164420858)

[**3.2.2.4 Recording and Saving Functionality:** 24](#_Toc164420859)

[3.3 SYSTEM DESIGN 24](#_Toc164420860)

[3.3.1 UML Diagrams 24](#_Toc164420861)

[**3.3.1.1 Use Case Diagram** 25](#_Toc164420862)

[**3.3.1.2 Sequence Diagram** 26](#_Toc164420863)

[**3.3.1.3 Class Diagram** 27](#_Toc164420864)

[**3.3.1.4 Component Diagram** 29](#_Toc164420865)

[**3.3.1.5 Deployment Diagram** 29](#_Toc164420866)

[**4. IMPLEMENTATION** 31](#_Toc164420867)

[4.1 LIST OF PROGRAM FILES 31](#_Toc164420868)

[4.2 DATASETS 33](#_Toc164420869)

[4.3 OTHER SUPPORT FILES 34](#_Toc164420870)

[**5. EXPERIMENTAL ENVIRONMENT** 36](#_Toc164420871)

[5.1 EXPERIMENTAL SETUP 36](#_Toc164420872)

[**6. RESULTS** 38](#_Toc164420873)

[6.1 HAND GESTURE AND TRACKING FINGER POSITION ACCURACY 38](#_Toc164420874)

[6.2 USER INTERACTION AND TOOL PERFORMANCE EVALUATION 38](#_Toc164420875)

[A. Interface 39](#_Toc164420876)

[B. Hand Land Marks Detection 39](#_Toc164420877)

[C. Drawing a Circle 39](#_Toc164420878)

[D. Drawing a Rectangle 39](#_Toc164420879)

[E. Drawing a Line 39](#_Toc164420880)

[F. Drawing Free Style 39](#_Toc164420881)

[G. Different Pen Sizes 39](#_Toc164420882)

[H. Adjustable Eraser 39](#_Toc164420883)

[I. Saved Screen Recordings 40](#_Toc164420884)

[6.3 TEST RESULTS 40](#_Toc164420885)

[**7.** **CONCLUSION, FINDINGS AND FUTURE SCOPE** 41](#_Toc164420886)

[7.1 CONCLUSION 41](#_Toc164420887)

[7.2 FINDINGS 41](#_Toc164420888)

[7.3 FUTURE SCOPE 42](#_Toc164420889)

[**REFERENCES** 44](#_Toc164420890)

**Table of Figures**

[Figure 3.1 : Proposed Method Architecture 22](#_Toc164420790)

[Figure 3.1 Air Canvas Using Computer Vision and Mediapipe Use Case Diagram 26](#_Toc164420791)

[Figure 3.2 Air Canvas Using Computer Vision and Mediapipe Sequence Diagram 27](#_Toc164420792)

[Figure 3.3 Air Canvas Using Computer Vision and Mediapipe Class Diagram 28](#_Toc164420793)

[Figure 3.4 Air Canvas Using Computer Vision and Mediapipe Component Diagram 29](#_Toc164420794)

[Figure 3.5 Air Canvas Using Computer Vision and Mediapipe Deployment Diagram 30](#_Toc164420795)

[Figure 4.1 : Workflow diagram for Air Canvas Application 35](#_Toc164420796)

[*Figure 6.1 Interface* 39](#_Toc164420797)

[*Figure 6.2 Hand Land Marks Detection* 39](#_Toc164420798)

[*Figure 6.3 Drawing a Circle* 39](#_Toc164420799)

[*Figure 6.4 Drawing a Circle* 39](#_Toc164420800)

[*Figure 6.5 Drawing Line* 39](#_Toc164420801)

[*Figure 6.6 Drawing Free Style* 39](#_Toc164420802)

[*Figure 6.7 Different Pen Sizes* 39](#_Toc164420803)

[*Figure 6.9 Saved Screen Recordings* 40](#_Toc164420804)

**Table of Tables**

[Table 2.1 : Literature Survey 20](#_Toc164420757)

[Table 4.1 Attributes from Extracted Frame 34](#_Toc164420758)

[Table 6.1 : Results Comparison Table 40](#_Toc164420759)

# **1. INTRODUCTION**

In recent years, the intersection of computer vision and human-computer interaction has spurred the development of innovative applications that redefine the way we interact with digital environments [1][2]. Among these, gesture-based interfaces have emerged as a captivating and intuitive means of communication[8]. Our project, titled "Air Canvas using OpenCV and MediaPipe," stands at the forefront of this technological convergence, presenting a novel approach to digital drawing and interaction.

Visualizing the data in order to learn and understand it can be accomplished through a variety of methods, such as the more traditional methods of teaching, which make use of a marker and a white board[9]. As a result of advances in technology, we have shifted from the more traditional classroom setting to one in which lessons are delivered via an online interface. Even in the midst of the pandemic, students have continued to rely on online classes as their primary educational resource. We are able to increase the interaction through the use of interfaces by utilising computer vision[2]. In this section, we develop intelligent user interfaces, which allow us to draw different sketches using a variety of coloured markers for the purpose of making it simple for everyone to comprehend and see data. The user can sketch or write utilising the air canvas that is provided by this report. There is no need to make any touch with the computer; all that is required is for the user to wave their finger in the air. This makes the process easier to understand and carry out for everyone; it can now be done with no difficulty at all.

The Air Canvas leverages the power of OpenCV, a robust computer vision library, and MediaPipe, a versatile framework for hand tracking, to create a dynamic and immersive drawing experience. Unlike traditional input devices[3], our system translates hand gestures into strokes on a virtual canvas, allowing users to draw, sketch, and interact with digital content in an entirely new way.

This report explores into the technical intricacies of our project, exploring the methods employed for hand tracking, gesture recognition, and canvas interaction. We provide insights into the underlying algorithms, programming techniques,   
and the integration of OpenCV and MediaPipe to achieve real-time, accurate   
hand tracking[7][12]. Additionally, we discuss the potential applications of the Air Canvas, ranging from digital art creation to interactive presentations.

As we navigate through the details of our implementation, it becomes evident that the Air Canvas is not just a technical feat but a bridge between the physical and digital realms. The project holds promise for diverse fields, including art, education, and user interface design.

# **2 . LITERATURE SURVEY**

Gesture-based interaction [1][2], enabling users to create drawings through hand gestures. These methods employ modular designs to ensure flexibility and ease of maintenance, albeit potentially sacrificing precision in comparison to traditional tools.

3D virtual-touch system [3] leveraging embedded optical sensors and IR backlighting to precisely ascertain fingertip positions above the display surface. While offering portability and compactness, this method contends with constraints such as limited depth range and susceptibility to occlusion effects.

Vision-Based tracking systems [4][6] for real-time drawing on digital canvases. While Silhouette-Based and Deep Ensemble Network-Based Approach [4] employs deep ensemble networks for precise brush tip tracking, Color Thresholding [6] utilizes OpenCV for HSV parameterization. However, both confront challenges including color thresholding errors and limitations in accuracy compared to direct tracking.

Using technologies like OpenCV, Gesture recognition a method incorporated solid-colored caps for drawing and navigation [5], Enhancing accessibility through webcam-based interaction using OpenCV and MediaPipe [7], albeit facing challenges related to environmental factors and technical implementation.

Integrating OCR for text recognition [10], enriches the system's versatility. Nevertheless, this adds complexity and may potentially impact real-time performance, particularly in scenarios demanding extensive character recognition.

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Method** | **Advantages** | **Disadvantages** |
| 1 | Color Based Tracking.[1] | 1. The system enables hands-free drawing, allowing users to interact with the digital canvas using gestures rather than physical tools.  2. The system's modular design, with distinct components for color tracking, contour detection, and frame processing, offers flexibility and ease of maintenance. It allows developers to update or replace individual modules without affecting the overall functionality of the system. | 1. Drawing in the air using hand gestures may lack the precision and control of traditional drawing tools like pens or styluses. This can result in less accurate or detailed artwork, particularly for complex designs or fine details..  2. The system's reliance on specific hardware components, such as a webcam and display unit, may limit its accessibility and usability. |
| 2 | Color Based Tracking Using OpenCV.[2] | 1. The air canvas system allows users to draw and interact with the digital canvas without the need for physical contact with a device. By simply waving their finger in the air, users can create drawings or write text.  2. The system can be set up with various hardware configurations, including built-in or external web cameras, making it adaptable to different environments and user preferences. | 1. Drawing in the air using finger waving may lack the precision and control of traditional drawing tools, leading to less accurate or detailed artwork. Users may find it challenging to create intricate designs or fine details due to the inherent limitations of the interaction method.  2. Setting up the system and calibrating it for accurate color detection and tracking may require technical expertise and time-consuming adjustments. |
| 3 | 3D Air- Touch System using an Embedded Optical Sensor Array.[3] | 1. The proposed 3D virtual-touch system offers a compact and portable solution for interacting with 3D images on mobile displays. By embedding optical sensors and angular scanning illuminators into the display pixels and edges, respectively, the system can be integrated seamlessly into mobile devices without requiring additional bulky hardware.  2. Unlike camera-based systems, which rely on high-resolution images for 3D positioning, this system operates without the need for a camera. By using embedded optical sensors and IR backlighting, it can accurately determine the 3-axis (x, y, z) position of a fingertip above the display surface, offering a more efficient and reliable interaction method. | 1. The depth range of the system is currently limited to up to 3 cm, which may restrict its applicability for certain interactions requiring greater depth sensitivity.  2.Multi-touch functionality may be limited by occlusion effects, where overlapping fingertips or objects obstruct each other's visibility to the sensors. This can result in inaccuracies or failures in detecting multiple touch points, particularly in scenarios with complex interactions or overlapping gestures. |
| 4 | Shihouette-Based and Deep Ensemble Network-Based Approach.[4] | 1. The deep ensemble network-based approach offers accurate tracking of the brush tip position on the canvas. By utilizing a combination of LSTM Autoencoder and 1-D CNN, the network captures complex relationships between the 3D pose of the brush handle and the 2D brush tip position, allowing for precise estimation during drawing sessions.  2. Unlike the silhouette-based approach, which may require specially aligned frames and cameras, the deep ensemble network approach reduces the complexity of the system setup. | 1. Training a deep ensemble network requires a significant amount of data and computational resources. Collecting and preprocessing data for network training can be time-consuming, and training the network itself may require powerful hardware and substantial training time to achieve optimal performance.  2. Implementing and maintaining a deep ensemble network-based system may introduce complexity, especially for users who are not familiar with deep learning techniques. |
| 5 | Object colour detection using OpenCV. [5] | 1. The Air Doodle application offers a user-friendly interface that allows users to interact with the system using a solid-colored cap for drawing and navigation. This intuitive interaction method enhances user experience and makes the application accessible to a wider audience, including those who may not be familiar with complex input devices or software interfaces.  2. With features such as writing/drawing, color change, screenshot capture, and screen clearing, the Air Doodle application provides users with versatile functionality for creative expression and practical use | 1. The effectiveness of the color detection algorithm and overall performance of the Air Doodle application heavily rely on the surrounding environment, including factors such as lighting conditions and the color and size of the object used for interaction.  2. Implementing features such as color detection, gesture recognition, and real-time interaction requires robust technical solutions and careful consideration of algorithm performance and computational resources. |
| 6 | Vision-Based Hand Tracking System by colour thresholding using OpenCV.[6] | 1. The developed system allows users to draw and erase on a digital canvas in real-time using an object tracked by the vision-based hand tracking system. This enables fluid and natural interaction with the digital drawing environment, enhancing user creativity and productivity.  2. The program offers convenient export options, allowing users to save their drawings as image files or record their drawing process as a video file. This feature facilitates sharing of drawings with others or creating instructional videos for educational purposes. | 1. The color thresholding method used for object tracking may be prone to errors when there are objects in the background with similar colors to the tracked object.  2. While the MediaPipe Hands tracking solution offers an alternative method for hand tracking, the accuracy of handwriting tracing may be limited compared to direct object tracking. |
| 7 | MediaPipe Hand Tracking Algorithm .[7] | The system enhances accessibility for individuals with disabilities, such as those with hearing impairments or difficulties using traditional keyboards. It provides an intuitive and effortless way to write and communicate, allowing users to express themselves more effectively and independently. | The effectiveness of the system may be influenced by environmental factors such as lighting conditions and background clutter. Variations in lighting or the presence of objects with similar colors to the tracked hand could lead to inaccuracies in hand gesture recognition and writing, affecting the system's reliability. |
| 8 | Gesture Analysis and Recognition .[8] | 1. HMMs provide a probabilistic framework for modeling gesture dynamics, enabling uncertainty quantification in gesture recognition.  2. HMMs inherently incorporate DTW, allowing for alignment of gesture sequences with different durations, which is crucial for accurate recognition. | 1. HMMs typically assume a first-order Markov property, which may not fully capture the complexity of some gesture sequences that exhibit higher-order dependencies.  2. The use of Gaussian mixture models (GMMs) for modeling observation probabilities in HMMs may not accurately capture the distribution of gesture features, especially in cases where the distribution is non-Gaussian or multimodal. |
| 9 | Object Tracking using Computer Vision.[9] | Drawing is visible in real-time on the canvas, providing immediate feedback to the user and facilitating iterative drawing processes | 1. The tracking accuracy may be limited, especially for complex or rapid movements, leading to inaccuracies in drawing.  2. The tool may only track one colored object at a time, limiting collaborative or multi-user drawing experiences. |
| 10 | Object Tracking using OpenCV and integration of OCR. [10] | 1. Integration of OCR enables the system to recognize and interpret the drawn characters or shapes, enhancing its versatility and potential applications.  2. The use of edge enhancement and normalization techniques improves the robustness of object localization, making the system less sensitive to noise, lighting variations, and background interference. | 1. The system heavily relies on accurately detecting the colored finger tip, which may be affected by variations in color and lighting conditions, leading to potential tracking errors.  2. Integrating OCR adds complexity to the system, requiring additional computational resources and potentially impacting real-time performance, especially in scenarios with large datasets or complex characters/shapes to recognize. |
| 11 | Hand Tracking and Gesture Recognition using MediaPipe.[11] | 1. By employing gesture recognition techniques, the system can interpret specific hand gestures to trigger various actions, such as starting or stopping drawing, changing colors, or adjusting brush sizes. This intuitive interaction simplifies the user interface and enhances usability.  2. The system architecture facilitates seamless integration with existing technologies, such as webcams or depth-sensing cameras, making it easily deployable in diverse environments without requiring specialized hardware. | 1. The accuracy and reliability of the system may be affected by environmental factors such as lighting conditions, background clutter, or occlusions, which can degrade performance and limit usability in certain scenarios.  2. While the system supports gesture recognition for basic actions such as drawing and erasing, the vocabulary of recognized gestures may be limited, restricting the range of interactions possible with the virtual canvas. Expanding the gesture vocabulary could enhance the system's versatility and usability. |
| 12 | Hand Recognition and tracking using Mediapipe, OpenCV and HSV.[12] | 1. By training the fingertip recognition model with images captured in distinct backgrounds, the system becomes more robust to background variations, ensuring consistent performance across different environments.  2. The proposed algorithm utilizes deep learning techniques to achieve high precision in fingertip detection, enabling accurate tracking of hand movements for air writing or gesture recognition. | 1. The effectiveness of the fingertip recognition model heavily depends on the quality and diversity of the training dataset. Inadequate or biased training data may lead to reduced accuracy and reliability of the system.  2. Despite efforts to mitigate the impact of environmental factors, such as background variation, the system may still be sensitive to certain conditions such as lighting changes or occlusions, affecting its performance in real-world scenarios. |

Table 2.1 : Literature Survey

# **3. PROPOSED METHOD**

## 3.1 PROBLEM IDENTIFICATION

The evolution of digital art creation tools has revolutionized the   
way artists express their creativity. However, conventional digital art applications often present challenges that inhibit the seamless translation of artistic vision into digital form. The primary problem lies in the limitations of existing input methods, such as mouse or stylus interactions, which fail to capture the natural fluidity and expressiveness of traditional artistic mediums[2][3][9].

Digital artists face a disconnect between their creative intentions and the   
constraints of mouse-based interfaces. The rigid nature of mouse movements can impede the flow of artistic expression, leading to a sense of detachment from the creative process. Similarly, stylus-based interfaces, while offering more precision, may still lack the tactile feedback and spontaneity inherent in traditional art mediums.

Furthermore, the learning curve associated with mastering complex software interfaces adds another layer of complexity for users transitioning from traditional to digital art. This barrier to entry can deter aspiring artists from fully exploring their creative potential and may limit the accessibility of digital art tools to a broader audience.

### 3.1.1 Existing Model

In summary, the identified problem revolves around the inadequacy of existing digital art applications[3][5] in providing natural, intuitive, and immersive user experiences. Addressing these limitations requires a paradigm shift towards interactive systems that leverage more intuitive input methods, such as hand gestures[7], to bridge the gap between traditional and digital artistic expression.

Existing digital art applications predominantly rely on   
conventional input methods, such as mouse or stylus interactions, to facilitate the creation of digital artwork. While these applications have paved the way for digital artistry, they exhibit several limitations that hinder the overall user experience and creative process.

Addressing these limitations requires the development of innovative solutions   
that leverage emerging technologies to enhance user interactions and foster creativity[1].

### 3.2.1 Proposed System Architecture

## 3.2 PROPOSED SYSTEM

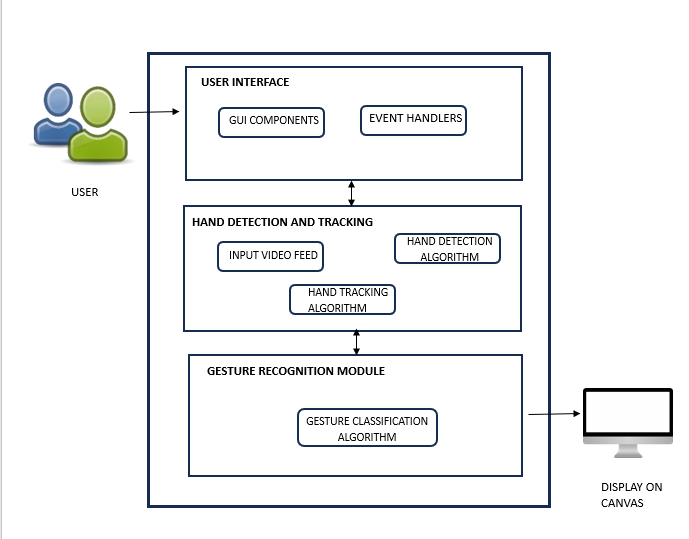


Figure 3.1 : Proposed Method Architecture

This architecture outlines a virtual painting application that leverages hand gestures for user input. The user interacts with the system, likely through a separate user interface (not shown here). A webcam captures video frames containing the user's hand. These frames are fed into a hand detection component that analyzes them to identify the user's hand and potentially extract features like fingertip locations. A finger identification and tracking component takes this information further, refining it to pinpoint specific fingers and track their movements over time. The core functionality lies in the canvas interaction component. It translates these identified finger movements into meaningful actions on the virtual canvas. This could involve drawing lines, changing colors, or other interactions depending on the application's design. Finally, the execution of these actions is reflected on the virtual canvas, providing the user with a visual representation

of their hand gestures and their impact on the painting. In essence, this architecture   
enables a hand-driven virtual painting experience.

### 3.2.2 Proposed System Illustration

The methodology employed in the development of the virtual   
painter application follows a structured approach aimed at leveraging hand gesture recognition technology to facilitate intuitive digital art creation. This methodology encompasses several key steps, each contributing to the design and implementation of a user-friendly and versatile application.

#### **3.2.2.1 Hand Tracking and Gesture Recognition:**

The foundation of the virtual painter application lies in its ability to  
 accurately track the movements of the user's hand and recognize specific gestures in real-time. This functionality is achieved through the utilization of the HandTracker class from the MediaPipe library, which integrates pre-trained machine learning models for hand detection and landmark localization. By analyzing the spatial configuration and temporal sequence of hand landmarks within video frames captured by the camera interface, the application can identify and interpret gestures performed by the user. This enables natural and intuitive interactions between the user and the virtual painter interface, forming the basis for seamless digital art creation.

#### **3.2.2.2 Drawing Interface and Tools:**

Central to the virtual painter application is its drawing interface, which provides users with a digital canvas for expressing their creativity. The interface offers a diverse range of drawing tools, including freehand drawing, lines, circles, rectangles, and erasers, catering to various artistic styles and preferences. Each drawing tool is associated with specific hand gestures or commands, enabling users to switch between tools effortlessly. Moreover, users can adjust parameters such as brush size, color, opacity, and line thickness using gesture-based interactions, enhancing the flexibility and expressiveness of the drawing tools. This comprehensive set of features empowers users to unleash their creativity and explore different artistic techniques within the virtual painter application.

#### **3.2.2.3 User Interaction and Control:**

Efforts are directed towards ensuring natural and intuitive interactions between  
the user and the virtual painter interface. Gesture-based interactions enable users to perform various actions, such as selecting drawing tools, adjusting parameters, navigating the canvas. By mapping specific hand gestures to distinct commands and interactions within the application, users can control and manipulate the drawing interface seamlessly. This intuitive control scheme enhances the user experience and facilitates a fluid and immersive drawing process, allowing users to focus on their artistic expression without being hindered by cumbersome input methods.

#### **3.2.2.4 Recording and Saving Functionality:**

The virtual painter application includes functionalities for recording the   
drawing process and saving artwork, providing users with options for preserving and sharing their creations. Users can initiate the recording feature by performing a specific hand gesture or activating a recording button within the drawing interface. The application captures the entire drawing process as a video, preserving the sequence of strokes and interactions in real-time. Additionally, users have the option to save their artwork as digital images in common formats such as PNG or JPEG. This functionality enhances the utility and value of the virtual painter application, enabling users to document their creative journey and share their creations with others effortlessly.

Overall, the methodology employed in the development of the virtual  
 painter application emphasizes the integration of hand gesture recognition technology with a user-friendly drawing interface, enabling intuitive digital art creation and fostering a supportive and engaging environment for artistic expression. Through iterative refinement and user feedback, the application evolves to meet the evolving needs and preferences of its users, ensuring a seamless and enjoyable drawing experience for all.

## **3.3 SYSTEM DESIGN**

### 3.3.1 UML Diagrams

Unified Modeling Language (UML) diagrams are graphical representations  
widely used in software engineering to model software systems. They provide a   
standardized and visual way to describe various aspects of a system's   
architecture, structure, behavior, and interactions. UML diagrams serve   
as a common language for communication among stakeholders involved   
in software development, including developers, designers, project managers, and clients. These diagrams cover a wide range of perspectives, including architecture, structure, behavior, interactions, and deployment, enabling stakeholders to  
understand and communicate complex software concepts more effectively.

#### **3.3.1.1 Use Case Diagram**

A use case diagram is used to represent the dynamic behavior of a system.   
 It encapsulates the system's functionality by incorporating use cases, actors, and their relationships. It models the tasks, services, and functions required by a system/subsystem of an application. It depicts the high-level functionality of a system and also tells how the user handles a system.

Following are the purposes of a use case diagram given below:

It gathers the system's needs.

It depicts the external view of the system.

It recognizes the internal as well as external factors that influence the system.

It represents the interaction between the actors.

Here's a description of the key aspects of a use case diagram:

**Actors:** Actors represent entities that interact with the system. They could be human users, external systems, or other software components. Actors are typically represented as stick figures or simple shapes outside the system boundary.

**Use Cases:** Use cases represent the specific functionalities or tasks that the system provides to its users. Each use case describes a set of interactions between the system and its actors to achieve a particular goal. Use cases are represented as ovals inside the system boundary and are labeled with descriptive names.

**Relationships:** Relationships between actors and use cases illustrate the interactions between them. The primary relationship in a use case diagram is the association between actors and the use cases they are involved in. This association is depicted by solid lines connecting actors to use cases.

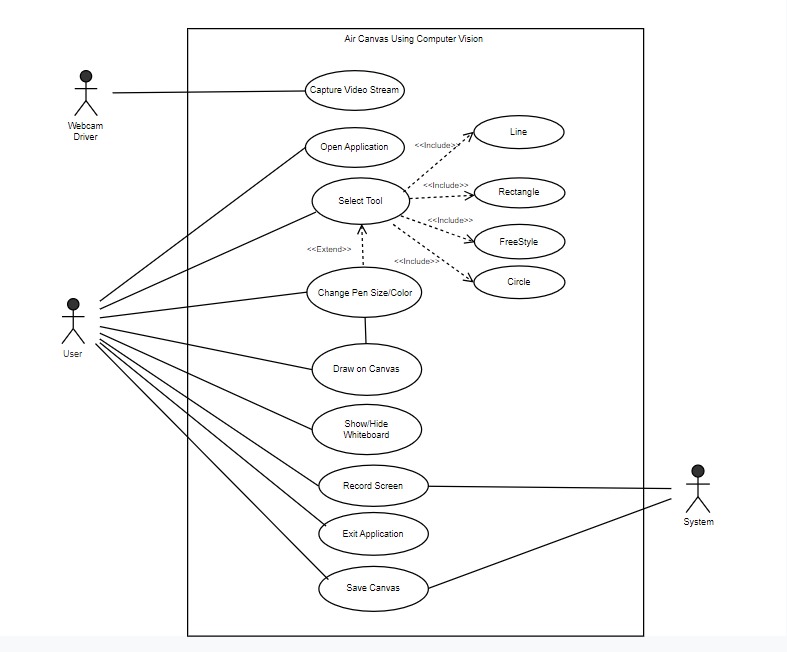


Figure 3.1 Air Canvas Using Computer Vision and Mediapipe Use Case Diagram

#### **3.3.1.2 Sequence Diagram**

Sequence Diagrams are interaction diagrams that detail how operations are   
carried out. They capture the interaction between objects in the context of a collaboration. Sequence Diagrams are time focus and they show the order of the interaction visually by using the vertical axis of the diagram to represent time what messages are sent and when.

Sequence Diagrams captures:

* The interaction that takes place in a collaboration that either realizes a use case or an operation (instance diagrams or generic diagrams)
* high-level interactions between user of the system and the system, between the system and other systems, or between subsystems (sometimes known as system sequence diagrams)

**Purpose of Sequence Diagram**

Model high-level interaction between active objects in a system

Model the interaction between object instances within a collaboration that realizes a use case

Model the interaction between objects within a collaboration that realizes an operation

Either model generic interactions (showing all possible paths through the interaction) or specific instances of a interaction (showing just one path through the interaction)

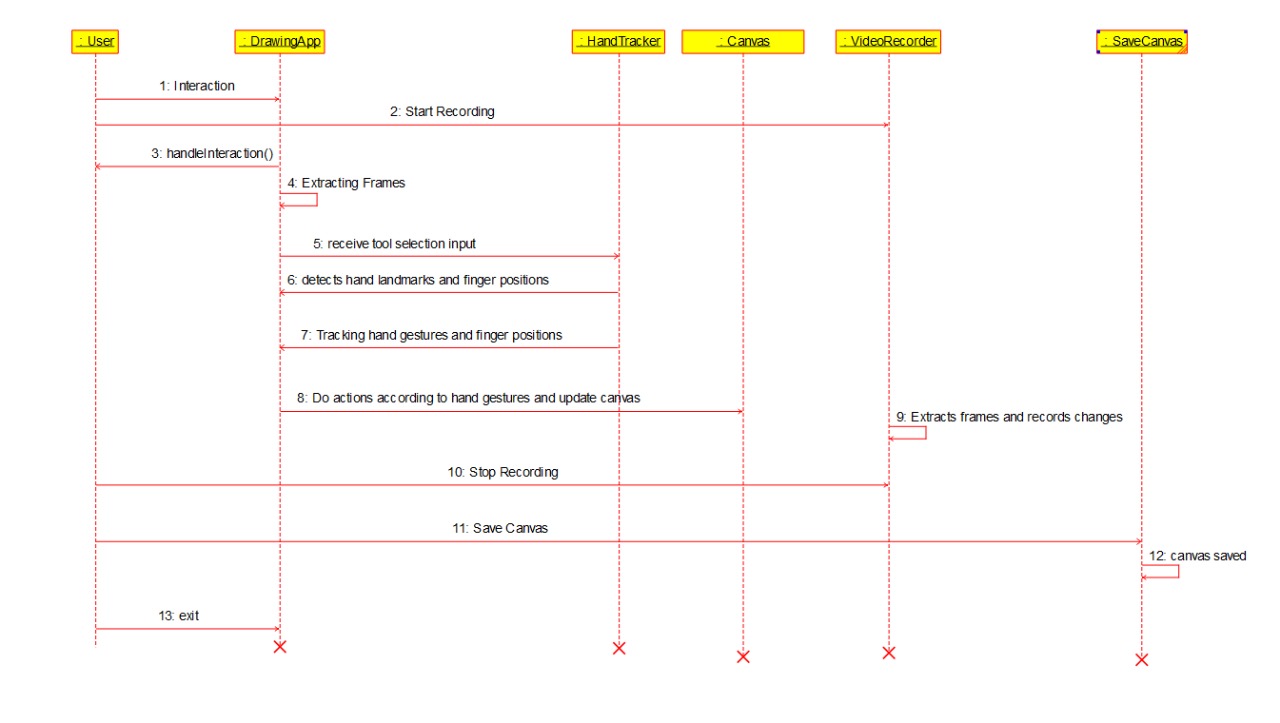


Figure 3.2 Air Canvas Using Computer Vision and Mediapipe Sequence Diagram

#### **3.3.1.3 Class Diagram**

The class diagram depicts a static view of an application. It represents the   
types of objects residing in the system and the relationships between them. A class consists of its objects, and also it may inherit from other classes. A class diagram is used to visualize, describe, document various different aspects of the system, and also construct executable software code.

It shows the attributes, classes, functions, and relationships to give an overview  
 of the software system. It constitutes class names, attributes, and functions in a separate compartment that helps in software development. Since it is a collection of classes, interfaces, associations, collaborations, and constraints, it is termed as a structural diagram.

**Purpose of Class Diagrams**

The main purpose of class diagrams is to build a static view of an application.   
 It is the only diagram that is widely used for construction, and it can be mapped with object-oriented languages. It is one of the most popular UML diagrams. Following are the purpose of class diagrams given below:

It analyses and designs a static view of an application.

It describes the major responsibilities of a system.

It is a base for component and deployment diagrams.

It incorporates forward and reverse engineering.

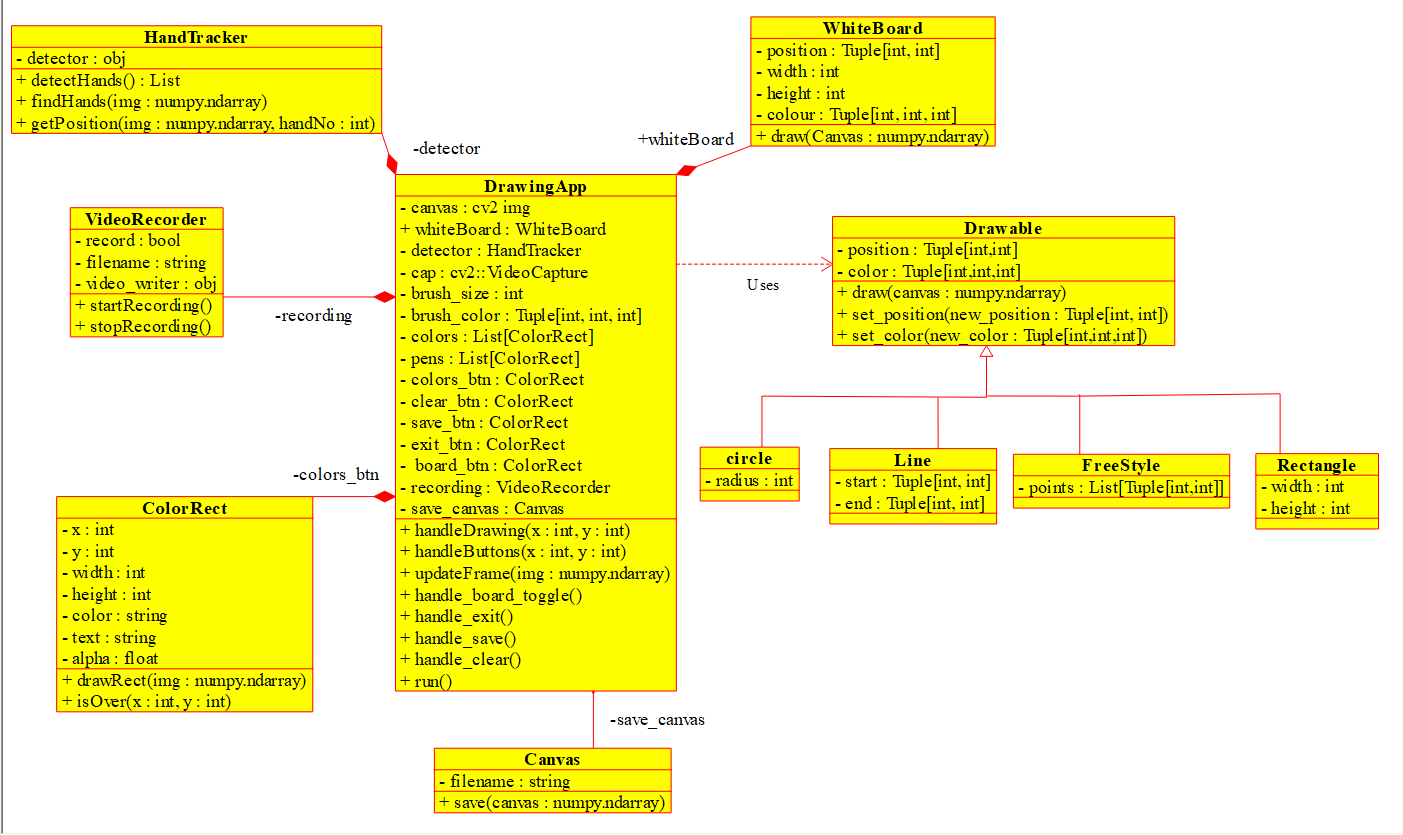


Figure 3.3 Air Canvas Using Computer Vision and Mediapipe Class Diagram

#### **3.3.1.4 Component Diagram**

[UML](https://en.wikipedia.org/wiki/Unified_Modeling_Language) Component diagrams are used in modeling the physical aspects of   
object-oriented systems that are used for visualizing, specifying, and documenting component-based systems and also for constructing executable systems through forward and reverse engineering. Component diagrams are essentially class diagrams that focus on a system's components that often used to model the static implementation view of a system.

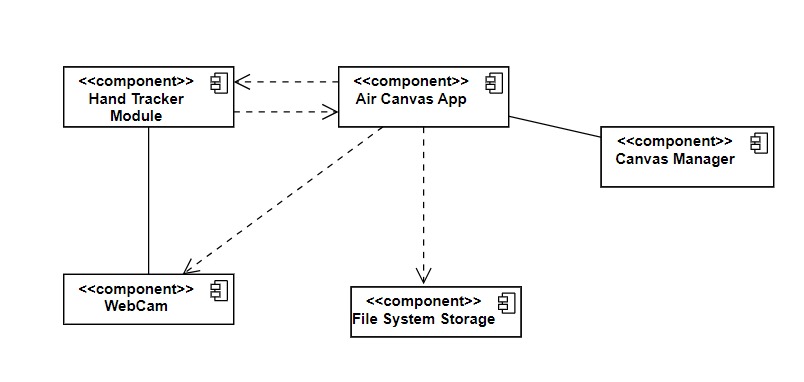


Figure 3.4 Air Canvas Using Computer Vision and Mediapipe Component Diagram

#### **3.3.1.5 Deployment Diagram**

In the context of the Unified Modeling Language (UML), a deployment   
diagram falls under the structural diagramming family because it describes an aspect of the system itself. In this case, the deployment diagram describes the physical deployment of information generated by the software program on hardware components. The information that the software generates is called an artifact. This shouldn't be confused with the use of the term in other modeling approaches like BPMN.

Deployment diagrams are made up of several UML shapes. The   
three-dimensional boxes, known as nodes, represent the basic software or hardware elements, or nodes, in the system. Lines from node to node indicate relationships, and the smaller shapes contained within the boxes represent the software artifacts that are deployed.

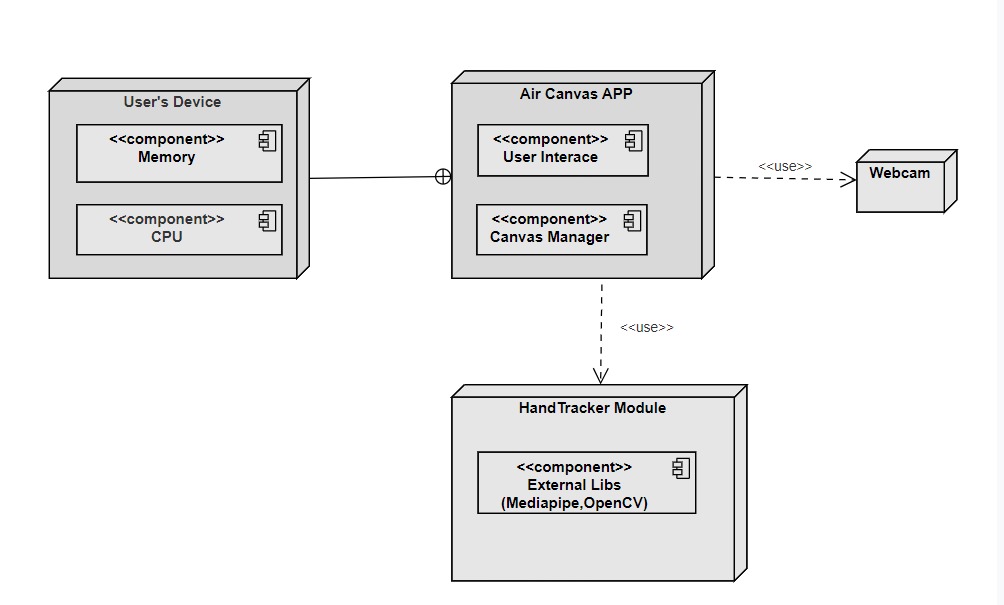


Figure 3.5 Air Canvas Using Computer Vision and Mediapipe Deployment Diagram

# **4. IMPLEMENTATION**

## 4.1 LIST OF PROGRAM FILES

1. **main.py:** This is the main Python script that serves as the entry point for the virtual painter application. It imports necessary modules, initializes key variables, sets up the drawing interface, handles user interactions through hand gestures, and incorporates functionalities for recording and saving artwork.
2. Initialization:

* The script initializes parameters such as mode, maxHands, detectionCon, and trackCon to configure the hand tracking behavior.
* It imports necessary libraries such as cv2 for OpenCV operations and HandTracker class from handTracker.py.

1. Video Capture:

* main.py captures video frames from the camera interface in real-time using OpenCV's VideoCapture module.

1. Hand Tracking:

* Utilizes the HandTracker class to detect and track hand movements in the captured video frames.
* The findHands method of the HandTracker class is invoked to locate hand landmarks and visualize them on the video frames.

1. Gesture Recognition:

* Implements gesture recognition functionalities to enable users to interact with the drawing interface using hand movements.
* The getPostion method of the HandTracker class retrieves the positions of detected hand landmarks.
* Gesture-based interactions, such as selecting drawing tools and adjusting parameters, are triggered based on the detected hand positions.

1. Drawing Interface:

* Provides a drawing interface where users can create artwork using intuitive hand gestures.
* Utilizes OpenCV drawing utilities to render various drawing tools and visual feedback on the video frames.

1. User Interaction:

* Facilitates intuitive user interactions with the drawing interface through gesture-based commands.
* Users can select drawing tools, adjust parameters (e.g., brush size, color), navigate the canvas, and perform undo/redo actions using hand gestures.

1. Recording and Saving Functionality:

* Implements functionalities for recording the drawing process and saving artwork as digital images.
* Users can initiate recording using predefined hand gestures or a dedicated recording button within the drawing interface.

1. **handTracker.py:** This module contains the HandTracker class, which is responsible for hand tracking and gesture recognition using the MediaPipe library. It utilizes pre-trained machine learning models to detect and track hand landmarks in real-time video frames, enabling the recognition of specific gestures performed by the user.

a) Initialization:

* The code initializes a `HandTracker` class object, which likely contains methods for hand tracking using the `mediapipe` library.
* Parameters such as `detectionCon` (detection confidence threshold) and `trackCon` (tracking confidence threshold) are set to control the sensitivity of hand detection and tracking.

b) Frame Processing:

* Inside the main loop of the application (capturing frames from the webcam), the `findHands` method of the `HandTracker` class is called. This method processes the current video frame to detect and track hands.

c) Image Conversion:

* Before passing the frame to the `mediapipe` library, the frame is converted from BGR (OpenCV's default color format) to RGB. This is because the `mediapipe` library expects RGB images.

d) Hand Landmark Detection:

* The `findHands` method processes the RGB image using the `mediapipe` hand module. It returns results, including the hand landmarks if detected.

e) Landmark Visualization:

* If hand landmarks are detected (`self.results.multi\_hand\_landmarks` is not empty), the code uses the `mpDraw.draw\_landmarks` method to visualize the landmarks on the original frame.
* Landmark List Extraction:
* The `HandTracker` class provides a method called `getPostion` that extracts the coordinates of the detected hand landmarks. These coordinates are stored in the `lmList` (landmark list).

f) Up Finger Detection:

* Another method called `getUpFingers` is provided to determine which fingers are raised. It returns a list (`upfingers`) indicating the state of  
  each finger.

## 4.2 DATASETS

Data Source:

The virtual painter application does not rely on external datasets . Instead, it utilizes video frames captured in real-time from the webcam feed during its execution. These video frames serve as the primary source of input data for the application, enabling  
 real-time hand tracking, gesture recognition, and digital art creation.  
Description:

* The dataset comprises a series of video frames, each representing a single frame from the webcam feed.
* Each frame is a 3-channel image with a resolution of 1280x720 pixels.
* The dataset does not include any pre-existing annotations or labels.

|  |  |  |
| --- | --- | --- |
| Attribute Name | Description | Data Type |
| Hand Landmarks | Coordinates of landmarks detected on the hand(s) | List of Tuples |
| Detected Hands Count | Number of hands detected in the frame | Integer |
| Hand Up Fingers | Binary values indicating whether each finger is raised | List of Booleans |
| Hand Detection Confidence | Confidence score for hand detection in the frame | Float |
| Hand Tracking Confidence | Confidence score for hand tracking in the frame | Float |
| Hand Pressure Confidence | Confidence score for hand pressure detection in the frame. | Float |
| Frame Width | Width of the video frame | Integer |
| Frame Height | Height of the video frame | Integer |
| Frame RGB Image | RGB representation of the video frame | 3D NumPy Array |

Table 4.1 Attributes from Extracted Frame

## 4.3 OTHER SUPPORT FILES

**a. tools.png**: This image file contains icons representing different drawing tools (e.g., lines, circles, rectangles) displayed within the virtual painter interface. It is loaded and displayed alongside the drawing canvas to provide users with visual cues for selecting drawing tools.

**b. images/:** This directory is created to store screenshots of the canvas captured by users during the drawing process. Each screenshot is saved as a PNG image file with a unique filename, allowing users to save their artwork for future reference or sharing.

**c. videos/:** This directory is created to store screen recordings of the drawing process captured by users. Each recording is saved as an AVI video file with a unique filename, enabling users to review their drawing process or create time-lapse videos of their artwork.

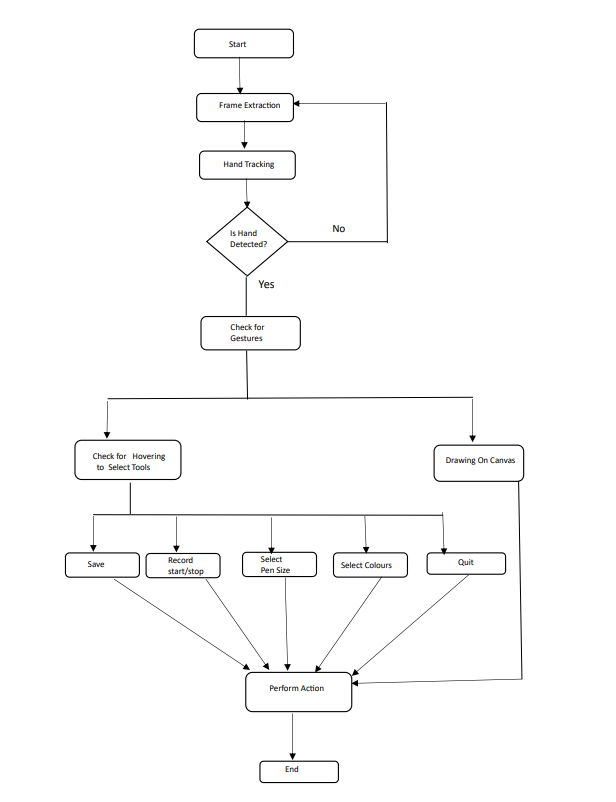
These files collectively form the foundation of the virtual painter application, providing the necessary functionalities for hand tracking, gesture recognition, drawing interface setup, user interaction handling, and artwork recording and saving.

Figure 4.1 : Workflow diagram for Air Canvas Application

# **5. EXPERIMENTAL ENVIRONMENT**

## 5.1 EXPERIMENTAL SETUP

Text Editors/IDEs:

• **Visual Studio:**

Visual Studio is an integrated development environment (IDE) created by Microsoft. It provides a comprehensive set of tools and services for software development, making it easier for developers to create, debug, test, and deploy applications. Visual Studio supports a wide range of programming languages, including C++, C#, Visual Basic, F#, Python, and more.

• **Anaconda :**

Anaconda Navigator includes support for launching various tools and applications, including text editors and IDEs like Jupyter Notebook, JupyterLab, Spyder, and VS Code. These tools can be launched and used directly from Anaconda Navigator, making it convenient for data scientists and developers working in Python.

Package Managers:

• **pip:** pip is the most common package manager for Python. It is used for installing, managing, and updating Python packages from the Python Package Index (PyPI) and other repositories.

• **Conda:** Conda is a package manager and environment management system for installing and managing dependencies in various programming languages, including Python. It is commonly used in data science and scientific computing projects.

**Libraries Used**

• **MediaPipe:**

MediaPipe is an open-source framework developed by Google that provides a set of machine learning (ML) solutions for building applications with perceptual computing and computer vision capabilities. It is designed to make it easier for developers to integrate machine learning models into their projects, especially those involving real-time processing of audio and visual data.

• **Numpy:**

NumPy is a powerful numerical computing library for the Python programming language. It provides support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays. NumPy is a fundamental package for scientific computing in Python and is widely used in fields such as data science, machine learning, signal processing, and more.

• **OpenCV:**

OpenCV (Open Source Computer Vision Library) is an open-source computer vision and machine learning software library. Originally developed by Intel, OpenCV has become a widely used tool in the field of computer vision due to its comprehensive set of functions and algorithms. It is written in C++ and has interfaces for Python, Java, and other languages.

# **6. RESULTS**

## 6.1 HAND GESTURE AND TRACKING FINGER POSITION ACCURACY

In this experiment observations are:

Average Accuracy of Closed Fist: 0.9857098565

Average Accuracy of Open Fist: 0.9800391105

Therefore, the system effectively detects hand gestures and tracks finger positions.

## 6.2 USER INTERACTION AND TOOL PERFORMANCE EVALUATION

User interaction testing is conducted to evaluate the responsiveness of the tools   
to different input methods. Edge cases, involving extreme conditions, are considered to assess the robustness of the drawing tools. Testing is performed for real-time drawing scenarios, and debugging tools are utilized to identify any underlying issues. User feedback is collected to gain insights into the overall user experience. The testing process is iterative, with continuous improvement based on identified issues, ensuring that the drawing tools meet high standards of precision and functionality. After all the continuous improvements and testing we could see that the drawing tools, such as line,   
circle, and rectangle are working accurately.

|  |  |
| --- | --- |
| Interface *Figure 6.1 Interface* | Hand Land Marks Detection *Figure 6.2 Hand Land Marks Detection* |
| Drawing a Circle *Figure 6.3 Drawing a Circle* | Drawing a Rectangle *Figure 6.4 Drawing a Circle* |
| A screenshot of a computer  Description automatically generatedDrawing a Line *Figure 6.5 Drawing Line* | A screen shot of a computer  Description automatically generatedDrawing Free Style *Figure 6.6 Drawing Free Style* |
| 1. A screenshot of a computer     Description automatically generatedDifferent Pen Sizes   *Figure 6.7 Different Pen Sizes* | A screenshot of a computer  Description automatically generatedAdjustable Eraser *Figure 6.8 Adjustable Eraser* |

A screenshot of a computer

Description automatically generated

### Saved Screen Recordings

*Figure 6.9 Saved Screen Recordings*

## 6.3 TEST RESULTS

|  |  |  |  |
| --- | --- | --- | --- |
| **Parameters**  **Methods** | **Accuracy** | **Frames Per Second (FPS)** | **Response Time (milliseconds)** |
| **Colour-based Hand Tracking** [13] | Moderate (83%) | 15 fps | 28 ms |
| **Optical Flow Hand Tracking** [14] | Good (89%) | 25 fps | 18 ms |
| **Depth-based Hand Tracking using stereo cameras** [15] | Moderate (82.93%) | 35 fps | 12 ms |
| **MediaPipe Hand Tracking (Proposed Method)** | Excellent (93%) | 35 fps | Potentially < 10 ms |

Table 6.1 : Results Comparison Table

# **CONCLUSION, FINDINGS AND FUTURE SCOPE**

## 7.1 CONCLUSION

The Air Canvas project brings the freedom of traditional art to the digital world. This innovative system uses computer vision technology to bridge the gap between how artists create physical art and how they create digital art.

Traditional tools like brushes and pencils offer a natural way to express  
 yourself creatively. Unfortunately, using a mouse or stylus for digital art can feel limiting and less natural.

By simply waving your hand in the air, you can create strokes on a virtual   
canvas. This allows artists to express themselves more naturally, just like with traditional tools.

This project explored the technical details behind the Air Canvas. We explained   
how the system tracks hand movements, recognizes gestures, and translates them into strokes on the canvas. Additionally, we showed how powerful libraries like OpenCV and MediaPipe work together to make real-time, accurate hand tracking possible.

The Air Canvas isn't just for artists! This technology has potential applications  
in interactive presentations. Imagine controlling elements on the screen with just your hand gestures.

By offering a more intuitive way to interact with digital tools, the Air Canvas  
 opens exciting possibilities for future research. We can expect this technology to influence artistic styles, user experience, and the overall creative process within digital art.

In essence, the Air Canvas acts as a bridge between the physical world and the digital one. It allows people to create and interact with digital content in a whole new way, paving the way for exciting advancements in the future.

## 7.2 FINDINGS

1. Future Directions: Potential areas for future research and development. This

includes further refining the hand gesture recognition algorithm for improved accuracy, expanding the application's features and capabilities, and exploring opportunities for integrating additional technologies such as augmented reality.

1. More Natural Creation: The Air Canvas allows artists to create art in a more natural way compared to using a mouse or stylus. It feels more like using traditional art tools, making it easier for artists to express themselves creatively.
2. Real-Time Tracking: The Air Canvas uses powerful libraries to track your hand movements very precisely and quickly. This makes the interaction between your hand and the virtual canvas feel smooth and seamless.

## 7.3 FUTURE SCOPE

In the future, artists around the world could team up and work on the same  
giant digital canvas, like a virtual art studio. This would be a game-changer, letting them create amazing things together and spark even more creative ideas.

Furthermore, the Air Canvas could be integrated with virtual reality technology, creating an artistic environment unlike any other. No longer confined to a flat canvas, artists could sculpt breathtaking virtual masterpieces or meticulously paint scenes that envelop them entirely in 3D.

Machine learning offers a powerful tool to personalize the Air Canvas experience. By analyzing an artist's style and past creations, the system could learn to suggest brushstrokes, color palettes, and even complete effects that resonate with the artist's unique vision. This type of AI-powered personalization would empower artists to explore the depths of their creativity and experiment with new techniques and styles.

However, the Air Canvas isn't designed to be exclusive to artists. Future   
iterations could explore alternative interaction methods beyond hand gestures. Voice commands could be incorporated, and new gesture recognition techniques could be developed to cater to users with physical limitations. Additionally, the system could be adapted to accommodate different hand shapes and sizes, ensuring a comfortable and enjoyable experience for everyone.

The potential of the Air Canvas extends far beyond artistic expression.   
Imagine presentations where presenters control elements on the screen with intuitive hand gestures, creating a more dynamic and engaging experience for the audience.

For individuals with speech or motor limitations, the Air Canvas could offer  
 a transformative communication tool. By incorporating eye-tracking technology or alternative gesture recognition, the Air Canvas could empower users to interact with the digital world and express themselves in entirely new ways. This has the potential to significantly improve communication accessibility for people with disabilities.

As the Air Canvas technology continues to evolve, we can expect even   
more groundbreaking features to emerge. The true future of the Air Canvas lies not just in creating stunning artwork, but in making the power of creation accessible and inclusive for everyone.

# **REFERENCES**

[1] Sayil More, Parachi Mhatre, Shruti Pakhare, Surekha Khot.Air Canvas:Draw in Air. International Research Journal of Engineering and Technology (IRJET) Feb 2022.

[2] B Anand ,T.Vinod, ,M Srinivas Rao, Interaction through Computer Vision Air Canvas.2022 IEEE International Conference on Advancements in Smart, Secure and Intelligent Computing (ASSIC) .

[3] Guo-Zhen Wang, Yi-Pai Huang, Tian-Sheeran Chang, and TsuHan Chen, "Bare Finger 3D Air-Touch System Using an Embedded Optical Sensor Array for Mobile Displays", Journal Of Display s

[4] Joolekha Bibi Joolee; Ahsan Raza; Muhammad Abdullah; Seokhee Jeon, Tracking of Flexible Brush Tip on Real Canvas: Silhouette-Based and Deep Ensemble Network-Based Approaches. IEEE Access ( Volume: 8) 22 June 2020.

[5] Soham Pardeshi, Madhuvanti Apar , Chaitanya Khot , Atharv Deshmukh, Air Doodle :“ A Realtime Virtual Drawing Tool”, International Journal for Research in Applied Science & Engineering Technology (IJRASET) Volume 10 Issue IV Apr 2022.

[6] Melvin Cabatuan , Isaiah Jassen Tupal . Vision-Based Hand Tracking System Development for Non-Face-to-Face Interaction. 2021 IEEE 13th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment, and Management (HNICEM).

[7] Shaurya Gulati, Ashish Kumar Rastogi, Raghav Pradhan, Chetan Gupta, Mayank Virmani, Rahul Jana, Paint / Writing Application through WebCam using MediaPipe and OpenCV, 2022 2nd International Conference on Innovative Practices in Technology and Management (ICIPTM).

[8] Vladimir I. Pavlovic, Rajeev Sharma, Thomas 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, JULY 1997.

[9] Gangadhara Rao Kommu, An Efficient Tool For Online Teaching Using OpenCV, International Journal of Creative Research Thoughts(IJCRT) Volume 9, Issue 6 June 2021.

[10] Saira Beg , M. Fahad Khan ,Faisal Baig ,Text Writing in Air. Journal of Information Display Volume 14, Issue 4, 2013.

[11] Hamzathul Favas, Abu Thahir M, Jaseela Thesni N, Mohammed Shahil, Aerosculpt: A Symphony of Creativity on the Air Canvas- Revolutionizing Digital Art and Design through Intuitive Hand Gestures. International Journal Of Advances In Engineering Architecture Science and Technology.

[12] D. Vijendra Kumar, G.Vijaya Raj Siddarth, R. Venkata Satya Sravani, I.Vishnu Vardhan Reddy, Y. Lalitha Sri Naga Durga Vyshnavi, Building a Air Canvas using Numpy and Opencv in Python. 2022 International Journal for Modern Trends in Science and Technology.

[13] K. Imagawa, Shan Lu and S. Igi, "Color-based hands tracking system for sign language recognition," Proceedings Third IEEE International Conference on Automatic Face and Gesture Recognition, Nara, 1998, pp. 462-467.

[14] Kishore, P.V.V. & Anil Kumar, D. & Prasad, M.V.D. & A.S.C.S.Sastry,. (2016). Optical Flow Hand Tracking and Active Contour Hand Shape Features for Continuous Sign Language Recognition with Artificial Neural Networks. 10.1109/IACC.2016.71.

[15] Xianghua Li, Jun-ho An, Jin-hong Min and Kwang-Seok Hong, "Hand gesture recognition by stereo camera using the thinning method," 2011 International Conference on Multimedia Technology, Hangzhou, China, 2011, pp. 3077-30