ST. JOSEPH'S FIRST GRADE COLLEGE

JAYALAKSHMIPURAM, MYSURU-570012

A PROJECT REPORT ON

"Virtual Painting /Air Canvas: Hand Tracking Using OpenCV and MediaPipe"

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

BACHELOR OF COMPUTER APPLICATION
FROM
UNIVERSITY OF MYSORE, MYSURU

Submitted by

Sahil [U01CI21S0088]

Under the Guidance of

Ms. Swathi A
Assistant Professor
Dept. of CS, SJFGC

DEPARTMENT OF COMPUTER APPLICATION
ST. JOSEPH'S FIRST GRADE COLLEGE
JAYALAKSHMIPURAM, MYSURU – 570012
2023-2024

ST. JOSEPH'S FIRST GRADE COLLEGE

JAYALAKSHMIPURAM, MYSURU-570012

CERTIFICATE

This is to certify that the project work entitled **"Virtual Painting/Air Canvas: Hand Tracking Using and MediaPipe"** is bonafide work carried out by **Sahil (U01CI21S0088**) Student Of 6th semester, BCA, in partial fulfillment for the award of the degree in **Bachelor of Computer Application** as prescribed by **University of Mysore**, during the Academic year **2023-2024**. It is certified that all the suggestions and corrections indicated for the Internal Assessment have been incorporated in the report deposited in the department library. The report has approved as it satisfies the requirements in respect of project work prescribed for the said degree.

Ms. Swathi A Assistant Professor Dept. of Computer Science	**Ms. Kanchana S R** Head of the Department Dept. of Computer Science	**Mr. Chandrashekara H N**

External Viva-Voce

Name of the Examiners	**Signature with Date**
1) _____	_____
2) _____	_____

ST. JOSEPH'S FIRST GRADE COLLEGE

JAYALAKSHMIPURAM, MYSURU-570012

DECLARATION

I **Sahil (U01CI21S0088),** student of 6th semester, **BCA, Computer Science, St. Joseph's First Grade College, Mysuru**, hereby declare that the project work entitled **"Virtual Painting /Air Canvas: Hand Tracking Using OpenCV and MediaPipe"** is a bonafide work carried out by me independently under the guidance of **Ms. Swathi A**, Assistant Professor, Department of Computer Science, **University of Mysore**, **Mysuru**. This project work is submitted to, in the partial fulfillment for the award of the degree in **Bachelor of Computer Application** in **Computer Science**, during the academic year **2023-2024**.

Date:

Place:

Sahil
(U01CI21S0088)

ACKNOWLEDGEMENT

If words are considered as the tokens of acknowledgement, then the words play the heralding role of expressing my gratitude.

With proud gratitude we thank God for all the blessings showered and for completion of the project successfully.

I heartily thank my beloved Principal, **Mr. Chandrashekara H N** for his wholehearted support and for his kind permission to undergo the project.

I wish to express my deepest gratitude to **Ms. Kanchana S R**, Head Department of Computer Science, SJFGC, for her constant encouragement and inspiration in taking up this project.

I gracefully thank my guide **Ms. Swathi A**, Assistant Professor, Department of Computer Science, who gave her valuable suggestions throughout the project period and devoting her precious time in making this project a success.

In the end, I offer my sincere thanks to my family members and friends for their valuable suggestions, encouragement and unwavering support.

Sahil [U01CI21S0088]

TABLE OF CONTENTS

Description	Page no
Inner cover page	i
Certificate	ii
Declaration	iii
Acknowledgement	iv
Table of content	v-vi
List of figures	vii
List of Tables	viii
Abstract	ix
Chapter 1 Introduction	**1-4**
1.1 Domain	1
1.2 Overview	1-2
1.3 Objective	2
1.4 Existing System	3
1.5 Proposed System	4
Chapter 2 Literature Survey	**5-8**
2.1 Introduction	5
2.1 Survey Papers	5-8
Chapter 3 System Requirement Specification	**9-14**
3.1 Introduction	9
3.2 Purpose	9
3.3 Functional Requirements	12
3.4 Non-Functional Requirements	13
3.5 Hardware Requirements	14
3.6 Software Requirements	14

Chapter 4 System Design **15-21**

4.1 Introduction 15

4.2 Overview 15

4.3 Data Flow Diagram 16-17

4.4 Activity Diagram 18-19

4.5 Sequence Diagram 20

4.6 Use case Diagram 21

Chapter 5 Implementation **22-28**

5.1 Introduction 22

5.2 Working System Code 22-28

Chapter 6 System Testing **29-32**

6.1 Introduction 29

6.2 Objective of Testing 29

6.3 Testing Methods 30-31

6.4 Test Reports 32

Chapter 7 Snapshots **33-34**

Conclusion **35**

Future Enhancement **36**

References **37**

LIST OF FIGURES

Figure No.	Figure Name	Page No.
3.2.1	Hand Landmarks	11
4.3.1	Data Flow Diagram Level 0	16
4.3.2	Data Flow Diagram Level 1	17
4.4	Activity Diagram	18
4.5	Sequence Diagram	20
4.6	Use case Diagram	21
7.1	Hand Detect	33
7.2	Index Finger Detect	33
7.3	Drawing Board	34
7.4	Pen Sizes	34

LIST OF TABLES

Table No.	Table Name	Page No.
6.4	Test case	32

ABSTRACT

Drawing is fundamental to all other arts. It is how artists structure, plan and negotiate space. Many years back the natural artist used to draw on stone by using charcoal or branches of trees. Cave paintings are used by them to communicate with other about the animals in the forest. From this to now we have many options available for drawing. Drawing can be done by using computer, smartphones even other devices are there for drawing. Now we have many options for drawing. Air canvas is one of the ideas based on drawing in air. It made of system that can catch movements of artists and can draw even without touching keyboard, mouse or touchpad. Air canvas uses python programming language along with useful libraries like OpenCV and MediaPipe which are more helpful in work of identification or recognition.

CHAPTER 1

INTRODUCTION

1.1 Domain

To develop this AI-based project, we will utilize OpenCV and Python, which are essential tools in the field of computer vision and machine learning. OpenCV, an open-source library, provides a vast array of algorithms designed for tasks such as face detection and recognition, object identification, human activity classification in videos, tracking camera movements, monitoring moving objects, and extracting 3D shapes. It is primarily written in C++ but offers extensive Python bindings, making it accessible and flexible for various applications. Python, known for its simplicity and readability, is widely used in artificial intelligence, scientific computing, and data analysis. It supports numerous libraries and frameworks, including TensorFlow, Keras, and PyTorch, which facilitate the development of AI and machine learning models. In this project, we will first capture hand movements using a camera or sensor. These captured frames will be processed in real-time using OpenCV to detect and recognize hand gestures. The recognized gestures will be translated into drawing instructions for the virtual canvas. The processing involves several steps: capturing the video feed, preprocessing the frames to enhance detection accuracy, applying machine learning algorithms to recognize gestures, and updating the virtual canvas accordingly. By leveraging the robust features of OpenCV for image processing and Python's powerful machine learning libraries, we aim to create an efficient and responsive air canvas system for virtual hand-tracking. This combination of technologies will enable the development of a sophisticated human-computer interface that allows users to draw and interact with a digital canvas through hand gestures.

1.1 Overview

In this paper, we introduce a virtual paint application that uses hand gestures for real-time drawing or sketching on the canvas. Hand gesture-based paint application can be implemented using cameras to capture hand movement. To accomplish activities like as tool selection, writing on the canvas, and clearing the canvas, an intangible interface is created and implemented using vision-based real-time dynamic hand gestures. The images of the hands are taken with the system's web camera and processed in real time with a single-shot detector model and media pipe, allowing the machine to communicate with its user in a fraction of a second.

Based on idea of drawing in air the air canvas is made. Air canvas work using function of movements tracking where user must do movements in front of the input devices as like camera. This idea of performing drawing function in front of camera just like waving in air makes air canvas extraordinary way of drawing using AI. System is smart enough to capture and calculated every movement of user in front of it and can perform drawing function. The key requirements of air canvas is only that the user must be on distance where it can be clearly visible to camera so that camera can take proper input. It is very well carried out by using python libraries named as OpenCV and MediaPipe which has ready to used ML solutions for recognition and tracking.

Currently, people and machines interact mostly through direct contact methods such as the mouse, keyboard, remote control, touch screen, and other similar devices, whereas people communicate primarily through natural and intuitive non-contact methods such as sound and physical motions. Many researchers have attempted to help the computer identify other intentions and information using non-contact methods like as voice, facial expressions, physical motions, and gestures. A gesture is the most crucial aspect of mortal language, and gestures also play a significant role in mortal communication.

1.2 Objectives

1. **Accurate Hand Detection:** Develop a robust system capable of accurately detecting and tracking hand movements in real-time using a camera or sensor.

2. **Gesture Recognition:** Implement algorithms to recognize a variety of hand gestures, which will be translated into drawing commands on the virtual canvas**.**

3. **Real-time Processing:** Ensure the system processes video frames and updates the canvas in real- time to provide immediate feedback and interaction for the user.

4. **User-Friendly Interface:** Create an intuitive and responsive user interface that allows users to easily interact with the virtual canvas using natural hand gestures.

5. **Scalability and Flexibility:** Design the system to be scalable, allowing for the addition of more complex gestures and functionalities in the future. Ensure it is flexible enough to adapt to different hardware and use cases

1.3 Existing System

Existing hand gesture recognition systems, as mentioned in the provided references, leverage a variety of technologies and methods to enable human-computer interaction (HCI) through hand gestures. These systems are designed with the primary goal of offering users intuitive and natural means of interacting with computers or devices, thereby enhancing user experience and accessibility.

The technologies and methodologies employed in these systems can include computer vision algorithms, machine learning models, sensor-based approaches, and combinations thereof. Computer vision algorithms are utilized to analyses images or video feeds captured by cameras.

Disadvantages:

- Increased hardware usage can significantly escalate costs and complexity, potentially limiting accessibility: Integrating advanced hardware components often leads to higher production costs, which can make the technology prohibitively expensive for both manufacturers and end-users.
- Hardware integration adds substantial complexity to setup and maintenance, requiring specialized technical expertise: Implementing and maintaining sophisticated hardware systems can be challenging and often necessitates specialized knowledge
- Specialized hardware may hinder portability, restricting deployment in mobile or wearable devices: Devices that rely on bulky or non-standard hardware components can be impractical for portable applications. This limits their use in mobile devices and wearables, which are increasingly important in today's on-the-go lifestyle.
- Power-intensive hardware components can drain device batteries quickly, severely impacting usability: High power consumption is a critical drawback for hardware-intensive technologies.
- Environmental factors such as lighting, clutter, and occlusions can disrupt hand gesture recognition, leading to inaccuracies and user frustration: Hand gesture recognition systems are highly sensitive to external conditions

1.4 Proposed System

In this proposed framework, we are going to utilize camera and the screen for the reading inputs and displaying outputs. We are using our hand fingers to drawing on the output screen. We have to be on safe distance where our hand can be fully visible in camera hence it is reading our input by recognizing movements of our fingers tip. Some other hand sign for selecting shapes and draw going to use as per given in used modules and libraries

The virtual paint application presented is based on the frames recorded by the PC's web camera. The frames are captured by the web camera and sent to the system. The video frames are transformed from BGR to RGB color to locate the hands in the video frame. The system then determines which finger is up by comparing the tip Id of the corresponding finger found via the MediaPipe to the respective coordinates of the up fingers, and then performs the appropriate function. The user can write anything on the screen if his or her index finger is raised.

Advantages Of Proposed System

1. Intuitive Interaction: The use of hand gestures for drawing provides a natural and intuitive way for users to interact with the system, making it easy to learn and use.

2. No Physical Contact Required: As the system relies on camera-based input, there is no need for physical contact with a device, which is hygienic and suitable for public or shared spaces.

3. Portability: The system can be used with any standard PC equipped with a web camera, making it highly portable and accessible without the need for specialized hardware.

4. Cost-Effective: Utilizing a web camera and open-source software libraries like OpenCV and MediaPipe reduces the overall cost of the system, making it an affordable solution for virtual drawing applications.

5. Real-time Processing: The framework processes video frames in real-time, providing immediate feedback and a smooth user experience, which is crucial for applications requiring precision and responsiveness.

6. Versatility in Gesture Recognition: The system is capable of recognizing various hand signs and gestures, allowing users to select different shapes, colours, and tools easily, enhancing the functionality and versatility of the application.

CHAPTER 2

LITERATURE SURVEY

2.1 Introduction

A literature survey, also known as a literature review, entails a systematic examination and synthesis of existing scholarly works, such as articles, books, and conference papers, relevant to a specific research topic or inquiry. It serves to provide researchers with a comprehensive understanding of the current state of knowledge in their field, identifying key concepts, theories, methodologies, and findings. By critically analyzing and synthesizing existing literature, researchers can uncover gaps, inconsistencies, and opportunities for further investigation, thereby informing their own research design and contributing to the scholarly discourse. Additionally, literature surveys support the development of well-informed arguments, enhance research credibility, and demonstrate the researcher's engagement with relevant scholarship.

2.2 Survey Papers

1] TITLE: Marker Based Hand Gesture Recognition

YEAR : 2016

AUTHOR: Siam, Sayem & Sakel, Jahidul & Kabir, Md.

The proposed method for real-time hand gesture recognition leverages marker detection and tracking to enhance Human-Computer Interaction (HCI). Instead of traditional input devices like a mouse or touchpad, this innovative approach uses two distinct colored markers worn on the fingertips to generate hand movements and provide instructions to a computer equipped with a consumer-grade camera. The system captures video frames using a standard web camera and processes these frames to detect the colored markers. By converting the frames from BGR to a different color space, such as HSV or RGB, the markers are easily isolated through color segmentation techniques. The coordinates of the detected markers are extracted and interpreted to recognize specific gestures. These gestures, based on the relative positions and movements of the markers, allow users to perform various interactive functions such as moving the cursor, clicking, dragging, and selecting shapes. The method can track up to eight distinct hand movements, providing a versatile and intuitive way to interact with a virtual canvas or other applications. This approach not only simplifies interaction but also enhances accessibility, making it suitable for diverse user groups and various computing environments

2] TITLE: Skeleton-Based Dynamic Hand Gesture Recognition

YEAR: 2016

AUTHOR: De Smedt, H. Wannous and J. Vandeborre.

The system introduces a method using skeletal representations for accurate dynamic gesture recognition. This approach promises superior capture of nuanced hand movements compared to traditional 2D methods, significantly enhancing the accuracy and interpretation of user intentions. By modeling the skeletal structure of the hand, the system can recognize intricate gestures in real-time, providing a more precise and responsive user experience. However, there are notable limitations to this method. The effectiveness of skeletal representation can degrade in complex environments with occlusions or clutter, where parts of the hand may be hidden from the camera's view, leading to incomplete or inaccurate gesture detection.

3] TITLE: Augmented Airbrush for Computer Aided Painting (CAP)

YEAR : 2017

AUTHOR : Roy Shilkrot, Pattie Maes, Joseph A. Paradiso, and Amit Zoran.

To work our expanded artificially glamorize, the client remains before the material, allowed to chip away at any piece of the composition, utilize any style, and counsel the PC screen in the event that the person wishesThe reference and material are lined up with an aligned focus point that relates to the virtual beginning. The client can move the gadget utilizing a coordinated strategy (testing the material, cutting it, journeying shapes, and so on), a more instinctive one (irregular strolling or nearby spotlight on a solitary region), or a blend of both. Strategies can be used, balancing precision with creative freedom. This hybrid approach facilitates meticulous crafting of certain elements while allowing for improvisation in others, enabling the creation of unique, dynamic artworks that merge structured detail with artistic expression. The system’s flexibility and adaptability cater to a wide range of artistic styles and preferences, supporting both detailed, methodical work and free-form, experimental exploration.

4] TITLE: An Arduino Based Gesture Control System

YEAR: 2018

AUTHOR: Belgamwar and S. Agrawa.

The developed system introduces a new Human-Computer Interaction (HCI) technique that integrates a camera, an accelerometer, a pair of Arduino microcontrollers, and ultrasonic distance sensors. The primary concept behind this interface is to capture hand motions using the ultrasonic distance sensors, which calculate the distance between the hand and the sensor to record gestures. This innovative approach allows for precise gesture recognition and control, enhancing the interactivity and responsiveness of the system.

However, the system has some limitations. The reliance on multiple hardware components increases the overall complexity and cost of the setup. Ultrasonic sensors, while effective, can be influenced by environmental factors such as sound reflections and interference from other ultrasonic sources, potentially leading to inaccurate distance measurements. The system may also require careful calibration and positioning of sensors to ensure optimal performance, which can be time-consuming and may not be user-friendly for non-technical users. Additionally, the integration of various hardware elements necessitates robust coordination and synchronization, which can be challenging to maintain consistently. Despite these drawbacks, the combination of different sensing technologies in this HCI technique offers a promising approach to gesture-based interaction

5] TITLE: Real Time Hand Gesture Recognition by Skin Color Detection

YEAR: 2019

AUTHOR: S. Khan, M. E. Ali, S. Das and M. M Rahman.

The system developed uses a skin color detection algorithm to convert American Sign Language (ASL) into text from real-time video. This approach leverages the distinct color properties of human skin to identify and track hand movements, translating them into corresponding text. However, there are several challenges and negative points associated with this method. Detecting hands based on skin color can be problematic due to the variability in skin tones and hand shapes among different individuals. Variations in lighting conditions can also affect the accuracy of skin color detection, leading to potential misinterpretations or missed detections. Additionally, background colors and objects that closely match skin tones can interfere with the detection algorithm, further complicating the recognition process.

6] TITLE: Virtual Painting Using Ball Tracking

YEAR: 2019

AUTHOR: Prajakta Vidhate, Revati Khadse and Saina Rasal.

The system introduces a virtual paint application that employs ball-tracking technology to facilitate hand movement tracking and writing on the screen. In this innovative approach, a glove with a ping- pong ball attached serves as a contour for tracking purposes. This method leverages the easily detectable shape and color of the ping-pong ball to track hand movements, allowing users to draw and interact with a virtual canvas effectively. However, there are several potential limitations to this approach. The accuracy and responsiveness of ball-tracking technology can be significantly influenced by external factors such as lighting conditions and the speed of hand movements. For instance, poor lighting or rapid hand movements might result in less precise tracking, impacting the user experience and overall effectiveness of the system. Additionally, the physical presence of a ping-pong ball on the glove might feel cumbersome to some users, potentially reducing the natural feel of the interaction. These factors need to be carefully managed to ensure the system's reliability and user satisfaction in various operating conditions.

7] TITLE: Dynamic Finger Gestures Using a Data Glove

YEAR : 2020

AUTHOR: M. Lee and J. Bae

The proposed system also explores a data glove-based approach for recognizing real-time dynamic hand gestures. This approach involves a data glove equipped with ten soft sensors that measure the joint angles of each of the five fingers, allowing for the precise collection of gesture data. The system utilizes techniques such as gesture spotting, which identifies the start and end of a gesture within a continuous sequence of movements, gesture sequence simplification, which reduces the complexity of gesture sequences for easier interpretation, and gesture recognition, which accurately identifies specific gestures based on the collected data.

Despite its advantages, there are some negative points to consider with this approach. The data glove can be cumbersome and uncomfortable to wear for extended periods, potentially limiting user adoption. The need for specialized hardware increases the overall cost and complexity of the system compared to simpler camera-based solutions. Calibration of the glove sensors can be time-consuming and may require frequent adjustments to maintain accuracy. Additionally, the reliance on physical sensors means that the system might be prone to hardware failures or wear and tear over time, affecting long-term reliability and performance.

CHAPTER 3

SYSTEM REQUIREMENT SPECIFICATION

3.1 Introduction

The introduction of the Virtual Painting Blueprint (VPB) gives a brief overview of what the VPB covers, such as its purpose, scope, terminology, abbreviations, references, and a diagram showing the VPB framework. This report aims to gather, analyze, and provide a deep understanding of "Virtual Artistry of Color" by explaining its complex concepts. It outlines detailed specifications of user-focused features related to virtual painting. Additionally, the VPB serves as a comprehensive guide for artists, developers, and enthusiasts who seek to explore the innovative intersection of technology and art. It also addresses the technical architecture, implementation strategies, and potential applications of virtual painting in various creative industries. By providing a structured approach, the VPB ensures a seamless integration of artistic expression with digital tools.

3.2 Purpose

The goal of the Virtual Painting Blueprint is to outline the technical and creative features required for developing a virtual painting application. This application aims to offer users flexibility in creating artwork efficiently and easily. In essence, this blueprint provides a detailed overview of our virtual painting software, including its features and objectives. It describes the target audience for the project and the user interface, hardware, and software requirements. It defines how our clients, team, and users perceive the application and its functionality.

3.2.1 Scope

The Virtual Painting Blueprint (VPB) covers a wide range of virtual painting aspects. It explores methodologies, technologies, and user-oriented features in virtual painting applications. Additionally, it evaluates existing platforms, considers applications in art, education, and entertainment, and discusses future trends. The scope extends to examining the integration of emerging technologies such as augmented reality (AR) and artificial intelligence (AI) in virtual painting. It also addresses the challenges and opportunities in developing intuitive and immersive user interfaces. Ultimately, the VPB aims to offer concise recommendations for enhancing virtual painting experiences, fostering innovation, and expanding the accessibility and impact of virtual art in diverse fields

3.2.1 Python

Python supports many programming paradigms that include object-oriented, imperative, functional and procedural and also has a large standard and complete library Python enables seamless integration with other technologies essential for ML workflows. Its flexibility allows for the implementation of complex algorithms and workflows with ease, making it indispensable for machine learning

Python is a deciphered, significant level, broadly useful programming language. Made by Guido van Rossum and first delivered in 1991, Python's plan reasoning accentuates code meaningfulness with its prominent utilization of critical whitespace. Its language develops and object-arranged methodology plan to assist software engineers with composing clear, consistent code for little and huge scope ventures.

Python is progressively composed and trash gathered. It underpins numerous programming standards, including procedural, object-arranged, and practical programming. Python is frequently portrayed as a "batteries included" language because of its thorough standard library. Python is a multi-worldview programming language. Article arranged programming and organized writing computer programs are completely upheld, and a significant number of its highlights uphold useful programming and angle situated programming (counting by metaprogramming and metaobjects (enchantment methods)).] Many different standards are upheld by means of expansions, including plan by agreement and rationale programming

3.2.2 OpenCV

OpenCV is a library which is used for image recognize. It will identify our hand tracking and Drawing. It is library basically design to work on image processing and image recognition. Object detection image processing methods are included in the OpenCV computer vision library. Real-time computer vision applications can be created by utilising the OpenCV library forthe Python programming language. The processing of images and videos as well as analytical techniques like face and object detection use the OpenCV library.

3.2.3 MediaPipe

MediaPipe is a Google open-source framework that was initially released in 2019. MediaPipe has some built-in computer vision and machine learning capabilities. A machine learning inference pipeline is implemented using MediaPipe. ML inference is the process of running real data points. The MediaPipe framework is used to solve AI challenges that mostly include video and audio streaming. MediaPipe is multimodal and platform independent. As a result, cross-platform apps are created using the framework. Face detection, multi-hand tracking, hair segmentation, object detection, and tracking are just a few of the applications that MediaPipe has to offer. MediaPipe is a framework with a high level of fidelity. Low latency performance is provided through the MediaPipe framework. It's in charge of synchronizing time-series data.

The MediaPipe framework has been used to design and analyze systems using graphs, as well as to develop systems for application purposes. In the pipeline configuration, all of the system's steps are carried out. The pipeline that was designed can run on a variety of platforms and can scale across desktops and mobile devices. Performance evaluation, sensor data retrieval, and a collection of components are all part of the MediaPipe framework. Calculators are the parts of the system. The MediaPipe framework uses a single-shot detector model for real-time detection and recognition of a hand or palm. It is first trained for the palm detection model in the hand detection module since palms are easier to train. It designates a hand landmark in the hand region, consisting of 21 joint or knuckle coordinates as shown in the Figure.

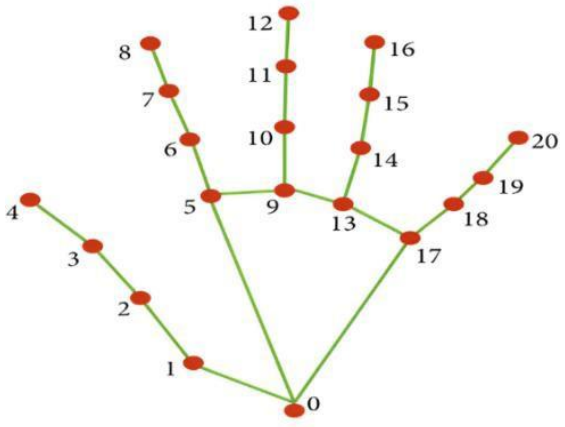

Fig 3.2.1 Hand Landmarks

3.2.4 NumPy

NumPy is a fundamental library for scientific computing in Python, providing data structures essential for deploying OpenCV with Python. Its name, short for Numerical Python, reflects its core purpose: handling multidimensional array objects and facilitating a variety of operations on them. NumPy offers a comprehensive set of functions for mathematical and logical operations, making it invaluable for tasks involving numerical computations and data manipulation. As a Python package, NumPy is widely used across various domains, including image processing, machine learning, and scientific research, thanks to its efficiency and versatility. Its seamless integration with OpenCV allows developers to leverage both libraries synergistically, enabling powerful image processing workflows within Python environments.

3.3 Functional Requirements

Functional requirements specify the specific functions or features a system must perform to satisfy the user's needs or business requirements. These requirements describe what the system should do in terms of input, processing, and output. They typically include detailed descriptions of system behaviors, data processing, and interactions with users or other systems.

- **Using Camera to Capture Input:**
  - Utilize a Python library like OpenCV to access the camera feed and capture frames.
  - OpenCV (Open Source Computer Vision Library) is a powerful tool for accessing and manipulating the camera feed. It provides simple functions to capture frames from a camera, which can then be processed for gesture recognition and other tasks.
- **Detect Hand Positions and Finger Tips:**
  - Employ hand detection algorithms such as OpenCV's Hand Tracking Module or MediaPipe Hands to detect and track hand positions and finger tips in the captured frames.
- **Choose Different Colors, and Sizes:**
  - Implement a user interface (UI) using a library like opencv to allow users to choose different , colors, and sizes.
- **Save the Work on Canvas as Image:**
  - Implement functionality to save the contents of the canvas as an image file using Python's built-in file I/O capabilities.

3.4 Non-Functional Requirements

1. **Reliability requirements**: The system must accurately interpret hand gestures at least 95% of the time, ensuring consistent and reliable mapping of gestures to painting actions.

2. **Scalability requirements:** The system should support an increasing number of concurrent users as usage grows, maintaining real-time responsiveness even with a large number of simultaneous users painting via gestures.

3. **Usability requirements:** The user interface should provide clear visual feedback for recognized gestures and intuitive controls for selecting colors, and brush sizes, ensuring a seamless painting experience for users of all skill levels.

4. **Availability requirements**: The system should be accessible 24/7, with minimal downtime for maintenance, ensuring users can engage in virtual hand gesture painting whenever they desire without interruption. Scheduled maintenance should be infrequent and communicated well in advance to minimize disruption to users.

5. **Maintainability requirements:** The codebase should be well-organized and thoroughly documented, enabling easy maintenance and updates to accommodate new gesture recognition algorithms or features in the future.

3.4.1 Feasibility Study

1. Technical Feasibility:

Hand Gesture Recognition: Advanced computer vision techniques, such as convolutional neural networks (CNNs) or deep learning models, are readily available for accurate hand gesture recognition.

Real-time Processing: Real-time Processing: Modern hardware and software frameworks, such as GPUs and libraries like OpenCV, support real-time processing, enabling swift interpretation of hand gestures. This capability is crucial for applications like air canvas, where the immediate translation of an artist's movements into digital strokes is essential for a seamless and intuitive user experience In addition to OpenCV, other advanced frameworks s often used to implement machine learning models that enhance the accuracy and responsiveness of gesture recognition systems.

User Interface Development: Our air canvas application boasts a user interface crafted entirely from scratch, without relying on any pre-existing UI libraries or frameworks. Every aspect of the user interface, from the layout to the interactive elements, has been meticulously designed and implemented using basic functions and programming techniques.

2. Economic Feasibility:

Cost of Development: The development cost includes expenses for hardware (computing devices with cameras), software (development tools, libraries), and human resources (developer salaries, training). However, open-source libraries and existing frameworks can significantly reduce development costs.

Potential Revenue: Revenue streams may include direct sales of the painting system, subscription-based models for premium features, or monetization through ads if the system is offered as a free service.

3. Operational Feasibility:

User Adoption: Conducting user studies and feedback sessions during development can ensure the system meets user expectations and preferences, increasing the likelihood of user adoption.

Maintenance: Establishing clear maintenance procedures, including regular updates for improved performance and security patches, ensures smooth operation of the system post-deployment.

Technical Support: Providing user support channels, such as documentation, tutorials, and customer service, ensures users can troubleshoot issues and receive assistance when needed.

3.5 HARDWARE REQUIREMENTS

- Processor: Intel I5 or Above
- RAM: 4GB or Above
- Storage: 1GB or Above
- Web Camera

3.6 SOFTWARE REQUIREMENTS

- Operating system: Windows 7 or above
- Programming Language: python (version 3.7) or above
- Front End : Python Media pipe and OpenCV

CHAPTER 4

SYSTEM DESIGN

4.1 Introduction

System design is the process of defining the elements of a system such as the architecture, modules and components, the different interfaces of those components and the data that goes through that system. It is meant to satisfy specific needs and requirements of a business or organization through the engineering of a coherent and well-running system. Software architecture is the high level structure of a software system, the discipline of creating such structures, and the documentation of these structures. It is the set of structures needed to reason about the software system, and comprises the software elements, the relations between them, and the properties of both elements and relations. The architecture of a software system is a metaphor, analogous to the architecture of a building. It is a process of planning a new business system or replacing an existing system by defining its components or modules to satisfy the specific requirements. Before planning, you need to understand the old system thoroughly and determine how computers can best be used in order to operate efficiently .System Design focuses on how to accomplish the objective of the system.

4.2 Overview

This System Design delineates the architecture and implementation of the Air Canvas application, elucidating the purpose and rationale behind its major components. Air Canvas facilitates users in drawing and crafting artwork in mid-air through hand gestures captured by a camera, engendering an innovative and interactive milieu. The system entails several key components:

- **Activity Diagram:** Depicts the flow of activities within the application, illustrating the sequence of actions performed by users and the system's responses.
- **Data Flow Diagram:** Demonstrates the flow of data between various components of the system, elucidating how information is processed and transformed.
- **Use Cases:** Describes the functionalities and interactions between actors (users) and the system, delineating the tasks users can perform and the system's responses.
- **Sequence Diagram:** Illustrates the sequence of interactions between users and the system, delineating the order in which actions occur and the communication between components.

4.3 Data Flow Diagram

The diagram illustrates the data flow in an Air Canvas Virtual Painting application, where the user's gestures are captured as input, processed by the Air Canvas system, and then displayed as visual output on a device. This flow ensures user actions are accurately translated into a virtual painting

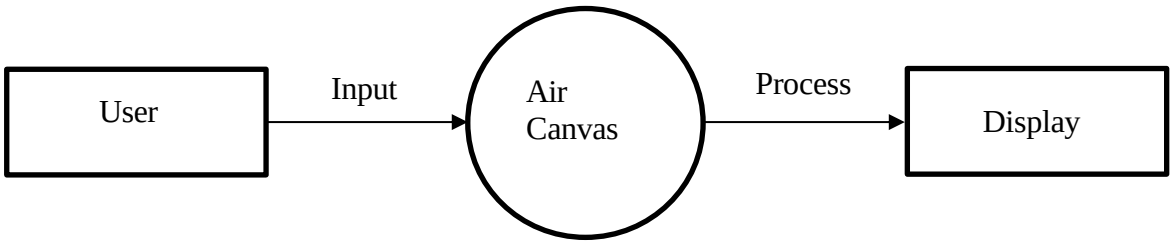

Fig 4.3.1: Data Flow Diagram Level 0

Working:

The user interacts with the system using gestures or movements captured by sensors, which serve as input actions such as drawing, selecting colors, and adjusting brush sizes. The Air Canvas system handles the logic for these functionalities, including processing the gestures and translating them into corresponding digital actions. This processed input is then visualized on the display device, allowing the user to see the results of their gestures in real-time. In the process flow, the user performs gestures or movements, which are captured as input by the system's sensors, such as cameras or motion-detecting devices. The captured input is sent to the Air Canvas system, where it is interpreted to determine the user's intended actions. The system then translates these actions into digital commands that affect the virtual canvas, such as rendering a stroke, updating the color palette, or adjusting brush settings. The resulting digital artwork is displayed on the output device, providing immediate feedback to the user and allowing for dynamic interaction with the virtual canvas. This integration of gesture capture, processing, and real-time visualization creates an intuitive and responsive drawing experience, enhancing the creative process for users

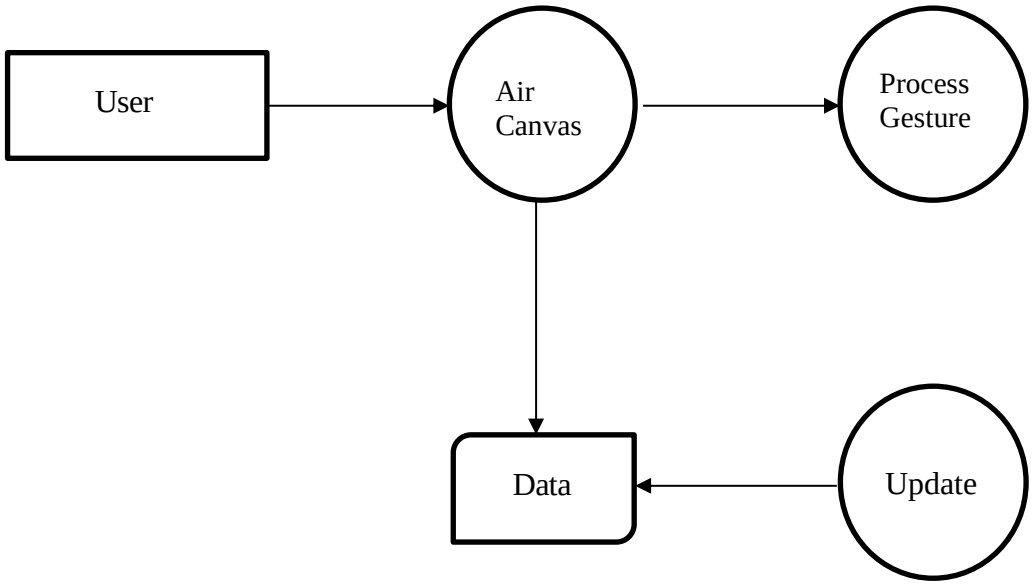

Fig 4.3.2: Data Flow Diagram Level 1

Working:

The user interacts with the Air Canvas application, which serves as the central system for handling user input and managing the drawing process. Through gestures and movements, the user can perform actions such as drawing, selecting colors, and adjusting brush sizes. The application interprets these gestures to determine the appropriate drawing actions, ensuring that the user's intentions are accurately captured. The Air Canvas system includes a component dedicated to processing gestures. This component interprets the user's gestures and translates them into specific drawing actions. The resulting data, which includes information about strokes, colors, and brush settings, is stored within the system. This stored drawing data is crucial for maintaining the state of the artwork and ensuring that updates can be made seamlessly. As the user continues to interact with the application, new gestures are captured and sent to the gesture processing component. This component updates the stored drawing data based on the latest input. The Air Canvas application then uses this updated data to refresh the canvas, ensuring that all changes are reflected in real-time on the display. This process allows for dynamic and continuous interaction, providing immediate visual feedback to the user and enhancing the overall drawing experience

4.4 Activity Diagram

This activity diagram represents the workflow of a program that detects finger movements and performs actions based on those movements. Here's a step-by-step explanation of the diagram:

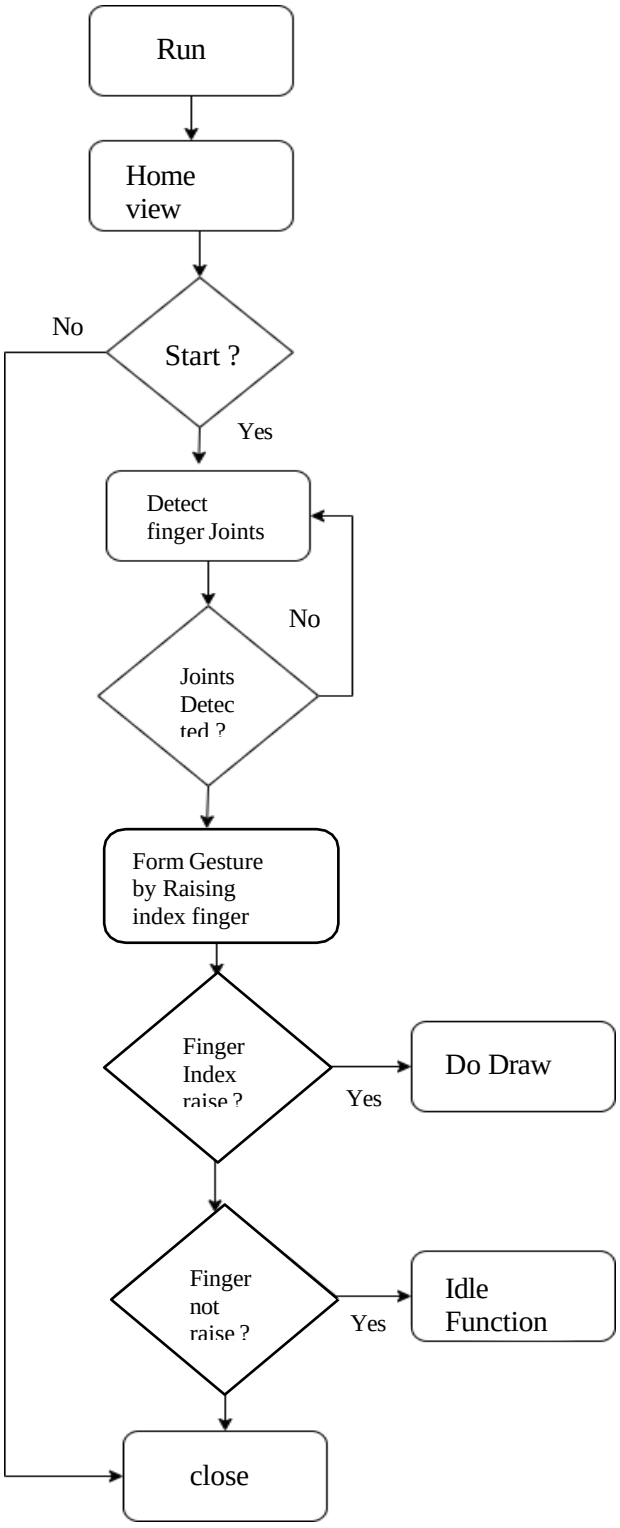

Fig 4.4: Activity Diagram

Working:

The process begins with the initial state, where the system is ready to start. The user initiates the process by running the program, which then transitions to the home view screen. At this stage, the system checks if the program should start. If the decision is "No," the process loops back to checking if the program should start. If "Yes," the system proceeds to the next step of detecting finger joints. The system starts detecting finger joints and continuously checks if a finger joint is detected. If no finger joint is detected, the system continues to loop back to the detection phase. Once a finger joint is detected, the system moves to the next step of forming a gesture by raising the index finger. The system then checks if the index finger is raised. If the index finger is raised, the system performs the "Do Draw Line Function," which allows the user to draw a line. If the system detects that no fingers are raised, it proceeds to the "Do Idle Function," which signifies that the system is in an idle state and no drawing action is performed. If the system does not detect the index finger being raised and no other specific gesture, it proceeds to "Close System," effectively shutting down the application. This sequence of actions, dictated by user inputs and finger gestures, captures the interactive flow of the program. It outlines the different states and decision points within the program, ensuring a structured response to the user's gestures. The diagram effectively illustrates how the system transitions between states based on real-time input, maintaining a seamless user experience from start to finish.

4.5 Sequence Diagram

Sequence Diagram Depict Cooperations Among Classes as Far as a Trade of Messages After Some Time. They're Likewise Called Occasion Charts.

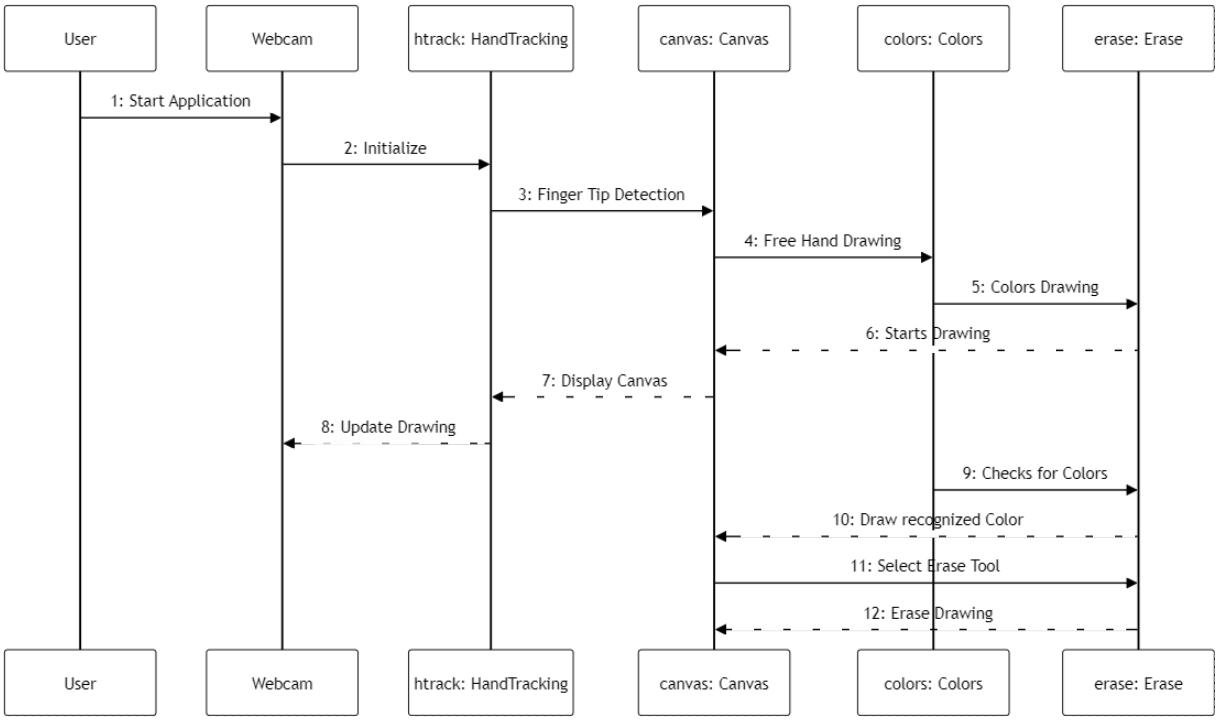

Fig 4.5: Sequence Diagram

Working:

The process begins when the user initiates the application. The application then initializes the webcam to capture video. Using HandTracking (Htrack), the system detects the positions of the fingertips for precise control. The detected fingertip positions enable freehand drawing. The system checks if drawing should be done in color and starts drawing on the canvas accordingly. The canvas is displayed to the user in real-time. As the user draws, the system updates the drawing with new inputs. It checks if different colors are needed and uses the detected color to draw. If the user selects the erase tool, the system allows parts of the drawing to be erased. This process ensures a seamless and intuitive drawing experience, with real-time updates and color management, providing a responsive environment for creating digital artwork.

4.6 Use Case Diagram:

The motivation behind use case diagram is to catch the dynamic part of a framework. In any case, this definition is too nonexclusive to even think about describing the reason, as other four outlines (action, grouping, cooperation, and State chart) likewise have a similar reason.

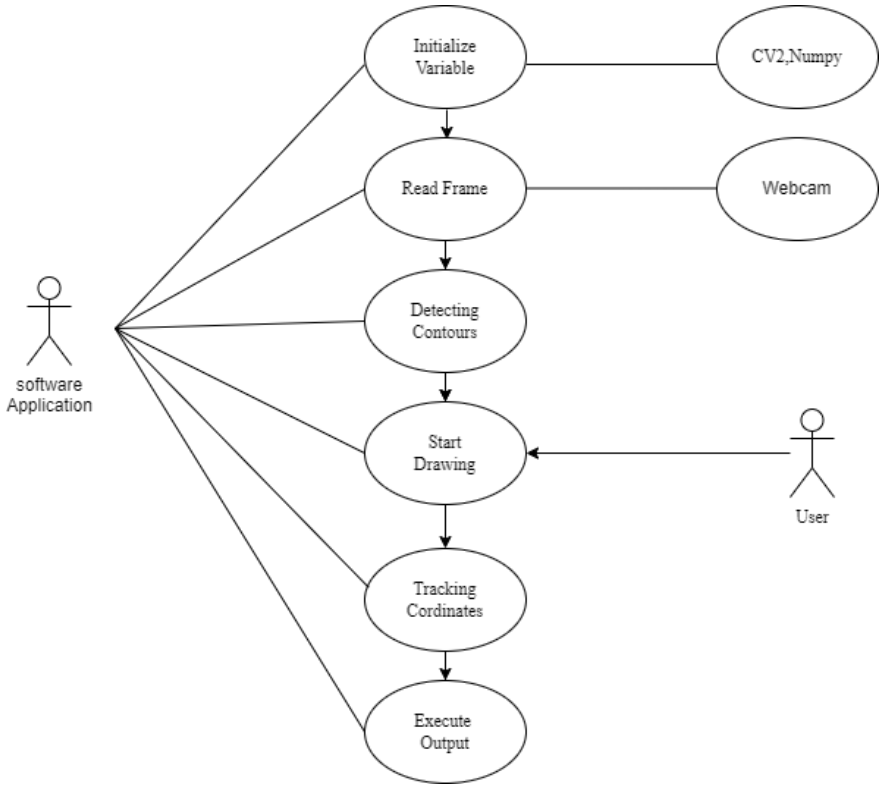

Fig 4.6: Use Case Diagram

Working:

The process begins by initializing variables and loading essential libraries like OpenCV and NumPy for image processing and numerical computations. The system then captures frames from the webcam to provide a live video feed. These frames are processed to detect objects or contours, identifying shapes and edges within the frame for drawing. Based on the detected contours, the system starts drawing and tracks the coordinates of the object to ensure accurate and continuous rendering on a digital canvas. Finally, the system executes the output by displaying or saving the drawing, providing real-time visual feedback to the user. The components involved include the webcam for capturing video frames, the user who interacts with the application, and the libraries CV2 and NumPy for handling image processing and numerical operations.

CHAPTER 5

IMPLEMENTATION

5.1 Introduction

A crucial phase in the system lifecycle is the successful implementation of the new system design. Implementation simply means converting a new system design into operation. Implementation is a stage in which the design is converted into working system. Implementation is the process of bringing the developed system into operational use and turning it to the user. This stage is considered to be the most crucial stage in the development of a successful system since a new system is developed and the users are given the confidence of its effectiveness.

5.2 Working System Code

```
from handTracker import *
import cv2
import mediapipe as mp
import numpy as np
import random
import datetime
class ColorRect():
    def __init__(self, x, y, w, h, color, text='', alpha=0.5):
        self.x = x
        self.y = y
        self.w = w
        self.h = h
        self.color = color
        self.text = text
        self.alpha = alpha
```

```
 def drawRect(self, img, text_color=(255, 255, 255), fontFace=cv2.FONT_HERSHEY_SIMPLEX, fontScale=0.8, thickness=2):
        alpha = self.alpha
        bg_rec = img[self.y: self.y + self.h, self.x: self.x + self.w]
        white_rect = np.ones(bg_rec.shape, dtype=np.uint8)
        white_rect[:] = self.color
        res = cv2.addWeighted(bg_rec, alpha, white_rect, 1 - alpha, 1.0)
        img[self.y: self.y + self.h, self.x: self.x + self.w] = res
 text_size = cv2.getTextSize(self.text, fontFace, fontScale, thickness)
        text_pos = (int(self.x + self.w / 2 - text_size[0][0] / 2), int(self.y + self.h / 2 + text_size[0][1] / 2))
        cv2.putText(img, self.text, text_pos, fontFace, fontScale, text_color, thickness)
 def isOver(self, x, y):
        return self.x + self.w > x > self.x and self.y + self.h > y > self.y
def save_canvas(canvas):
    filename = f"canvas_{datetime.datetime.now().strftime('%Y-%m-%d_%H-%M-%S')}.png"
    cv2.imwrite(filename, canvas)
    print(f"Canvas saved as {filename}")
detector = HandTracker(detectionCon=0.8)
cap = cv2.VideoCapture(0)
cap.set(3, 1280)
cap.set(4, 720)
canvas = np.zeros((720, 1280, 3), np.uint8)
px, py = 0, 0
color = (255, 0, 0)
brushSize = 5
eraserSize = 20
colorsBtn = ColorRect(200, 0, 100, 100, (120, 255, 0), 'Colors')
```

```
colors = [
    ColorRect(300, 0, 100, 100, (int(random.random() * 255) - 1, int(random.random() * 255), int(random.random() * 255))),
    ColorRect(400, 0, 100, 100, (0, 0, 255)),  # Red
    ColorRect(500, 0, 100, 100, (255, 0, 0)),  # Blue
    ColorRect(600, 0, 100, 100, (0, 255, 0)),  # Green
    ColorRect(700, 0, 100, 100, (0, 255, 255)),  # Yellow
    ColorRect(800, 0, 100, 100, (0, 0, 0), "Eraser"),  # Erase (black)
]
clear = ColorRect(900, 0, 100, 100, (100, 100, 100), "Clear")
pens = [ColorRect(1100, 50 + 100 * i, 100, 100, (50, 50, 50), str(penSize)) for i, penSize in enumerate(range(5, 25, 5))]
penBtn = ColorRect(1100, 0, 100, 50, color, 'Pen')
boardBtn = ColorRect(50, 0, 100, 100, (255, 255, 0), 'Board')
whiteBoard = ColorRect(50, 120, 1020, 580, (255, 255, 255), alpha=0.1)
saveBtn = ColorRect(1000, 0, 100, 100, (0, 255, 255), 'Save')
coolingCounter = 20
hideBoard = True
hideColors = True
hidePenSizes = True
while True:
    if coolingCounter:
        coolingCounter -= 1
  ret, frame = cap.read()
    if not ret:
        break
    frame = cv2.resize(frame, (1280, 720))
    frame = cv2.flip(frame, 1)
 detector.findHands(frame)
```

```
positions = detector.getPostion(frame, draw=False)
  upFingers = detector.getUpFingers(frame)
if upFingers:
      x, y = positions[8][0], positions[8][1]
      if upFingers[1] and not whiteBoard.isOver(x, y):
         px, py = 0, 0
if not hidePenSizes:
            for pen in pens:
              if pen.isOver(x, y):
                 brushSize = int(pen.text)
                 pen.alpha = 0
               else:
                 pen.alpha = 0.
 if not hideColors:
            for cb in colors:
              if  cb.isOver(x,  y):
                 color = cb.color
                 cb.alpha = 0
              else:
                 cb.alpha = 0.5
if clear.isOver(x, y):
                 clear.alpha = 0
                 canvas = np.zeros((720, 1280, 3), np.uint8)
            else:
                 clear.alpha = 0.5
       if saveBtn.isOver(x, y) and not coolingCounter:
            coolingCounter = 10
            saveBtn.alpha = 0
```

```
  save_canvas(canvas)
        else:
            saveBtn.alpha = 0.5
         if colorsBtn.isOver(x, y) and not coolingCounter:
            coolingCounter = 10
            colorsBtn.alpha = 0
            hideColors = not hideColors
            colorsBtn.text = 'Colors' if hideColors else 'Hide'
          else:
            colorsBtn.alpha = 0.5
if penBtn.isOver(x, y) and not coolingCounter:
            coolingCounter = 10
            penBtn.alpha = 0
            hidePenSizes = not hidePenSizes
            penBtn.text = 'Pen' if hidePenSizes else 'Hide'
          else:
            penBtn.alpha = 0.5
      if boardBtn.isOver(x, y) and not coolingCounter:
            coolingCounter = 10
            boardBtn.alpha = 0
            hideBoard = not hideBoard
            boardBtn.text = 'Board' if hideBoard else 'Hide'
          else:
            boardBtn.alpha = 0.5
        elif upFingers[1] and not upFingers[2]:
         if whiteBoard.isOver(x, y) and not hideBoard:
            cv2.circle(frame, positions[8], brushSize, color, -1)
            if px == 0 and py == 0:
```

```
 px, py = positions[8]
                if color == (0, 0, 0):
                    cv2.line(canvas, (px, py), positions[8], color, eraserSize)
                else:
                    cv2.line(canvas, (px, py), positions[8], color, brushSize)
                px, py = positions[8]
        else:
            px, py = 0,
    colorsBtn.drawRect(frame)
    cv2.rectangle(frame, (colorsBtn.x, colorsBtn.y),
 (colorsBtn.x + colorsBtn.w, colorsBtn.y + colorsBtn.h), (255, 255, 255), 2)
boardBtn.drawRect(frame)
    cv2.rectangle(frame, (boardBtn.x, boardBtn.y),
(boardBtn.x + boardBtn.w, boardBtn.y + boardBtn.h), (255, 255, 255), 2)
 if not hideBoard:
        whiteBoard.drawRect(frame)
        canvasGray = cv2.cvtColor(canvas, cv2.COLOR_BGR2GRAY)
        _, imgInv = cv2.threshold(canvasGray, 20, 255, cv2.THRESH_BINARY_INV)
        imgInv = cv2.cvtColor(imgInv, cv2.COLOR_GRAY2BGR)
        frame = cv2.bitwise_and(frame, imgInv)
        frame = cv2.bitwise_or(frame, canvas)
 if not hideColors:
        for c in colors:
            c.drawRect(frame)
            cv2.rectangle(frame, (c.x, c.y), (c.x + c.w, c.y + c.h), (255, 255, 255), 2)
clear.drawRect(frame)
        cv2.rectangle(frame, (clear.x, clear.y),
(clear.x + clear.w, clear.y + clear.h), (255, 255, 255),2)
```

```
penBtn.color = color
    penBtn.drawRect(frame)
  cv2.rectangle(frame, (penBtn.x, penBtn.y), (penBtn.x + penBtn.w, penBtn.y + penBtn.h), (255, 255, 255), 2)
  if not hidePenSizes:
        for pen in pens:
            pen.drawRect(frame)
            cv2.rectangle(frame, (pen.x, pen.y),
(pen.x + pen.w, pen.y + pen.h), (255, 255, 255), 2)
saveBtn.drawRect(frame)
    cv2.rectangle(frame, (saveBtn.x, saveBtn.y),
 (saveBtn.x + saveBtn.w, saveBtn.y + saveBtn.h), (255, 255, 255), 2)
cv2.imshow('video', frame)
    k = cv2.waitKey(1)
    if k == ord('q') or cv2.getWindowProperty('video', cv2.WND_PROP_VISIBLE) < 1:
        break
cap.release()
cv2.destroyAllWindows()
```

CHAPTER 6

TESTING

6.1 Introduction

Testing is the way toward running a framework with the expectation of discovering blunders. Testing upgrades the uprightness of the framework by distinguishing the deviations in plans and blunders in the framework. Testing targets distinguishing blunders – prom zones. This aides in the avoidance of mistakes in the framework. Testing additionally adds esteems to the item by affirming the client's necessity.

The primary intention is to distinguish blunders and mistake get-prom zones in a framework. Testing must be intensive and all around arranged. A somewhat tried framework is as terrible as an untested framework. Furthermore, the cost of an untested and under-tried framework is high. The execution is the last and significant stage. It includes client preparation, framework testing so as to guarantee the effective running of the proposed framework. The client tests the framework and changes are made by their requirements. The testing includes the testing of the created framework utilizing different sorts of information. While testing, blunders are noted and rightness is the mode.

6.2 Objectives Of Testing

Testing in a cycle of executing a program with the expectation of discovering mistakes. An effective experiment is one that reveals an up 'til now unfamiliar blunder. Framework testing is a phase of usage, which is pointed toward guaranteeing that the framework works accurately and productively according to the client's need before the live activity initiates.

As expressed previously, testing is indispensable to the achievement of a framework. Framework testing makes the coherent presumption that if all the framework is right, the objective will be effectively accomplished. A progression of tests is performed before the framework is prepared for the client acknowledgment test.

6.3 Testing Methods

System testing is a stage of implementation. This helps the weather system works accurately and efficiently before live operation commences. Testing is vital to the success of the system. The candidate system is subject to a variety of tests: online response, volume, stress, recovery, security, and usability tests series of tests are performed for the proposed system are ready for user acceptance testing.

Unit Testing:

Unit testing chiefly centers around the littlest unit of programming plan. This is known as module testing. The modules are tried independently. The test is done during the programming stage itself. In this progression, every module is discovered to be working acceptably as respects the normal yield from the module.

Integration Testing:

Mix testing is an efficient methodology for developing the program structure, while simultaneously leading tests to reveal blunders related with the interface. The goal is to take unit tried modules and manufacture a program structure. All the modules are joined and tried in general.

Output Testing:

Subsequent to performing approval testing, the following stage is yield trying of the proposed framework, since no framework could be valuable on the off chance that it doesn't create the necessary yield in a particular configuration. The yield design on the screen is discovered to be right. The organization was planned in the framework configuration time as indicated by the client needs. For the printed copy likewise, the yield comes according to the predefined prerequisites by the client. Subsequently yield testing didn't bring about any amendment for the framework.

User Acceptance Testing:

Client acknowledgment of a framework is the vital factor for the achievement of any framework. The framework viable is tried for client acknowledgment by continually staying in contact with the imminent framework clients at the hour of creating and making changes at whatever point required.

Black Box Testing

For an Air Canvas Virtual Painting application, black box testing involves evaluating the application without knowledge of its internal code, focusing on inputs and outputs. This includes verifying functionality such as gesture recognition, color selection, brush size adjustment, and undo/redo features. Additionally, usability testing assesses the intuitiveness and ease of use, performance testing measures responsiveness and handling under high load, and compatibility testing ensures consistent performance across different devices and operating systems. The goal is to ensure the application works as expected from the user's perspective.

White Box Testing

White box testing for an Air Canvas Virtual Painting application involves examining the internal code and logic. This includes unit tests to verify the accuracy and efficiency of gesture recognition algorithms, functions for color and brush size handling, and action management. Integration testing ensures seamless interaction between modules like gesture inputs and painting outputs. Code coverage analysis identifies and tests all paths, branches, and conditions, while security assessments review the code for vulnerabilities. Performance profiling identifies and optimizes bottlenecks, ensuring the application is robust, efficient, and secure. The focus is on the internal workings and code quality

Security Testing

Security testing is a type of software testing designed to uncover vulnerabilities and ensure that the application is protected against threats such as unauthorized access, data breaches, and other security risks. The primary goal of security testing is to identify potential weaknesses in the system's defenses so that they can be addressed before malicious entities exploit them. Involves identifying potential vulnerabilities and ensuring the application is protected against threats such as unauthorized access, data breaches, and other security risks. For the Air Canvas Virtual Painting application, this would include verifying that saved artwork is securely stored, user data is protected, and there are no security flaws in the code that could be exploited.

6.4 TEST REPORTS

The users test the developed system when changes are made according to the needs. The testing phase involves the testing of the developed system using various kinds of data. An elaborate testing of data is prepared and system is tested using the test data.

Test Cases

No	Description	Input	Expected Value	Actual Value	Result
1	To draw free style of required colour on the board with the help of hand	Hand Tracking through webcam	Line of required colour is obtained from set of available colours	Line of required colour is obtained from set of available colours	PASS
2	To draw a ellipse virtually through Hand on a board	Hand Tracking through webcam	Ellipse is obtained with the help of little and index finger	Ellipse is obtained with the help of little and index finger	PASS
3	To clear the board with the help of hand tracking through clear button	Hand Tracking through webcam	Board gets cleared i.e, all the shapes drawn are erased	Board gets cleared i.e, all the shapes drawn are erased	PASS
7	To erase the unwanted parts on canvas	Hand Tracking through webcam	The pixels on canvas at index fingertip gets erased	The pixels on canvas at index fingertip gets erased	PASS
8	Trying to draw on canvas using middle finger	Hand Tracking through webcam	Drawing is not possible i.e, no changes will be depicted on canvas	Drawing is not possible i.e, no changes will be depicted on canvas	PASS
9	Saving the work done on canvas	Placing finger on save button	The work gets saved	The work gets saved	PASS

Table 6.4: Test Cases

CHAPTER 7

SNAPSHOTS

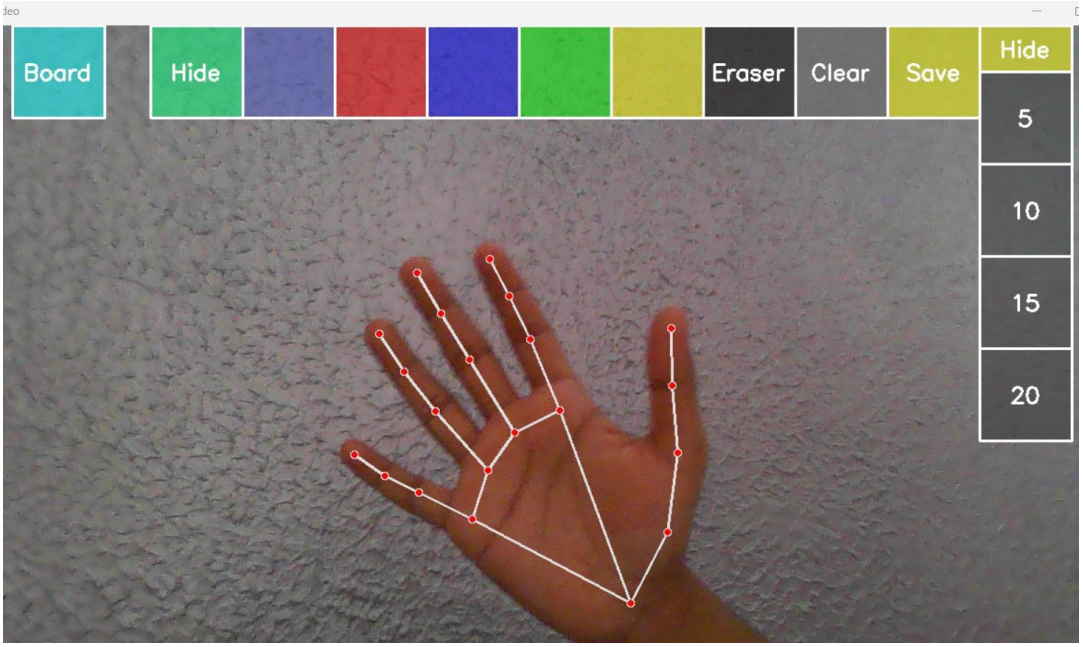

Fig 7.1: Hand Detect.

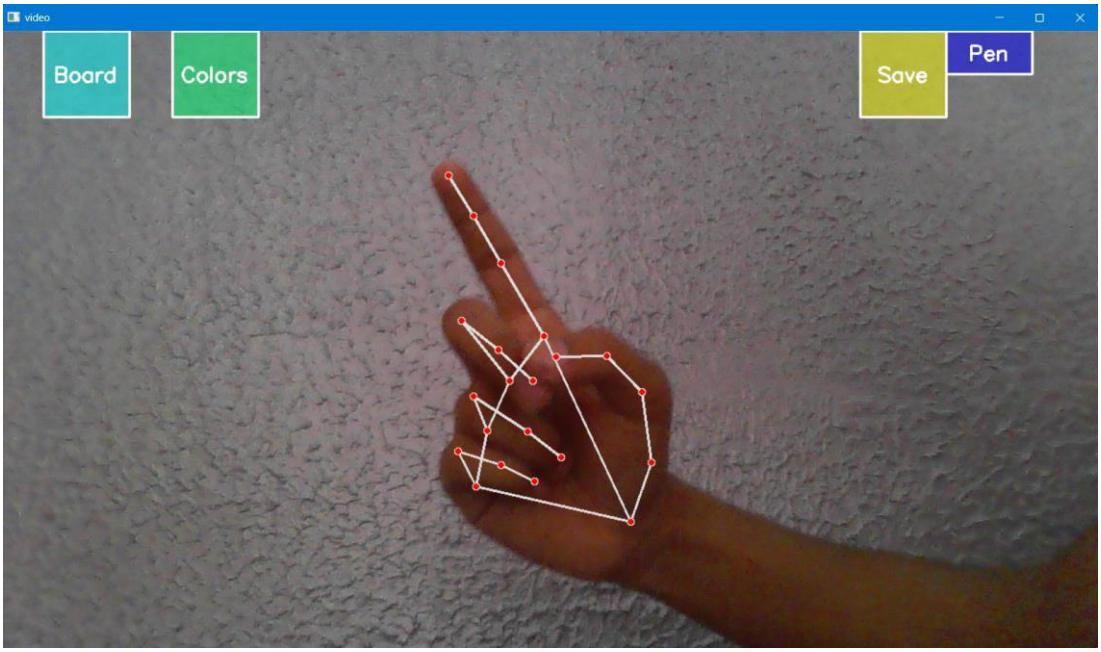

Fig 7.2: Index Finger Detect.

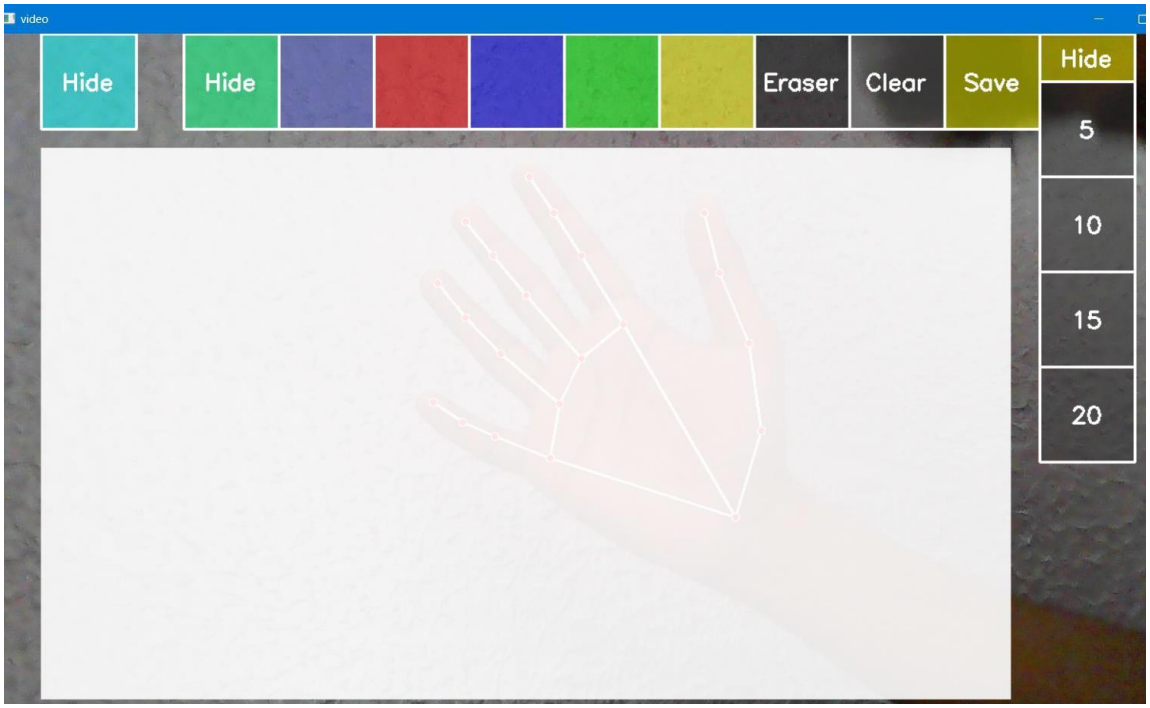

Fig 7.3: Drawing Board.

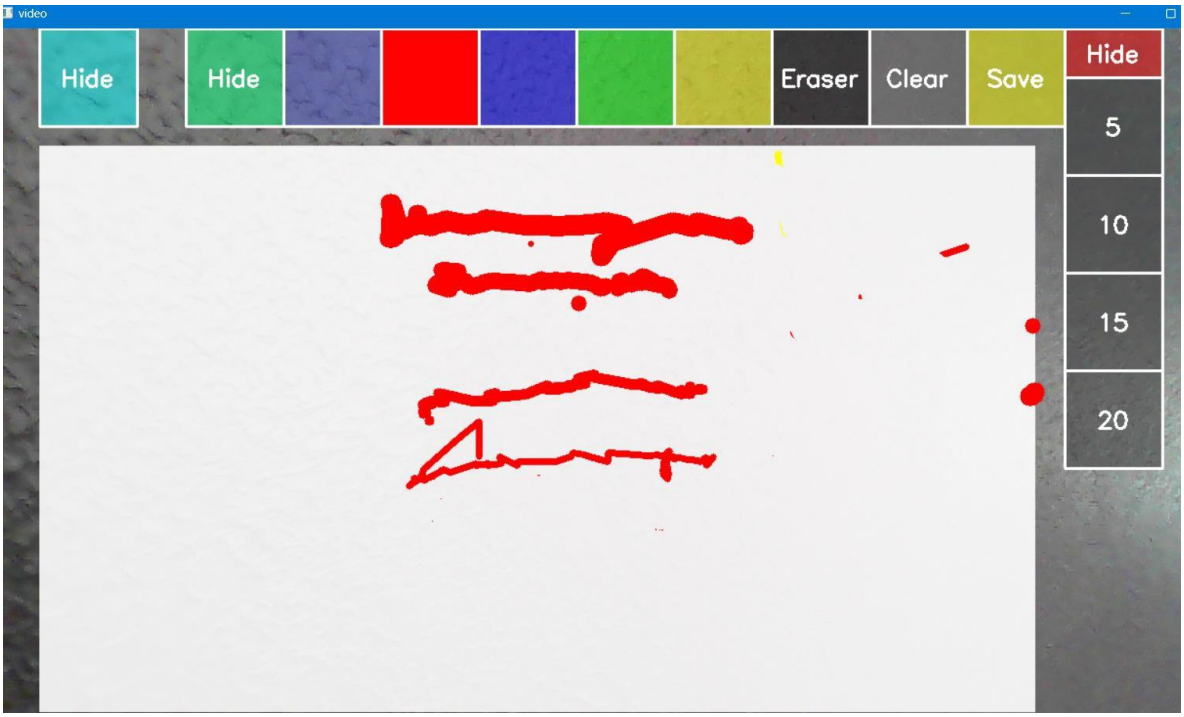

Fig 7.4: Pen Sizes.

CONCLUSION

This project has the potential to challenge traditional writing methods by providing a hands-free, intuitive way to take notes and interact with digital interfaces. By eliminating the need to carry a cell phone or other devices to take notes, it offers a more convenient and accessible solution. This innovation is particularly beneficial for individuals who rely on alternative communication methods, as it facilitates easier and more effective interaction.

Furthermore, the system's functionality can be expanded to control Internet of Things (IoT) devices, broadening its application beyond just note-taking and virtual painting. The air painting feature exemplifies the creative possibilities enabled by this technology, allowing users to draw and interact with a virtual canvas in an engaging manner.

As advancements in Artificial Intelligence continue, the efficiency and accuracy of air writing and gesture recognition are expected to improve, making these systems even more robust and user-friendly. This project not only enhances current HCI technologies but also paves the way for future innovations in how we interact with the digital world, offering significant benefits across various domains such as education, accessibility, and smart home control.

FUTURE ENHANCEMENT

Future work and enhancements for the virtual painting system could include several innovative features to expand its capabilities and improve user experience. Integrating the gesture recognition system with Augmented Reality (AR) and Virtual Reality (VR) platforms would create more immersive and interactive experiences, particularly beneficial for gaming, education, and professional applications. Enhanced user customization would allow users to tailor gestures and commands to their preferences, increasing the system's versatility and user-friendliness.

Developing functionality for controlling Internet of Things (IoT) devices using hand gestures would transform smart home interactions, making device management more intuitive and streamlined. Additionally, incorporating AI-driven predictive text and autocomplete features could speed up the writing process and enhance the user experience by predicting and suggesting words or phrases, reducing the effort required for text input. Utilizing cloud computing for data processing and storage would enable more complex computations, real-time updates, and multi-device access, ensuring robust performance and scalability. These enhancements would significantly broaden the functionality and appeal of the virtual painting system, making it a more powerful and user-friendly tool for a wide range of applications.

REFERENCES

[1] Siam, Sayem & Sakel, Jahidul & Kabir, Md.. Human Computer Interaction Using Marker Based Hand Gesture Recognition., 2016

[2] Q. De Smedt, H. Wannous and J. Vandeborre, "Skeleton-Based Dynamic Hand Gesture Recognition," 2016 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 2016, pp. 1206-1214, doi: 10.1109/CVPRW.2016.153.

[3] Roy Shilkrot, Pattie Maes, Joshep A.Paradiso and Amit Zoran, "Augmented Airbrush for Computer Aided Painting," 2017 International Conference on Recent Trends in Information Technology (ICRTIT), 2017, pp. 1-6, doi: 10.1109/ICRTIT.2017.643432.

[4] S. Belgamwar and S. Agrawal, "An Arduino Based Gesture Control System for Human- Computer Interface," 2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA), 2018, pp. 1-3, doi: 10.1109/ICCUBEA.2018.8697673.

[5] S. Khan, M. E. Ali, S. Das and M. M. Rahman, "Real Time Hand Gesture Recognition by Skin Color Detection for American Sign Language," 2019 4th International Conference on Electrical Information and Communication Technology (EICT), 2019, pp. 1-6, doi:10.1109/EICT48899.2019.9068809.

[6] Prajakta Vidhate, Revati Khadse, Saina Rasal, "Virtual paint ball tracking application by hand gesture recognition system", 2019 International Journal of Technical Research and Applications, 2019, pp. 36-39.

[7] M. Lee and J. Bae, "Deep Learning Based Real-Time Recognition of Dynamic Finger Gestures Using a Data Glove," in IEEE Access, vol. 8, pp. 219923-219933, 2020, doi:10.1109/ACCESS.2020.3039401

[8] P. Ramasamy, G. Prabhu and R. Srinivasan, "An economical air writing system converting finger movements to text using web camera," 2016 International Conference on Recent Trends in Information Technology (ICRTIT), 2020, pp. 1-6, doi: 10.1109/ICRTIT.2016.7569563.

[9] Y. Wu and C. -M. Wang, "Applying hand gesture recognition and joint tracking to a TV controller using CNN and Convolutional Pose Machine," 2020 24th International Conference on Pattern Recognition (ICPR), 2018, pp. 3086-3091, doi: 10.1109/ICPR.2018.8546209.

[10] R. Lyu et al., "A flexible finger-mounted airbrush model for immersive freehand painting," 201 IEEE/ACIS 16th International Conference on Computer and Information Science (ICIS), 2017, pp. 395-400, doi:10.1109/ICIS.2020.7960025.