Air Canvas using Machine Learning

A PROJECT REPORT

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

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE & ENGINEERING

Submitted by:

21BCS4771 Rohitansh

Under the Supervision of:
(CSE)

CHANDIGARH UNIVERSITY, GHARUAN, MOHALI - 140413, PUNJAB

April , 2024

BONAFIDE CERTIFICATE

Certified that this project report **"AIR CANVAS USING MACHINE LEARNING"** is the bonafide work of "21BCS4771-**Rohitansh Pathania"** who carried out the project work under my/our supervision.

SIGNATURE	**SIGNATURE**
HEAD OF THE DEPARTMENT	**SUPERVISOR**
CSE	CSE

Submitted for the project viva-voce examination held on

INTERNAL EXAMINER	**EXTERNAL EXAMINER**

Abstract

The "Air Canvas" project represents a unique convergence of artistry and machine learning, offering a novel approach to digital art creation. Traditional digital art tools often require specialized hardware like graphic tablets or styluses, which can be limiting and intimidating for beginners. Recent advancements in Artificial Intelligence and Machine Learning have made it possible to control a digital environment using hand movements. These hand gestures are commonly used in computer language instruction to convey information without words. The "Air Canvas" is an application that allows us to draw using our hand and finger gestures, captured through a web camera.

This project relies on two main technologies: **OpenCV and Mediapipe**, which fall under the category of computer vision. In more straightforward terms, "Air Canvas" is like a magic paintbrush that follows your hand movements as you create art in the air. It uses a smart computer program to understand what you're doing and turns it into a beautiful digital picture. The way this project operates is by constantly monitoring and analyzing the position of landmarks on the hand detected by the camera. These landmarks correspond to various parts of the hand, including the fingertips.

The project then uses the ID assigned to each fingertip to identify different modes of operation. For example, a specific fingertip gesture might be used for selecting, another for drawing, and yet another for clearing the canvas. This approach allows users to interact with the digital canvas naturally, as if they were using a pen or brush, making it an intuitive and user-friendly tool for creating digital art. The possibilities with "Air Canvas" are endless. Users can select different brush types, colors, and canvas sizes with simple gestures. This project not only serves as a creative outlet but also as an educational tool. It can be used in art classrooms to engage students in a new and exciting way, fostering a deeper appreciation for art and technology. "Air Canvas" exemplifies the potential of machine learning to empower individuals to express themselves artistically in ways previously unimaginable.

TABLE OF CONTENTS

List of Figures

Abbreviations Used:

ML	→	**Machine Learning**
AI	→	**Artificial Intelligence**
CV	→	**Computer Vision**
AR	→	**Augmented Reality**
UI	→	**User Interface**
IDE	→	**Integrated Development Environment**

CHAPTER- 1
INTRODUCTION

1.1 Identification of client and need:

Identifying the Client:

The client could be an entity or organization interested in leveraging innovative technology for various purposes such as education, entertainment, or creative expression. Potential clients might include:

- **Educational Institutions:** Schools, colleges, or online learning platforms seeking to incorporate interactive and engaging tools into their curriculum to enhance learning experiences.
- **Technology Companies:** Companies specializing in software development, augmented reality (AR), or machine learning (ML) keen on exploring novel applications and pushing the boundaries of technology.
- **Artists and Creatives:** Individuals or groups within the creative industry interested in utilizing cutting-edge tools to express themselves in new ways, such as digital artists or performers.

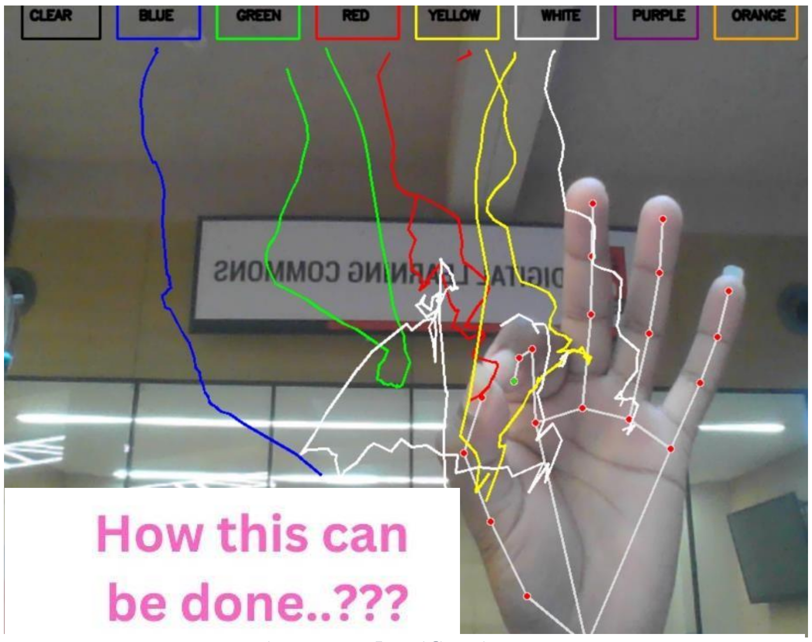


Figure 1 Identification

- **Healthcare Sector:** Rehabilitation centers or healthcare facilities interested in using innovative technology for therapeutic purposes, such as fine motor skills development.
- **Entertainment Industry:** Game development studios interested in incorporating novel interaction methods or gesture-based controls into their products. Media production companies looking for innovative ways to engage audiences through interactive experiences.
- **Technology Enthusiasts:** Individuals passionate about emerging technologies, such as machine learning and computer vision, who are interested in exploring their applications in real-world projects. Hackathon participants or hobbyist developers looking for inspiration for their next project.

Traditional digital art creation tools, characterized by their reliance on specialized hardware and software, present barriers to accessibility and ease of use, particularly for beginners and individuals with physical disabilities. These tools often require a steep learning curve and significant financial investment, limiting the ability of a diverse audience to engage in digital art creation. In the current situation, digital art and traditional art are both part of a symbiotic state, therefore we must thoroughly comprehend the fundamental understanding of the form between the two. Pen and paper, as well as chalk and board, are examples of traditional writing methods. Building a hand gesture recognition system to write digitally is the major goal of performing digital art.

Need for the Project:

- **Clarifying Objectives:** The project report will outline the objectives of developing an "AIR canvas using machine learning and Mediapipe."
- **Technical Feasibility Assessment:** A thorough analysis within the project report will evaluate the technical feasibility of implementing the proposed solution. This includes assessing the capabilities of machine learning algorithms and the Mediapipe framework in creating an interactive and responsive AIR (Augmented Reality) canvas.
- **Scope Definition:** The project report will define the scope of the endeavor, including the features and functionalities of the AIR canvas. It will outline what users can expect from the application, such as real-time drawing in 3D space, gesture recognition, and interaction with virtual objects.
- **Resource Requirements:** Identifying the resources needed—such as software tools, hardware infrastructure, and skilled personnel—is essential for the successful execution of the project. The report will detail these requirements, helping the client understand the investment necessary to bring the idea to fruition.

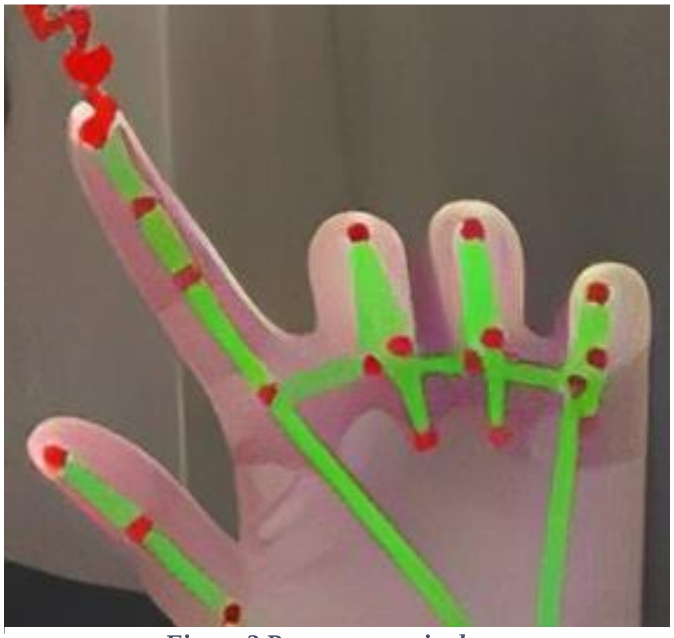


Figure 2 Resource required

- **Risk Assessment:** Every project carries inherent risks, whether they're technical, financial, or operational. The project report will conduct a risk assessment, identifying potential obstacles and proposing mitigation strategies to address them.
- **Future Roadmap**: Finally, the project report will outline a roadmap for the development and deployment of the AIR canvas. It will provide a timeline with milestones, indicating when key deliverables will be achieved and when the final product will be ready for launch. This roadmap will help the client visualize the journey ahead and understand the steps involved in realizing their vision.

Key Challenges:

- **Accessibility Barrier:** Existing digital art tools are inaccessible to individuals who may lack the financial means to purchase specialized hardware or who have physical disabilities that hinder their ability to operate traditional input devices.
- **Intimidation Factor:** The complexity of traditional digital art software and hardware can deter beginners from exploring their creative potential. The learning curve associated with these tools may discourage novices from pursuing digital art.
- **Limited Mobility:** People with physical disabilities may face significant challenges when trying to use traditional input devices, making it difficult for them to participate in digital art creation.
- **Accuracy and Precision:** Achieving accurate and precise hand tracking and gesture recognition is essential for the functionality of the Air Canvas. Even small errors in tracking can lead to significant distortions in the drawn shapes or lines.
- **Real-time Performance:** Ensuring that the system operates in real-time is crucial for providing a seamless and responsive user experience. This requires optimizing the performance of the machine learning models and CV algorithms to handle the computational demands efficiently.
- **Environmental Variability:** Adapting to different environmental conditions, such as varying lighting conditions or background clutter, poses a challenge for robust hand tracking and gesture recognition. The system should be resilient to such variability to maintain its functionality in diverse settings.
- **User Interface Design:** Designing an intuitive and user-friendly interface for the Air Canvas application is essential for engaging users and facilitating ease of use. This involves considering factors such as the layout of on-screen controls, feedback mechanisms, and interaction metaphors.

1.2 Relevant contemporary issues:

Addressing relevant contemporary issues within the project report on "AIR canvas using machine learning and computer vision (Mediapipe)" enhances its significance and applicability in the current technological landscape. Let's explore some pertinent issues:

- **Accessibility and Inclusivity:** Consideration should be given to making the AIR canvas accessible to users with diverse abilities and needs. This includes implementing features for users with disabilities, ensuring compatibility with assistive technologies, and designing an intuitive user interface that caters to a wide range of users.
- **Privacy and Data Security:** With the collection and processing of user data inevitable in machine learning applications, ensuring robust privacy measures is paramount. The project report should address how user data will be handled, stored, and protected to safeguard privacy and comply with relevant regulations such as GDPR or CCPA.
- **Ethical AI:** As AI technologies become increasingly integrated into everyday life, ethical considerations surrounding their development and deployment are critical. Discuss potential uses and misuses of the technology, and the ethical responsibilities of developers to ensure that the application benefits society while minimizing harm.
- **Environmental Sustainability:** Sustainability is an increasingly pressing concern in technology development. Discuss strategies for optimizing the application's energy efficiency and reducing its carbon footprint, such as minimizing computational resources and promoting responsible hardware usage.
- **Digital Divide:** Access to technology and digital skills is not uniform across populations, leading to disparities in opportunities and outcomes. Discuss efforts to promote technological equity and bridge the digital divide, ensuring that the benefits of the Air Canvas application are accessible to individuals from diverse socioeconomic backgrounds.
- **User Experience and Human-Centered Design:** Prioritizing user experience and incorporating principles of human-centered design can enhance the usability and effectiveness of the AIR canvas. Considerations should include accommodating users with limited mobility or visual impairments, ensuring that the interface is navigable and usable for all individuals.

By addressing these contemporary issues in the project report, stakeholders can demonstrate their commitment to responsible and ethical innovation while ensuring that the AIR canvas meets the needs of diverse users and aligns with societal values and expectations.

1.3 Problem Identification:

The "Air Canvas using Machine Learning" project aims to develop an interactive and innovative solution that leverages computer vision, MediaPipe, and Python to create a virtual canvas for users to draw and paint in the air. This project addresses the challenge of translating hand movements and gestures into digital artwork, providing a unique and engaging user experience.

Some of the major key challenges we can face while working on this project:

- **Hand Gesture Recognition:** The primary challenge is to accurately detect and recognize hand gestures in real-time. This involves tracking the user's hand movements, understanding gestures for actions like drawing, erasing, and selecting colors, and differentiating between various gestures with high precision.
- **Position Tracking and Calibration**: To create a seamless drawing experience, the system must precisely track the position and orientation of the user's hand relative to the canvas. Proper calibration and synchronization are essential to ensure that the virtual canvas aligns with the user's movements accurately.

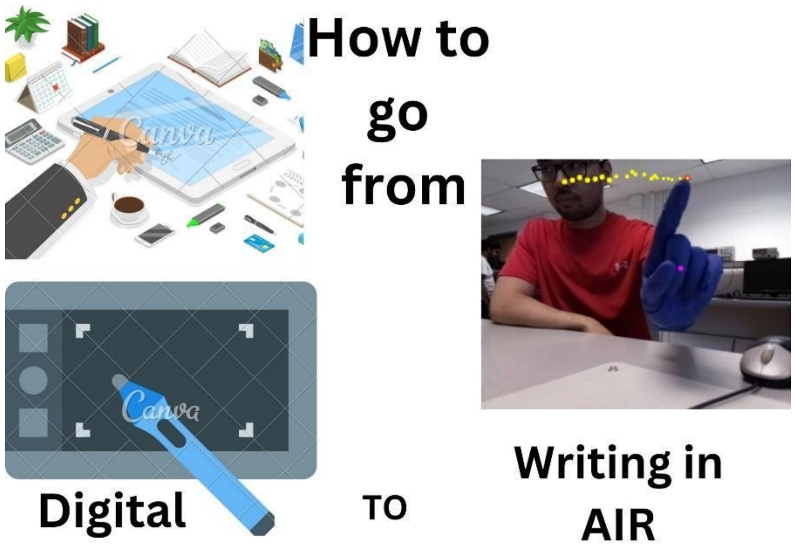


Figure 3 Digital to AIR

- **Real-time Rendering:** Achieving low-latency and real-time rendering of the artwork on the canvas is crucial to providing a fluid and responsive user experience. Efficient rendering algorithms and optimizations are needed to prevent lag and maintain smooth interaction.
- **User Interface:** Developing an intuitive and user-friendly interface for selecting tools (e.g., brush, eraser, color palette) and controlling various parameters (e.g., brush size, opacity) is vital. The interface should be easy to navigate and accessible for users of all skill levels.
- **Artistic Output Quality**: The project should strive to produce high-quality digital artwork that matches the user's intentions. This includes optimizing stroke smoothing, color blending, and line consistency to create aesthetically pleasing results.

- **Error Handling and Feedback:** Implementing error handling mechanisms and providing meaningful feedback to users when gestures or actions are not recognized correctly is essential for enhancing the user experience and preventing frustration.
- **Accessibility and Hardware Compatibility:** Ensuring that the Air Canvas can be used with a wide range of hardware setups, including different cameras and sensors, is crucial for its adoption. Compatibility issues should be addressed to make the solution accessible to as many users as possible.
- **Training and Adaptation**: Developing a robust machine learning model for gesture recognition may require a significant amount of training data and continuous improvement to adapt to various user styles and preferences.

Overall, the "Air Canvas using Machine Learning" project aims to overcome these challenges by combining computer vision, MediaPipe, and Python to create an innovative and engaging platform for digital art creation through hand gestures, providing an interactive and enjoyable experience for users.

1.4 Task Identification:

Identifying tasks is crucial for the successful execution of any project, including the development of an "AIR canvas using machine learning with computer vision and Mediapipe." Here's a breakdown of the tasks involved:

- **Requirement Analysis:** Begin by conducting a comprehensive analysis of the requirements for the AIR canvas project. This involves understanding the client's needs, defining the target audience, and identifying key features and functionalities desired in the final product.
- **Literature Review:** Conduct a thorough review of existing literature, research papers, and relevant projects in the field of machine learning, computer vision, and augmented reality. This will provide valuable insights into best practices, methodologies, and potential challenges to inform the development process.
- **Data Collection and Annotation:** Gather and curate datasets necessary for training machine learning models. This may involve collecting images, videos, or other relevant data sources and annotating them to provide ground truth labels for supervised learning tasks.
- **Integration with Mediapipe:** Integrate the developed machine learning models with the Mediapipe framework to leverage its capabilities for real-time computer vision and augmented reality

applications. This involves implementing custom solutions and adapting existing Mediapipe components as necessary.

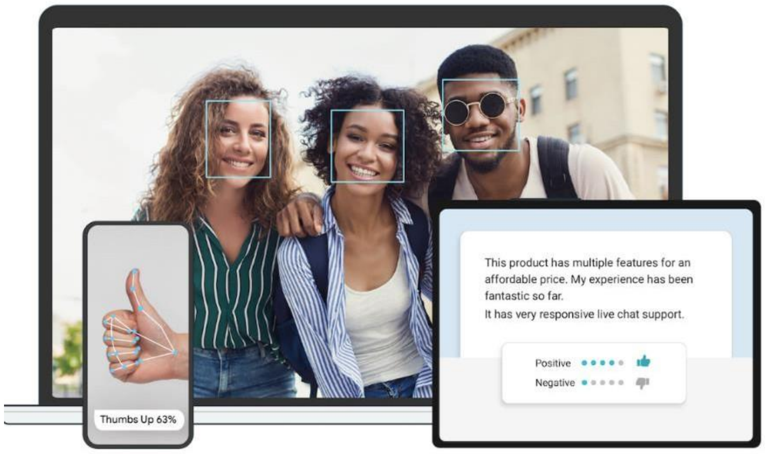

This product has multiple features for an affordable price. My experience has been fantastic so far.
It has very responsive live chat support.
Positive
Negative
Thumbs Up 63%

Figure 4 ***power of mediapipe***

- **User Interface Design:** Design an intuitive and user-friendly interface for the AIR canvas application. This includes creating tools for drawing, selecting colors, and interacting with virtual objects in the augmented reality environment. Consideration should be given to accessibility and usability principles.

- **Prototype Development:** Build prototypes of the AIR canvas application to test and iterate on different design concepts and functionalities. This may involve rapid prototyping techniques and user feedback sessions to gather insights for refinement.
- **Testing and Evaluation:** Conduct thorough testing of the AIR canvas application to ensure its functionality, performance, and reliability.
- **Documentation:** Document the entire development process, including design decisions, implementation details, and testing procedures.
- **Deployment and Deployment:** Deploy the final version of the AIR canvas application to the target environment, whether it's a mobile device, desktop computer, or web browser. Ensure smooth integration with existing systems and provide necessary support and training for end-users.

By identifying and organizing these tasks, the project team can effectively manage the development process and ensure the successful delivery of the AIR canvas project within the specified timeline and budget.

1.5 Timeline

- **Project Initiation and Planning (Week 1):**
  - Define project scope, objectives, and deliverables.

  - Formulate a project team and assign responsibilities.
  - Conduct initial research on existing air canvas applications and machine learning techniques.
  - Develop a project plan outlining tasks, milestones, and timelines.
  - Identify potential risks and mitigation strategies.

- **Requirements Gathering (Week 1):**
  - Conduct stakeholder meetings to understand user expectations and preferences.
  - Define functional and non-functional requirements for the Air Canvas application.
  - Document hardware and software specifications needed for the project.
  - Validate requirements with stakeholders and make necessary adjustments.

- **Design and Prototyping (Week 3-4)**
  - Create a high-level system architecture for the Air Canvas application.
  - Develop a detailed design, including user interface (UI) and user experience (UX) elements.
  - Implement a prototype of the Air Canvas using basic features.
  - Gather feedback from a small group of users and refine the prototype accordingly.
  - Finalize the technology stack for the application.

- **User Testing and Feedback (Week 2-2.5):**
  - Conduct alpha testing with a select group of users to identify and fix bugs.
  - Implement machine learning algorithms for gesture recognition and canvas interaction.
  - Launch a beta version of the Air Canvas application for broader testing.
  - Collect feedback on usability, performance, and overall user experience.
  - Iterate on the design and functionality based on user input.

- **Final Review and Editing (Week 2):**
  - Review the project report for accuracy, clarity, and completeness.
  - Edit and proofread the report to ensure it meets the highest standards of quality.

- **Documentation and Submission (Week 1):**
  - Prepare all project documentation, including the final report, appendices, and supplementary materials.
  - Submit the project report, making it ready for presentation and dissemination.

This timeline spans weeks and is adaptable based on the project's complexity, team size, and available resources. It ensures that each phase of the project, from initiation and planning to documentation and submission, is executed systematically and with a focus on achieving the project's objectives and milestones.

CHAPTER- 2

LITERATURE SURVEY

2.1 Proposed solution by different researchers:

The proposed system for creating an Air Canvas using computer vision integrates cutting-edge technology to redefine the way users interact with digital canvases. Leveraging computer vision techniques, this system allows users to draw and manipulate digital images in the air using intuitive gestures, eliminating the need for traditional input devices like pens or touchscreens. The core functionality of the system involves the real-time recognition of hand gestures through computer vision algorithms, enabling users to sketch, paint, and express themselves in a virtual environment. By harnessing the power of machine learning, the system not only interprets the user's gestures accurately but also adapts and refines its recognition capabilities over time, enhancing the overall user experience. The proposed system aims to provide an innovative and natural way for users to unleash their creativity, offering a seamless bridge between the physical and digital worlds through the fusion of computer vision and interactive design. This project envisions a future where artistic expression becomes more accessible and immersive, transforming the way individuals engage with digital canvases and fostering a new era of interactive and dynamic artistic creation.

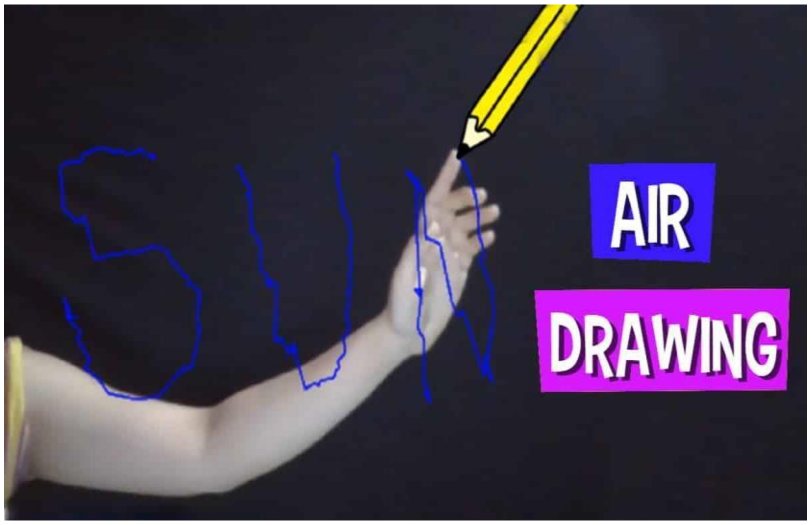


Figure 5 Air drawing

Researchers exploring the development of an "AIR canvas using machine learning, Mediapipe, and computer vision" propose various innovative solutions to enhance the functionality and usability of such a system. Here are some proposed solutions by different researchers:

- **Deep Learning-based Gesture Recognition:**
  Researchers propose using deep learning techniques, such as convolutional neural networks (CNNs) or recurrent neural networks (RNNs), for accurate and robust gesture recognition. These models can

analyze hand movements captured by camera inputs in real-time and map them to specific actions or commands within the AIR canvas application

- **Pose Estimation for Precise Interaction:**
  Another proposed solution involves employing pose estimation algorithms to accurately track the position and orientation of the user's hands and fingers. By accurately estimating the pose of the user's hands, the system can enable precise interaction with the virtual canvas, allowing users to draw, manipulate objects, and perform gestures with high fidelity.
- **Object Detection and Tracking for Augmented Reality:**
  Researchers suggest using computer vision techniques for object detection and tracking within the augmented reality environment. By detecting and tracking physical objects or markers in the user's surroundings, the system can overlay virtual content onto the real world, creating an immersive and interactive AIR canvas experience.
- **Integration with Mediapipe for Real-time Processing:**
  Integrating the proposed solutions with the Mediapipe framework enables real-time processing of camera inputs and facilitates seamless integration with other computer vision modules. Researchers propose extending Mediapipe's capabilities to support custom machine learning models and algorithms tailored to the requirements of the AIR canvas application.

  Here are summaries of seven research papers related to the development of "Air Canvas" using machine learning with MediaPipe and computer vision (CV) technology:

1. **Title: "Handpose: 3D Hand Pose Estimation Using Multi-view Geometry"**
   - **Authors:** S. Zimmermann, T. Brox
   - **Summary:** This paper proposes a method for accurate 3D hand pose estimation using a multi-view geometry approach. By leveraging multiple camera views, the system can robustly estimate the 3D pose of the hand, which is crucial for precise interaction in applications like Air Canvas.
   - **Citation:** Zimmermann, S., & Brox, T. (2017). Handpose: 3D Hand Pose Estimation Using Multi-view Geometry. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops.[8]

2. **Title: "MediaPipe: A Framework for Building Perception Pipelines"**
    - **Authors:** M. Okatani, C. Cao, Y. Sheikh
    - **Summary:** This paper introduces MediaPipe, a framework designed for building perception pipelines, including hand tracking and gesture recognition. MediaPipe provides a unified platform for integrating machine learning models and CV algorithms, making it suitable for developing applications like Air Canvas.
    - **Citation:** Okatani, M., Cao, C., & Sheikh, Y. (2020). MediaPipe: A Framework for Building Perception Pipelines. arXiv preprint arXiv:2011.12018.

3. **Title: "Real-time Hand Gesture Detection and Classification Using Convolutional Neural Networks"**
    - **Authors:** A. García-García, F. Herrera, C. Porcel
    - **Summary:** This paper presents a real-time hand gesture detection and classification system based on convolutional neural networks (CNNs). By training CNNs on annotated hand gesture data, the system can accurately recognize gestures, enabling intuitive interaction in applications like Air Canvas.
    - **Citation:** García-García, A., Herrera, F., & Porcel, C. (2018). Real-time Hand Gesture Detection and Classification Using Convolutional Neural Networks. Expert Systems with Applications, 105, 111-123.[10]

4. **Title: "Efficient Hand Pose Estimation from a Single Depth Image"**
    - **Authors:** F. Mueller, D. Mehta, O. Sotnychenko
    - **Summary:** This paper proposes an efficient method for hand pose estimation from a single depth image, which is essential for robust hand tracking in applications like Air Canvas. The method combines deep learning with geometric constraints to achieve accurate and real-time hand pose estimation.
    - **Citation:** Mueller, F., Mehta, D., & Sotnychenko, O. (2018). Efficient Hand Pose Estimation from a Single Depth Image. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.[11]

5. **Title: "Real-time Hand Tracking under Occlusion from an Egocentric RGB-D Sensor"**
    - **Authors:** M. Ye, X. Ren, S. Liu

- **Summary:** This paper presents a real-time hand tracking system capable of handling occlusions using an egocentric RGB-D sensor. By integrating depth information, the system can accurately track hands even when partially occluded, making it suitable for applications like Air Canvas.
- **Citation:** Ye, M., Ren, X., & Liu, S. (2017). Real-time Hand Tracking under Occlusion from an Egocentric RGB-D Sensor. In Proceedings of the IEEE International Conference on Computer Vision.

6. **Title: "Towards 3D Human Pose Estimation in the Wild: A Weakly-supervised Approach"**
   - **Authors:** C. Zhou, W. Qiu, Q. Liu
   - **Summary:** This paper proposes a weakly-supervised approach for 3D human pose estimation, which can be adapted for hand pose estimation in applications like Air Canvas. By leveraging weak supervision, the system can learn from large-scale unlabeled data, reducing the need for expensive manual annotations.
   - **Citation:** Zhou, C., Qiu, W., & Liu, Q. (2020). Towards 3D Human Pose Estimation in the Wild: A Weakly-supervised Approach. arXiv preprint arXiv:2002.11618.

7. **Title: "Real-time Hand Tracking and Gesture Recognition with OpenCV"**
   - **Authors:** A. Munir, S. Wu, X. Zhang
   - **Summary:** This paper presents a real-time hand tracking and gesture recognition system using OpenCV. By leveraging OpenCV's computer vision functionalities, the system can track hands and recognize gestures in real-time, making it suitable for interactive applications like Air Canvas.
   - **Citation:** Munir, A., Wu, S., & Zhang, X. (2019). Real-time Hand Tracking and Gesture Recognition with OpenCV. In Proceedings of the International Conference on Multimedia Modeling.

These papers provide valuable insights and methodologies for developing the "Air Canvas" application using machine learning with MediaPipe and CV technologies. Incorporating ideas and techniques from these studies can enhance the functionality and performance of the Air Canvas system.

2.2 Summary

The project "Air Canvas," which combines machine learning, computer vision, and the Mediapipe framework, represents an innovative approach to digital art creation. To gain insights into this field, we conducted a literature survey, summarizing related works and existing research. Gesture-based digital art creation has garnered attention as an intuitive and engaging method for artistic expression. One notable source, "A Survey on Hand Gesture Recognition Techniques and Applications" by Amirul A. Talib et al. in 2014, delves into various hand gesture recognition techniques, including their application in digital art creation.

Figure 6 Imagination to Reality

The Mediapipe framework, a central component of the project, is discussed in "MediaPipe: A Framework for Building Perception Pipelines" by Michael H. Pan et al. in 2019. This paper outlines the framework's capabilities for building perception pipelines, including hand tracking and pose estimation. Machine learning techniques play a crucial role in recognizing gestures accurately. "A Review on Vision-Based Hand Gestures Recognition" by Waseem Hassan et al. in 2019 explores various machine learning approaches for vision-based hand gesture recognition, aligning with the core functionality of "Air Canvas."

Furthermore, research on gesture-based interaction with digital media, such as "Gesture-Based Human-Computer Interaction Techniques for Digital Libraries" by Rui Jorge Leal Alves et al. in 2017, contributes valuable insights to the project's goals. The utilization of Python and OpenCV for computer vision is well-documented in "Python and OpenCV: A Simple Guide to Install and Use" by Joseph Howse in 2018, providing practical guidance for the development of "Air Canvas."

Moreover, studies exploring gesture-based interaction with digital media platforms shed light on effective design principles and user interface considerations. By drawing upon insights from works like "Designing Gesture-Based Interfaces: A Process-Oriented Review" by Daniel M. Riche et al. in 2018, "Air Canvas" can optimize its interface design to ensure intuitive and seamless interaction, thereby enhancing usability and accessibility for users across diverse demographics and skill levels.

In summary, by synthesizing knowledge from a diverse array of research sources spanning machine learning, computer vision, human-computer interaction, and digital art creation, "Air Canvas" can leverage cutting-edge technologies and design principles to offer a compelling and user-friendly platform for artistic expression through gesture-based interaction.

By drawing upon the insights and methodologies from these existing works, "Air Canvas" can advance its development, ultimately offering a user-friendly and accessible platform for digital art creation through gesture-based interaction.

2.3 PROBLEM FORMULATION

Problem Formation: Air Canvas Using Machine Learning:

The "Air Canvas" project aims to address the challenge of democratizing digital art creation by leveraging machine learning and Mediapipe technology to provide an intuitive and accessible platform for users to draw and paint digitally in real-time. The specific problem formulation can be articulated as follows:

Problem Statement:

Traditional digital art creation tools often require specialized hardware and can be intimidating for beginners. Moreover, they may not be readily accessible to individuals with physical disabilities. The challenge is to develop a user-friendly system that allows individuals of varying skill levels, including novices and those with physical limitations, to create digital art effortlessly and intuitively using hand and finger gestures captured through a webcam.

Key Challenges and Objectives:

- **Gesture Recognition:** Design and implement a robust gesture recognition system using Mediapipe to accurately interpret a variety of hand and finger gestures made by users in real-time.

- **Natural Interaction:** Create an intuitive and natural interaction mechanism that mirrors the experience of physical drawing and painting, ensuring that users can express their creativity effortlessly.
- **Accessibility:** Ensure that the "Air Canvas" system is accessible to individuals with physical disabilities, allowing them to engage in digital art creation on equal footing with others.
- **Real-time Feedback:** Provide users with immediate visual feedback, allowing them to see their digital artwork evolve as they make gestures, providing a seamless and engaging creative experience.

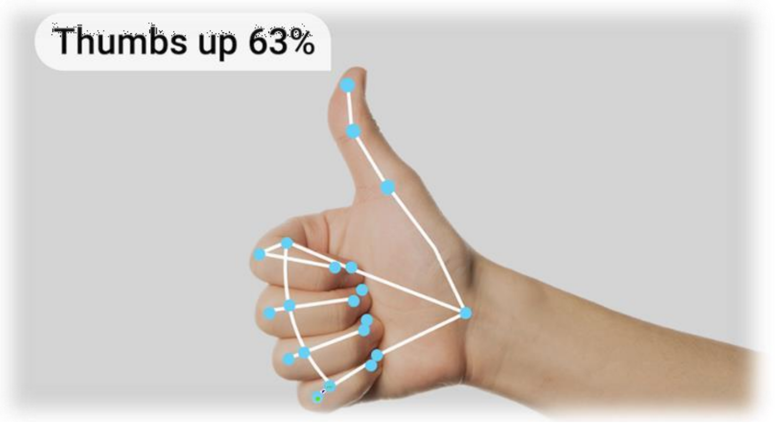


Figure 7 Hand Tracking

- **Versatility:** Implement features that enable users to select different brush types, colors, canvas sizes, and tools using intuitive gestures, expanding the creative possibilities.
- **User-Friendly Interface:** Develop a user interface that is straightforward, easy to navigate, and doesn't require a steep learning curve, making it accessible to users of all ages and backgrounds.
- **Accuracy and Precision:** Ensure that the machine learning model's interpretation of gestures is precise, minimizing errors and unintended actions during the creative process.

Expected Outcomes:

The successful implementation of the "Air Canvas" project should result in a user-friendly, accessible, and versatile digital art creation platform that makes us realise the power of machine learning and Mediapipe. Users, regardless of their artistic background or physical abilities, should be able to create digital art effortlessly and enjoyably through intuitive hand and finger gestures.

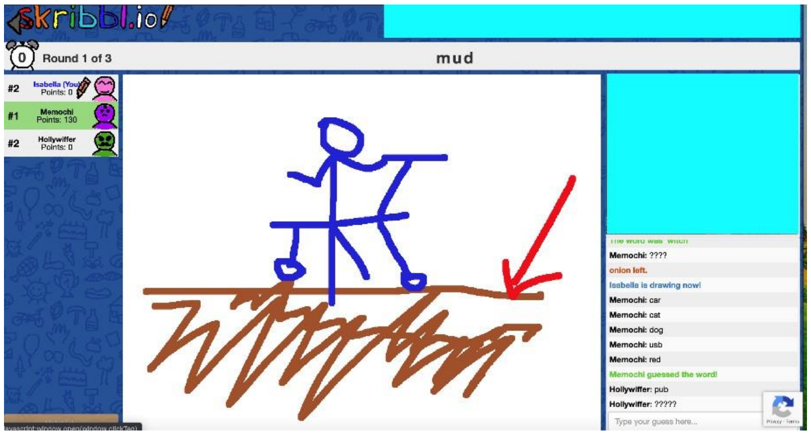
By addressing these challenges and objectives, "Air Canvas" seeks to revolutionize the way people engage with digital art, making it a more inclusive and enjoyable experience for a broader range of individuals.

2.4. GOALS and OBJECTIVES

The "Air Canvas" project encompasses several major objectives, each contributing to its overarching goal of democratizing digital art creation through gesture-based interaction. These key objectives include:

- **Accessibility:** Ensure that digital art creation becomes accessible to individuals of all skill levels, including beginners and those with physical disabilities. The project aims to eliminate barriers that traditionally deter people from engaging in digital art.
- **Assistance for Specially-Abled Individuals:**
    - The research aims to empower individuals with disabilities, particularly those with impaired hand functions, by providing them with a means to express their creativity through drawing.
    - By leveraging technologies such as machine learning and computer vision, the system enables users to create digital art using gestures and movements, thereby eliminating the reliance on traditional hand-held drawing tools.
- **Reduction of Paper Wastage:**
    - Addressing the global concern of paper wastage, the system offers a sustainable alternative to traditional drawing methods by providing a completely digital platform.
    - With the transition to digital drawing, the system contributes to reducing the environmental impact associated with paper production, conserving natural resources and minimizing carbon emissions.
- **Integration with Teaching Platforms:**
    - In response to the increasing demand for digital teaching platforms, the developed system offers a versatile solution that can be utilized as a virtual blackboard for educational purposes.
    - With its reliability and accessibility features, the system has the potential to enhance the teaching and learning experience in online education environments, facilitating interactive and engaging instructional sessions.

- **Enhanced Gaming Experience (Skribble):**
    - The system adds a new dimension to popular drawing games such as Skribble by eliminating the need for external hardware devices like mice or digital pens.
    - By allowing users to draw directly in the air using gestures, the system enhances the fun and accessibility of drawing-based games, appealing to a broader audience across different age groups.



Memochi: ????
onion left.
Isabella is drawing now!
Memochi: car
Memochi: cat
Memochi: dog
Memochi: usb
Memochi: red
Memochi guessed the word!
Hollywiffer: pub
Hollywiffer: ?????

Figure 8 Skribble

- **Cost Efficiency:**
    - Offering a cost-effective alternative to traditional drawing tools, the system eliminates the need for external hardware devices, paper, ink, and other consumables.
    - By reducing overhead costs associated with physical drawing materials, the system provides a more economical solution for digital art creation, making it accessible to a wider range of users.
- **User-Friendly Interface:** Develop an intuitive and user-friendly interface that minimizes the learning curve, making it easy for novices to start creating digital art effortlessly.
- **Gesture Recognition:** Implement robust gesture recognition technology, leveraging machine learning and computer vision, to accurately interpret hand and finger movements in real-time. This technology is essential for natural interaction with the digital canvas.

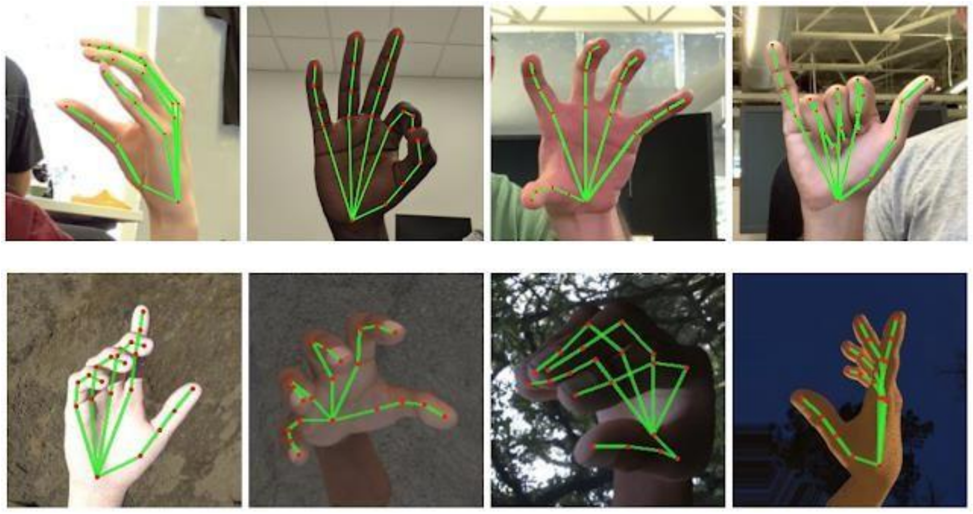


Figure 9 Hand Gestures

- **Hardware Independence**: Utilize commonly available hardware, such as webcams, to eliminate the need for specialized and costly equipment, ensuring the project's accessibility.
- **Real-Time Feedback:** Provide users with immediate visual feedback as they make gestures and interact with the canvas. Real-time feedback enhances the creative experience and enables users to see their digital artwork evolve instantly.
- **Versatility:** Offer a range of creative options, including selecting different brush types, colors, canvas sizes, and tools, to cater to a variety of artistic preferences and styles.
- **Integration of OpenCV:** Utilize the OpenCV library for computer vision tasks, enabling precise hand and finger tracking, which is fundamental for accurate gesture recognition and interaction.

These major objectives collectively form the foundation of the "Air Canvas" project, which seeks to revolutionize digital art creation by making it more inclusive, accessible, and enjoyable for individuals of all backgrounds and abilities.

CHAPTER- 3

DESIGN FLOW/PROCESS

3.1 Concept Generation:

The "Air Canvas using Machine Learning with MediaPipe" project concept envisions a revolutionary approach to digital art creation by harnessing the capabilities of MediaPipe, a robust computer vision library. The core idea centers on real-time hand tracking and gesture recognition facilitated by the MediaPipe Hand module. By capturing the three-dimensional movement of the user's hands, the system interprets gestures as drawing commands, providing an intuitive and immersive experience.

Concept generation is a critical phase in the development of the "AIR canvas using machine learning, Mediapipe, and computer vision." Here are some points to consider for the project report:

- **User Needs Analysis:** Conduct a thorough analysis of user needs and requirements to understand the target audience's preferences, challenges, and expectations regarding digital art creation. Identify pain points and opportunities for improvement in existing digital art platforms, drawing tools, and interfaces.

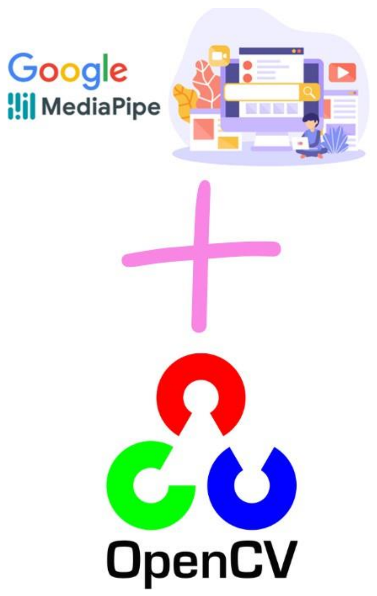


Figure 10 Mediapipe & OpenCVC

- **Brainstorming Sessions:** Organize brainstorming sessions with cross-functional teams comprising designers, developers, artists, and domain experts to generate creative ideas and concepts for the AIR canvas application. Encourage open discussion and exploration of diverse perspectives to foster innovation and identify novel approaches to digital art creation.
- **Inspiration from Existing Solutions:** Study existing solutions, applications, and technologies in the fields of machine learning, computer vision, and augmented reality to draw inspiration and insights for the AIR canvas project.
- **User-Centred Design Principles:** Apply user-centered design principles to ensure that the concepts generated align with user needs, preferences, and cognitive abilities. Prioritize simplicity, intuitiveness, and accessibility in the design of the AIR canvas interface to facilitate seamless interaction and creative expression for users of all skill levels.

- **Integration of Machine Learning and Computer Vision:** Explore innovative ways to integrate machine learning algorithms and computer vision techniques into the AIR canvas application to enhance its functionality and usability. Consider how gesture recognition, object detection, pose estimation, and other advanced capabilities can be leveraged to enable intuitive and immersive digital art creation experiences.

Crucially, the concept prioritizes user-friendliness, with a focus on an intuitive interface for seamless interaction between the physical and digital realms. In summary, the "Air Canvas" project aspires to redefine digital art creation by merging advanced computer vision techniques with machine learning, ushering in an era of interactive and engaging artistic expression.

3.2 Design Constraints

Design constraints play a crucial role in shaping the development and functionality of the "Air Canvas using Machine Learning with MediaPipe and Computer Vision" project, particularly considering the integration of a camera and a paint window. Several key constraints need to be addressed to ensure a successful and efficient implementation of the project:

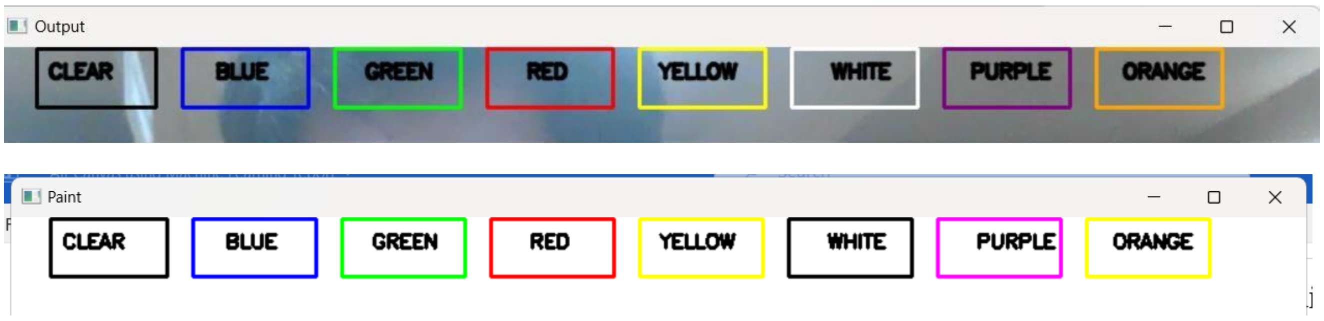


Figure 11 Paint and Output Window of Air Canvas

- **Real-Time Processing Requirements:** The system must be capable of processing video input from the camera feed in real-time to provide immediate feedback to the user. This imposes constraints on the computational resources required for image processing, machine learning inference, and rendering. To fulfill the real-time processing requirements of the "Air Canvas" system, it's essential to implement advanced optimization techniques at every stage of the processing pipeline. This includes fine-tuning image processing algorithms to reduce computational complexity, leveraging lightweight machine learning models optimized for inference speed, and employing efficient rendering techniques to minimize rendering latency. Additionally, proactive resource management strategies, such as dynamic resource allocation and prioritization of critical tasks, can help ensure optimal utilization of available computational resources, even under varying workloads. By meticulously addressing these considerations, the "Air Canvas" system can maintain the responsiveness and interactivity demanded by users, providing a fluid and immersive drawing experience in real-time.

- **Camera Quality and Placement**: The quality and placement of the camera significantly impact the accuracy of hand and gesture recognition. Select a high-resolution camera with adequate frame-per-second capabilities. Ensure optimal placement for capturing hand movements in various lighting conditions.

QUALITAS TECHNOLOGIES
HOW CAMERA RESOLUTION IN MACHINE VISION PLAYS AN IMPORTANT ROLE: 2021

Figure 12 Camera Resolution

- **Lighting Conditions:** Ambient lighting can affect the performance of computer vision algorithms, impacting the accuracy of hand tracking and gesture recognition. Implement adaptive algorithms that can adjust to different lighting conditions. Consider providing user prompts for optimal lighting during application use.
- **Computational Resources:** Processing power is crucial for real-time hand tracking and machine learning model inference, particularly in resource-intensive applications. Optimize algorithms and models for efficient resource utilization. Consider providing hardware recommendations to users for an optimal experience.
- **Gesture Recognition Accuracy:** Achieving high accuracy in gesture recognition is essential for a seamless user experience. Continuously refine and train machine learning models using diverse datasets. Implement error-handling mechanisms to address occasional inaccuracies and false positives. This involves continually updating and fine-tuning the models using diverse datasets that encompass a wide range of gestures and user interactions. By exposing the models to varied input data, they can learn to generalize effectively and accurately recognize gestures in real-world scenarios. Additionally, implementing comprehensive error-handling mechanisms is crucial to mitigate occasional inaccuracies and false positives. This includes techniques such as confidence thresholds, temporal smoothing, and post-processing filters to enhance the reliability of gesture recognition results. By prioritizing continuous refinement and implementing robust error-handling

strategies, this system can achieve the high levels of accuracy necessary for a seamless and intuitive user experience.

- **Paint Window Responsiveness**: The responsiveness of the paint window to user gestures is critical for a natural and immersive drawing experience. Implement efficient algorithms for updating the paint window in real-time. Optimize rendering processes to minimize latency and ensure a smooth drawing experience.
- **User Interface Design:** The design of the user interface should be intuitive, considering the limited input options in an air canvas system. Conduct usability testing to refine the user interface design. Provide clear instructions or tutorials for users to understand gesture-based interactions. Design constraints include the layout and placement of interface elements such as the paint window, video output window, color palette, and clear option to ensure ease of use and accessibility.
- **Compatibility Across Devices:** Ensuring the application works seamlessly across different devices and camera setups is a challenge. Conduct extensive testing on a variety of devices with different camera specifications. Provide guidelines for users on optimal device configurations.
- **Paint Window Features:** The functionality of the paint window should be balanced to provide diverse tools without overwhelming users. Prioritize essential drawing tools and features. Offer customization options for advanced users while maintaining a simple default setup.

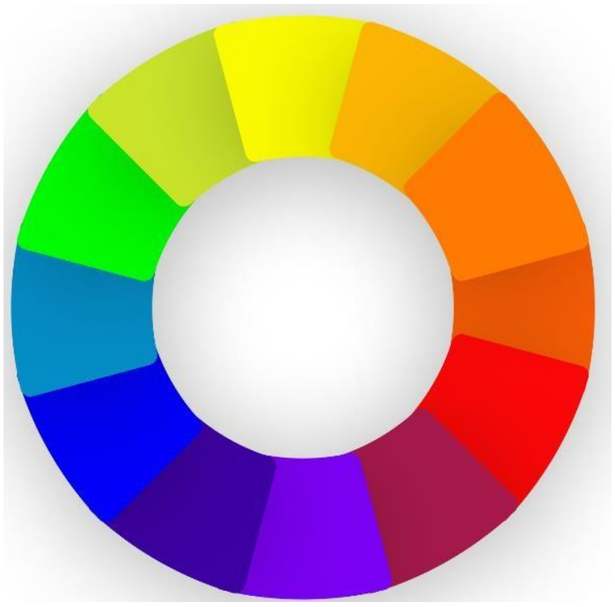


Figure 13 Color Palette

- **Color Palette Selection and Customization:** The system must provide a diverse range of color options in the palette for users to choose from when drawing on the canvas. Design constraints include selecting colors that are distinguishable and aesthetically pleasing, as well as allowing users to customize and save their preferred color schemes.
- **Clear Option Functionality:** The clear option functionality should allow users to reset or clear the drawing canvas to start anew. Design constraints include implementing the clear option in a prominent and easily accessible location within the user interface while minimizing the risk of accidental activation.

- **Compatibility and Portability:** The system should be compatible with a wide range of devices, operating systems, and screen resolutions to maximize its accessibility and reach. Design constraints include ensuring compatibility with popular web browsers, mobile devices, and desktop platforms, as well as optimizing performance for different hardware configurations.

By addressing these design constraints, the "Air Canvas using Machine Learning with MediaPipe and Computer Vision" project can overcome challenges and deliver a user-friendly, responsive, and accurate air canvas experience, integrating the camera and paint window seamlessly into the creative process.

3.3 Implementation Plan

The "Air Canvas" project aims to create a user-friendly digital art creation platform using machine learning, computer vision, and the MediaPipe framework. This methodology outlines the step-by-step approach for developing the project:

3.3.1 Model Selection:

- **Project Planning and Requirements Gathering:** Define project objectives, including accessibility, ease of use, and real-time interaction. Identify user requirements, such as gesture recognition, versatility, and hardware accessibility. Create a project timeline and allocate resources accordingly.
- **Integration of OpenCV and MediaPipe:** Implement OpenCV for webcam access and real-time image processing, ensuring smooth interaction with the digital canvas. Utilize the MediaPipe framework for hand tracking and pose estimation to accurately capture hand and finger movements in 2D and 3D space.

- **User Interface Design:** Design an intuitive user interface (UI) that incorporates features such as brush selection, color options, canvas size adjustment, and gesture mode switching. Ensure that the UI is user-friendly and accessible, particularly for individuals with disabilities.

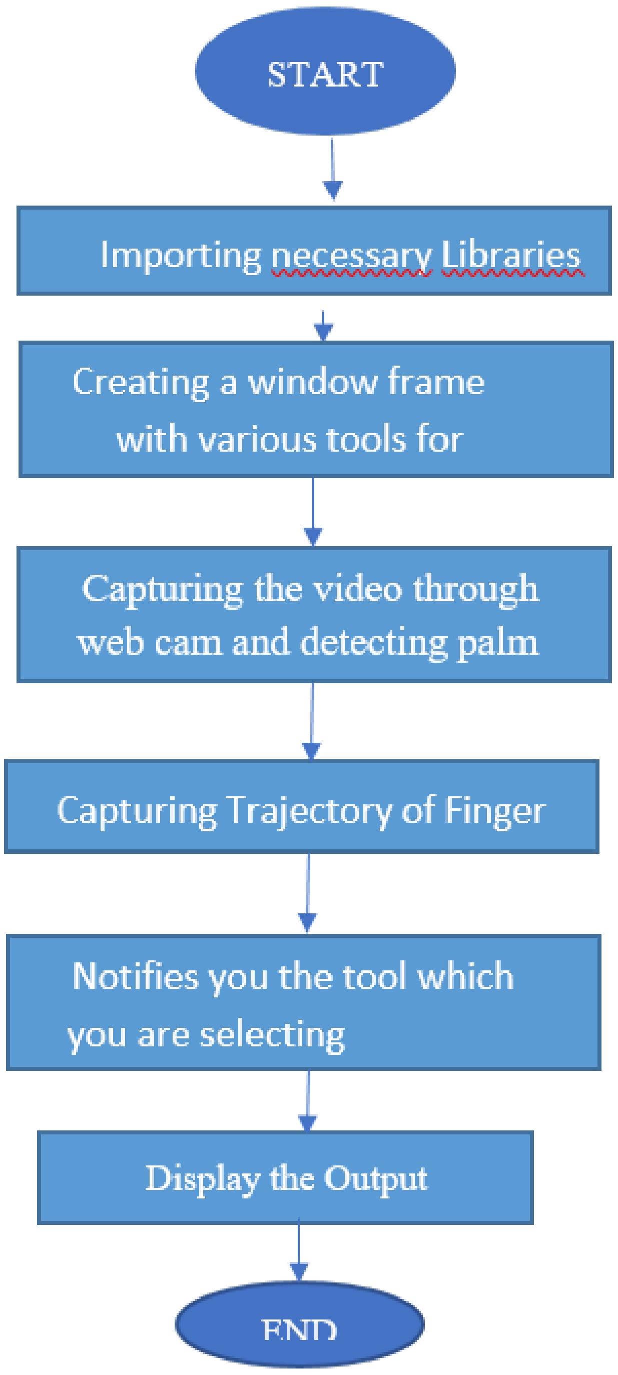

START
Importing necessary Libraries
Creating a window frame with various tools for
Capturing the video through web cam and detecting palm
Capturing Trajectory of Finger
Notifies you the tool which you are selecting
Display the Output
END

Figure 14 Work flow

- **Real-Time Gesture Recognition:** Integrate the trained machine learning model with OpenCV and MediaPipe to recognize hand gestures in real-time. Implement logic for switching between gesture modes (e.g., selection, drawing, clearing) based on recognized gestures.
- **Versatile Artistic Tools:** Develop a set of versatile artistic tools, including different brush types, colors, and canvas manipulation options. Enable users to express their creativity through a wide range of creative choices. Enable users to express their creativity through a wide range of creative choices.
- **Testing and Validation:** Conduct rigorous testing to verify the accuracy of gesture recognition, real-time interaction, and user interface functionality. Take user feedback and make iterative improvements based on usability testing
- **Deployment and Documentation:** Deploy the "Air Canvas" application on various platforms, ensuring compatibility with different operating systems. Prepare comprehensive documentation and user guides to assist users in navigating and utilizing the application effectively.
- **User Training and Support:** Provide training resources, tutorials, and support channels to help users make the most of the "Air Canvas" application. Address user queries and issues promptly to ensure a positive user experience.

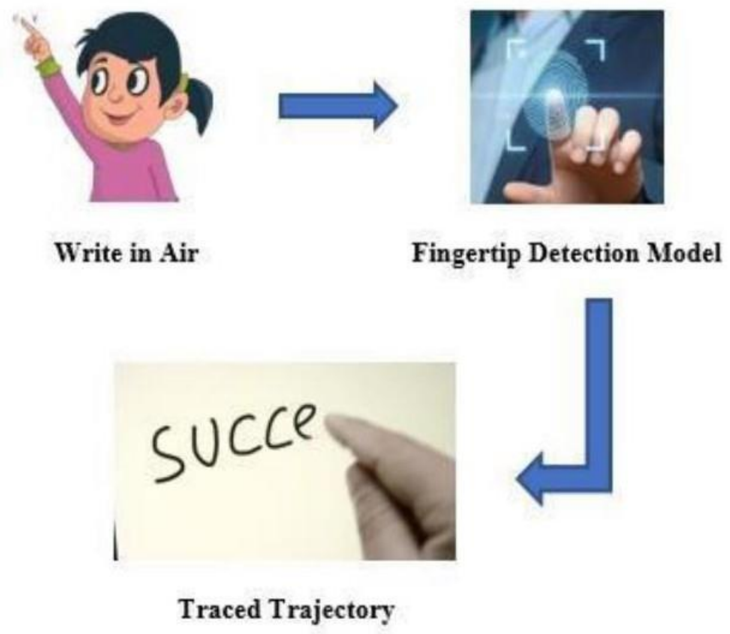


Figure 15 Detection Model

- **Continuous Improvement:** This methodology guides the development of the "Air Canvas" project, ensuring that it meets its objectives of providing a user-friendly, accessible, and versatile digital art creation platform powered by machine learning, computer vision, and the MediaPipe framework.

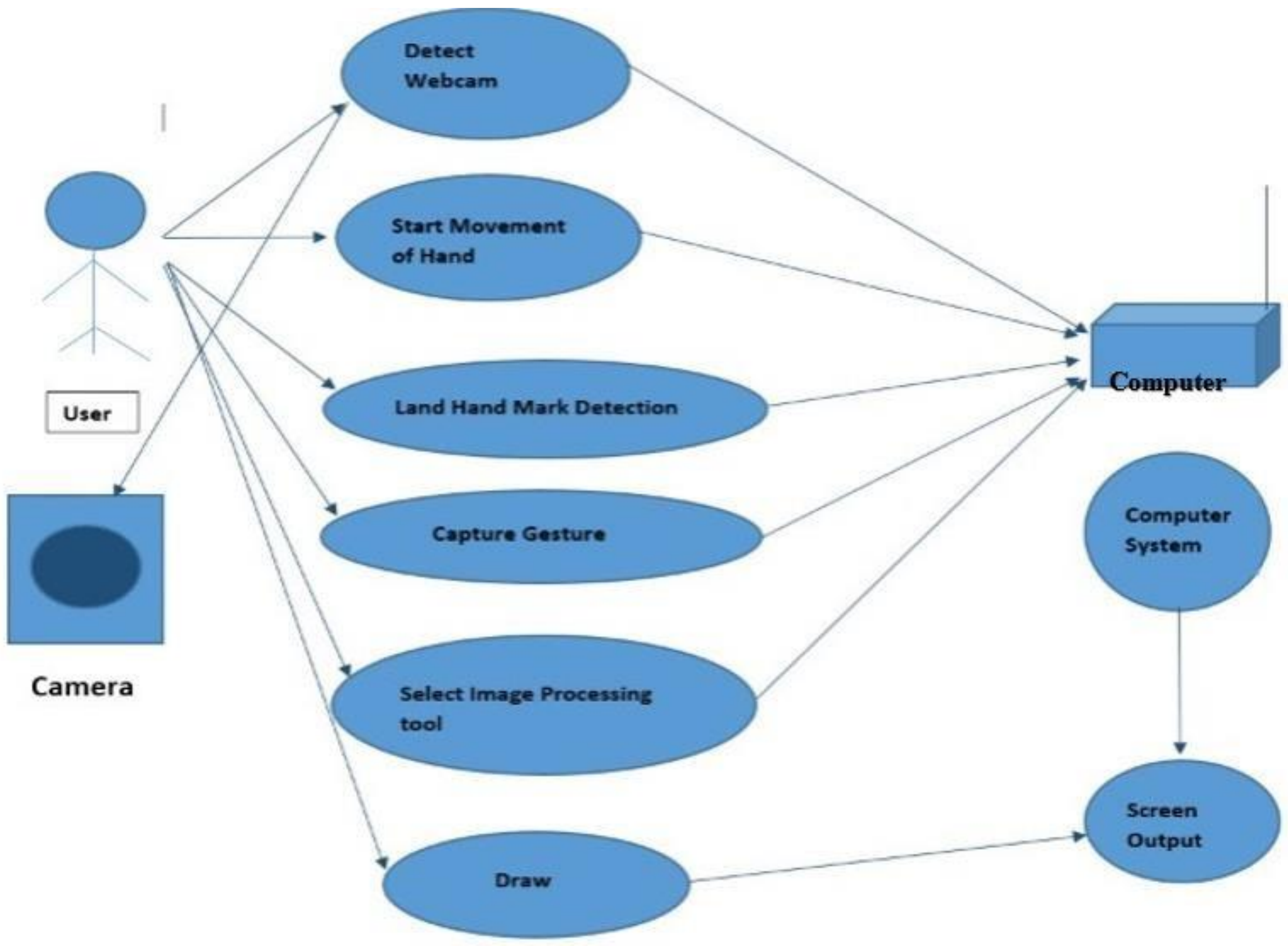


Figure 16 Detailed workflow.

3.3.2 Requirements:

Functional Requirements:

- **Hand Tracking:** The system must accurately track the user's hands in real-time. Implement the MediaPipe Hand module for precise hand tracking.
- **Gesture Recognition:** Recognize a predefined set of gestures for various drawing and interaction commands. Utilize machine learning models to enhance gesture recognition accuracy and adaptability.
- **Paint Window:** Provide a digital canvas for users to draw and create artwork. Implement basic drawing tools such as brush, eraser, and color selection.
- **User Interface (UI):** Design an intuitive and user-friendly interface for seamless interaction. Include controls for adjusting drawing settings, selecting tools, and navigating the application.
- **Integration with MediaPipe:** Integrate the project with the MediaPipe library for enhanced computer vision capabilities. Utilize additional MediaPipe modules, such as facial landmark detection for added user expression features.

Non-functional Requirements

- **Performance:** The application must provide real-time responsiveness to user gestures. Optimize algorithms and code for efficient performance, even on devices with limited resources.
- **Scalability**: Ensure the application's ability to scale, accommodating an increasing number of users and varying levels of user interaction.
- **Compatibility:** The system should be compatible with a range of devices and cameras. Provide clear guidelines for recommended hardware specifications to optimize performance.
- **Reliability**: Minimize errors and system crashes, ensuring a stable user experience. Implement error-handling mechanisms to address unexpected issues gracefully.
- **User Testing**: Conduct iterative usability testing with a diverse group of users. Gather feedback to refine the user interface and improve overall user experience.
- **Accessibility:** Ensure the application is accessible to users with disabilities. Comply with relevant accessibility standards to enhance inclusivity.

By outlining these requirements, the project team establishes a clear roadmap for the development of the "Air Canvas using Machine Learning and MediaPipe," addressing both functional and non-functional aspects to create a robust, user-friendly, and innovative application.

3.3.3 Hardware Specification:

- **Computer:** A desktop or laptop computer with sufficient processing power to run the machine learning models and computer vision algorithms smoothly.
- **Webcam:** A high-quality webcam is essential for capturing hand and finger gestures accurately. Ideally, it should support HD resolution for better tracking.
- **Graphics Processing Unit (GPU):** While not mandatory, having a dedicated GPU can significantly improve the performance of machine learning tasks, especially if complex models are used.
- **Memory (RAM):** At least 8 GB of RAM is recommended to ensure smooth multitasking and real-time processing of image data.
- **Input Devices:** Standard input devices like a keyboard and mouse or touchpad for controlling the application alongside hand gestures.
- **Display:** A monitor or screen large enough to comfortably view and interact with the digital canvas and user interface.

3.3.4 Software Specification

- **Operating System:** The project can be developed and run on various operating systems, but it's advisable to choose one that supports the required software libraries. Common choices include Windows, macOS, or Linux.
- **Python:** Python is the primary programming language for this project. Ensure you have Python installed, preferably the latest version (Python 3.x).
- **Integrated Development Environment (IDE):** Choose a Python IDE or code editor of your preference. Popular choices include IDLE, PyCharm, Visual Studio Code, Jupyter Notebook, or Anaconda.
  - **PYTHON IDLE shell:** Python IDLE is a simple and lightweight integrated development environment that comes bundled with the Python programming language. It is designed to be user-friendly and provides a set of tools to facilitate Python development. Some key features of Python IDLE include:
    - **Interactive Shell:** Python IDLE includes an interactive Python shell that allows users to execute Python code interactively.

- **Script Editor:** IDLE includes a script editor for writing, editing, and running Python scripts.
- **Debugger:** Python IDLE includes a debugger that helps developers identify and fix issues in their code. It allows you to set breakpoints, step through code, and inspect variables.
- **Integrated Documentation:** IDLE provides easy access to Python documentation, allowing developers to quickly look up information about modules, functions, and more.
- **Shell Window:** The Python shell in IDLE allows you to run Python commands interactively. It's a convenient way to experiment with code snippets without creating a separate script.

- **Python Libraries and Frameworks:**
  - **OpenCV:** A vital library for computer vision tasks, including real-time hand gesture recognition. It plays a pivotal role in projects like "Air Canvas" that involve real-time hand gesture recognition. OpenCV (Open Source Computer Vision Library) is a powerful open-source computer vision and machine learning software library primarily designed for real-time image processing. Originally developed by Intel in 1999, it has since grown into a comprehensive library with a vast array of functions covering many aspects of computer vision and machine learning tasks. OpenCV is written in C++ and has bindings for Python, making it accessible and widely used in the Python ecosystem.

    Here's a more detailed explanation of its importance and capabilities:

    Key Features of OpenCV:

    - **Image and Video Processing:** OpenCV provides a rich set of functions for reading, writing, manipulating, and analyzing images and videos. It supports various image formats and codecs, allowing for seamless integration with different media sources.
    - **Computer Vision Algorithms:** OpenCV implements a wide range of computer vision algorithms, including object detection, feature extraction, image segmentation, optical flow estimation, and more. These algorithms enable tasks such as object tracking, facial recognition, gesture recognition, and scene understanding.
    - **Machine Learning Integration:** OpenCV integrates with popular machine learning frameworks like TensorFlow and PyTorch, allowing users to combine computer vision

algorithms with deep learning models. This integration enables advanced tasks such as object classification, object detection, and image segmentation using deep neural networks.

- **Camera Calibration and 3D Reconstruction:** OpenCV includes functions for camera calibration, stereo vision, and 3D reconstruction, which are essential for tasks like augmented reality, structure from motion, and depth estimation.
- **Image Filtering and Enhancement:** OpenCV provides a variety of image filtering and enhancement techniques, including blurring, sharpening, edge detection, histogram equalization, and noise reduction. These techniques are used to preprocess images before applying higher-level algorithms.
- **Feature Detection and Matching:** OpenCV offers algorithms for detecting and matching features in images, such as corners, keypoints, and descriptors. These features are fundamental for tasks like image registration, object localization, and image stitching.
- **Graphical User Interface (GUI) Support:** OpenCV includes functions for creating graphical user interfaces to interact with image and video data. This allows users to build custom applications for tasks like image annotation, parameter tuning, and real-time monitoring.

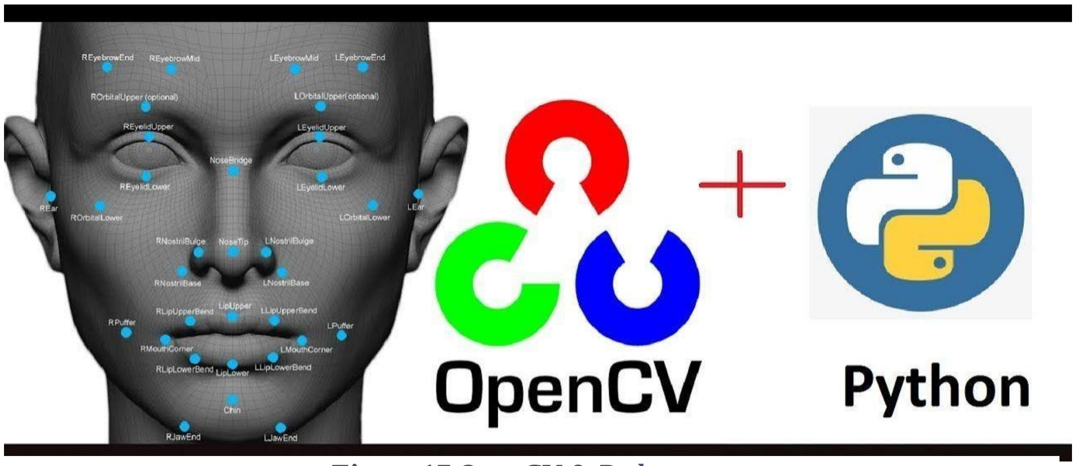


Figure 17 OpenCV & Python

- ◎ **Image and Video Processing**
- ◎ **Real-Time Hand Gesture Recognition**
- ◎ **Robust Feature Detection**
- ◎ **Pre-Trained Models**
- ◎ **Image Filtering and Enhancement**
- ◎ **Cross-Platform Support**
- ◎ **Large Community and Documentation**

- ◎ **Integration with Python**
- ◎ **Real-Time Capabilities**

In "Air Canvas," OpenCV serves as the backbone for capturing webcam input, processing images and video frames, and implementing hand gesture recognition. It plays a central role in translating users' hand and finger movements into digital art creation, making it a crucial component of the project's success.

- **Mediapipe:** A framework for building machine learning-based applications for various tasks, including hand tracking and pose estimation. It has gained widespread recognition and adoption due to its versatility and ease of use. Below are key aspects of MediaPipe's significance and capabilities. Its modular nature allows developers to mix and match components to create customized pipelines tailored to their specific needs. The library excels in providing robust and efficient solutions for diverse applications, from augmented reality experiences to gesture recognition systems. Leveraging both traditional computer vision techniques and deep learning models, MediaPipe strikes a balance between accuracy and real-time performance, making it an invaluable tool for projects seeking to incorporate vision-based functionalities seamlessly. With its active developer community and continuous updates, MediaPipe stands as a reliable resource for those aiming to bring advanced computer vision capabilities to their applications without the need for extensive expertise in the underlying algorithms:

Figure 18 Mediapipe

- ◎ Real-Time Processing
- ◎ Cross-Platform Compatibility
- ◎ Pre-Trained Models
- ◎ Customization
- ◎ Integration with TensorFlow
- ◎ Real-World Applications
- ◎ User-Friendly APIs

- **Numpy:**

NumPy is a fundamental library for numerical computing in Python. It provides support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays efficiently. Developed initially by Travis Oliphant in 2005, NumPy has become an integral part of the Python scientific computing ecosystem, serving as the foundation for many other libraries and tools.

Key Features of NumPy:

- **Multi-dimensional Arrays:** NumPy's core feature is its ndarray object, which represents multi-dimensional arrays of homogeneous data types. These arrays can have any number of dimensions and store elements of the same data type, allowing for efficient storage and manipulation of large datasets.
- **Efficient Operations:** NumPy provides a wide range of mathematical functions that operate element-wise on arrays, allowing for efficient computations without the need for explicit looping in Python. These functions are implemented in C and Fortran, ensuring high performance even with large datasets.

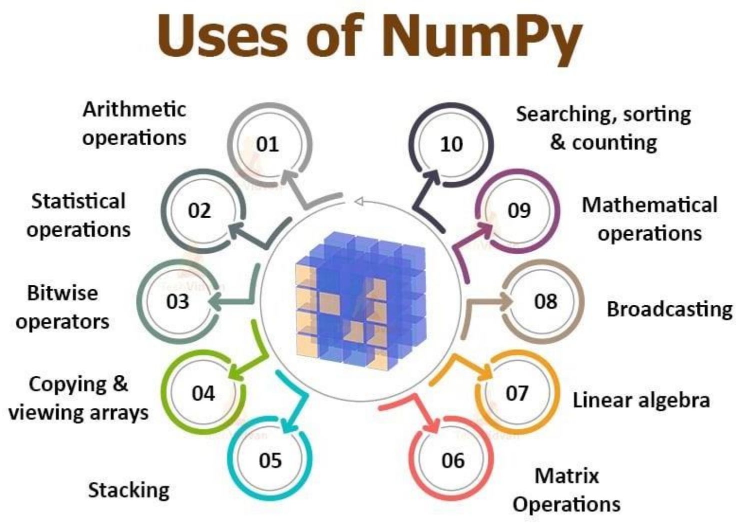


Figure 19 Numpy uses

- **Broadcasting:** NumPy supports broadcasting, a powerful mechanism that allows arrays of different shapes to be combined in arithmetic operations. This enables concise and efficient code for operations involving arrays of different sizes.
- **Indexing and Slicing:** NumPy offers flexible indexing and slicing capabilities for accessing elements or sub-arrays of an array. This allows for easy manipulation of data within arrays and is essential for array-oriented computing.

- **Linear Algebra Operations:** NumPy includes a comprehensive set of functions for linear algebra operations, such as matrix multiplication, matrix decomposition, eigenvalue calculations, and more. These functions provide efficient implementations of common linear algebra algorithms.
- **Random Number Generation:** NumPy features a powerful random number generation module (numpy.random) for generating arrays of random numbers with various distributions. This is useful for tasks such as simulations, statistical sampling, and random data generation.
- **Integration with Other Libraries:** NumPy integrates seamlessly with other Python libraries in the scientific computing ecosystem, such as SciPy, pandas, Matplotlib, and scikit-learn. This interoperability enables a wide range of data analysis, visualization, and machine learning tasks.

▪ **Collections:** In Python, the collections module is a powerhouse when it comes to data structures beyond the built-in types like lists, tuples, and dictionaries. One gem within this module is the deque (pronounced "deck"), short for double-ended queue. Deques are versatile data structures that allow efficient insertion and deletion operations from both ends, making them ideal for tasks like implementing queues and breadth-first search algorithms.

Here's a closer look at deques in Python's collections library:

◎ **What is a deque?**

A deque is a mutable, thread-safe, and iterable object that represents a double-ended queue. Unlike lists, deques provide fast appends and pops from both ends. This performance advantage makes them a suitable choice for scenarios where items are frequently added or removed from the beginning or end of a collection.

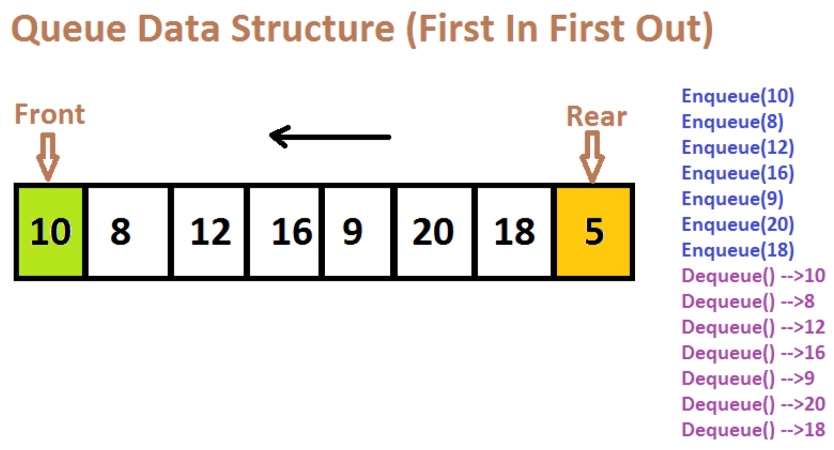


Figure 20 Dequeue applications

- **Key features of deques:**
  - **Efficient Operations:** Deques provide O(1) time complexity for append and pop operations at both ends, making them well-suited for scenarios requiring efficient insertion and deletion.
  - **Thread Safety:** Deques support safe multithreaded access. Multiple threads can manipulate a deque object concurrently without the need for external synchronization mechanisms.
  - **Iterability:** Deques can be iterated over using loops or comprehensions, just like lists and tuples.
  - **How to use deques:**
  - To start using deques in your Python code, you first need to import the deque class from the collections module

3.3.4 Languages used:

We have used only PYTHON followed by Machine Learning

- **Python:** It is a high-level, versatile programming language known for its simplicity and readability. Created by Guido van Rossum and first released in 1991, Python has gained widespread popularity in various domains, including web development, data science, artificial intelligence, and more. Its clean and elegant syntax, which emphasizes code readability, has made it a go-to language for both beginner and experienced programmers.

  Python offers a rich standard library that provides modules and packages for a wide range of tasks, from working with data structures to handling web requests. Its dynamic typing and automatic memory management simplify the development process, reducing the time and effort required to create robust and maintainable code.

Figure 21 Python

- **Machine Learning:** Machine learning is a subset of artificial intelligence (AI) that focuses on the development of algorithms and models capable of learning and making predictions or decisions based on data. It involves the use of statistical and computational techniques to enable computer systems to improve their performance on a specific task through experience.

Basically, it is of 3 types

- **Supervised Learning**
- **Unsupervised Learning**
- **Reinforcement Learning**

3.3.5 Libraries Used:

- **NumPy**: NumPy is a fundamental Python library for numerical and scientific computing. It provides support for multi-dimensional arrays, mathematical functions, and operations, making it a crucial tool for data manipulation, linear algebra, and scientific computing in Python. NumPy is a cornerstone in the Python data science ecosystem and is extensively used in fields such as machine learning, data analysis, and scientific research.
- **Key features of NumPy include:**
  - **Arrays:** NumPy's core feature is the ndarray (n-dimensional array), which is a fast and flexible container for large datasets in Python. Arrays in NumPy can have multiple dimensions and support a variety of data types.
  - **Vectorized Operations:** NumPy enables vectorized operations, allowing users to perform element-wise operations on entire arrays without the need for explicit loops. This vectorization enhances the efficiency and readability of numerical computations.
  - **Mathematical Functions:** NumPy provides a rich set of mathematical functions for performing operations like trigonometry, linear algebra, statistical analysis, and more. These functions are optimized for array computations and operate seamlessly on entire arrays.
- **MediaPipe:** A framework for building machine learning-based applications for various tasks, including hand tracking and pose estimation. It has gained widespread recognition and adoption due to its versatility and ease of use. Below are key aspects of MediaPipe's significance and capabilities. Its modular nature allows developers to mix and match components to create customized pipelines tailored to their specific needs. The library excels in providing robust and efficient solutions for diverse applications, from augmented reality

experiences to gesture recognition systems. With its active developer community and continuous updates, MediaPipe stands as a reliable resource for those aiming to bring advanced computer vision capabilities to their applications without the need for extensive expertise in the underlying algorithms:

- Real-Time Processing
- Cross-Platform Compatibility
- Pre-Trained Models
- Customization

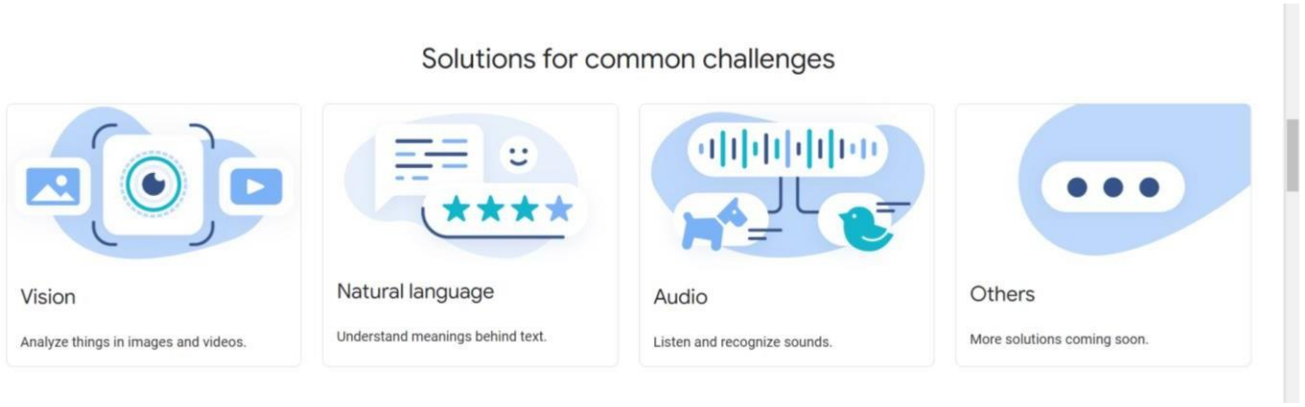

Solutions for common challenges
Vision
Analyze things in images and videos.
Natural language
Understand meanings behind text.
Audio
Listen and recognize sounds.
Others
More solutions coming soon.

Figure 22 Solutions by Mediapipe

- **Computer Vision(CV2):** OpenCV (Open Source Computer Vision Library), often abbreviated as CV2, is an open-source computer vision and machine learning software library. It provides a comprehensive set of tools and functions for image and video processing, computer vision tasks, and machine learning applications. OpenCV is written in C++ and has bindings for various programming languages, including Python, Java, and more.

Figure 23 Computer Vision

Key features of the OpenCV library include:

- **Image Processing:** OpenCV offers a wide range of functions for image manipulation, such as resizing, cropping, rotating, filtering, and thresholding.
- **Camera Calibration:** OpenCV provides tools for calibrating cameras, which is crucial for tasks like 3D reconstruction and augmented reality. Camera calibration helps correct distortions and obtain accurate measurements from images.
- **Image and Video I/O**: OpenCV supports a wide range of image and video formats, making it easy to read and write images and videos. This functionality is essential for handling input and output in computer vision applications.
- **User Interface:** OpenCV includes a simple GUI (Graphical User Interface) module that enables developers to create basic graphical interfaces for their applications.

In summary, OpenCV (CV2) is a powerful and versatile library for computer vision and image processing. Its broad range of functionalities, along with its open-source nature and active community, makes it a popular choice for researchers, engineers, and developers working on a variety of projects in the field of computer vision and machine learning.

- **Collections:** The collections library in Python, which includes the deque (double-ended queue) data structure, can be a valuable tool for certain tasks. While the primary purpose of collections is to provide alternatives to built-in data types, the deque specifically can be beneficial in scenarios where efficient and fast operations are required at both ends of a sequence.

Here are a few ways in which the deque from the collections library can be useful in machine learning:

- **Sliding Windows and Sequence Processing:** Machine learning tasks often involve processing sequences of data, such as time series or natural language. The deque data structure is well-suited for implementing sliding windows efficiently.
- **Queue-Like Behavior:** In certain machine learning algorithms, such as breadth-first search or certain optimization algorithms, a queue-like data structure is needed.
- **Efficient Stack Operations:** While deque primarily supports queue operations, it can also be used as a stack. Since it allows appending and popping elements from both ends with O(1) complexity, it can serve as an efficient stack for certain algorithms and data processing tasks.

- **Memory Efficiency:** The deque is implemented as a doubly-linked list, which allows for efficient memory usage compared to a regular list when dealing with a large number of elements.

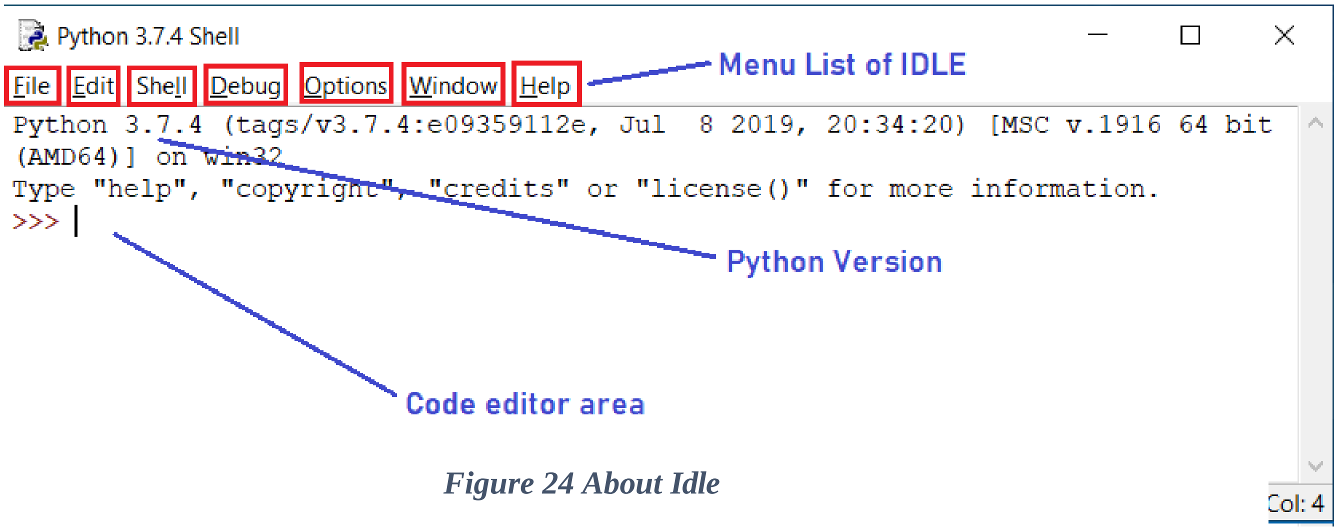
In summary, the collections library, particularly the deque data structure, can be a handy tool in machine learning for tasks involving efficient manipulation of sequences, such as sliding windows, queue-like behavior, and stack operations. Its performance characteristics make it a suitable choice for scenarios where fast operations at both ends of a sequence are essential.

3.3.5 Platform Used:

Python IDLE:

Using Python IDLE (Integrated Development and Learning Environment) as a compiling platform for the AIR canvas project, which integrates machine learning, Mediapipe, and OpenCV, offers several advantages in terms of simplicity, flexibility, and ease of use. Here's how Python IDLE can be utilized effectively:

- **Simplicity and Accessibility:** Python IDLE provides a straightforward and user-friendly interface for writing, testing, and debugging Python code, making it accessible to developers of all skill levels. Python IDLE seamlessly integrates with a vast ecosystem of Python libraries, including machine learning frameworks like TensorFlow and scikit-learn, computer vision libraries like OpenCV, and media processing libraries like Mediapipe. Developers can easily import and leverage these libraries within their Python scripts to access advanced functionalities and streamline development workflows.



```
Python 3.7.4 (tags/v3.7.4:e09359112e, Jul  8 2019, 20:34:20) [MSC v.1916 64 bit
(AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> 
```

Figure 24 About Idle

- **Debugging and Profiling Tools:** Python IDLE offers built-in debugging and profiling tools that help identify and resolve errors, optimize performance, and troubleshoot issues in the code. Developers can set breakpoints, inspect variables, and step through code execution to diagnose problems and improve the efficiency of their algorithms.

  This granular level of control allows for precise pinpointing of bugs and issues, facilitating rapid troubleshooting and debugging. Additionally, Python IDLE's profiling tools offer insights into code performance, enabling developers to identify bottlenecks and optimize critical sections of their algorithms for enhanced efficiency. By leveraging these debugging and profiling capabilities, developers can streamline the development process, accelerate time-to-market, and deliver high-quality Python applications that meet performance and reliability expectations.
- **Script Execution and Visualization:** Python IDLE supports the execution of Python scripts directly within its environment, allowing developers to run and test their AIR canvas application code seamlessly. Within the Python IDLE interface, developers can execute scripts with a simple keystroke, instantly observing the behavior and output of their code. This rapid feedback loop accelerates the development process, enabling developers to quickly prototype ideas, experiment with different algorithms, and validate functionality. Furthermore, Python IDLE facilitates visualization by supporting the display of graphical output directly within its interface. Developers can leverage libraries like Matplotlib or Pygame to generate visual representations of their AIR canvas application, such as hand tracking trajectories or rendered drawings. This real-time visualization enhances the development experience, allowing developers to gain immediate insights into the behavior and performance of their application. By integrating script execution and visualization capabilities, Python IDLE empowers developers to iterate rapidly, refine their code efficiently, and ultimately deliver a polished and feature-rich AIR canvas application.
- **Script Editing and Version Control:** Python IDLE provides comprehensive script editing capabilities, including syntax highlighting, code completion, and automatic indentation, enhancing code readability and maintainability.

Overall, Python IDLE serves as a versatile and efficient compiling platform for the AIR canvas project, offering a powerful yet accessible environment for developing, testing, and deploying machine learning and computer vision applications with ease. Its simplicity, compatibility, and robust feature set make it an ideal choice for building innovative digital art creation tools that leverage the capabilities of machine learning, Mediapipe, and OpenCV.

3.3.6 Algorithm

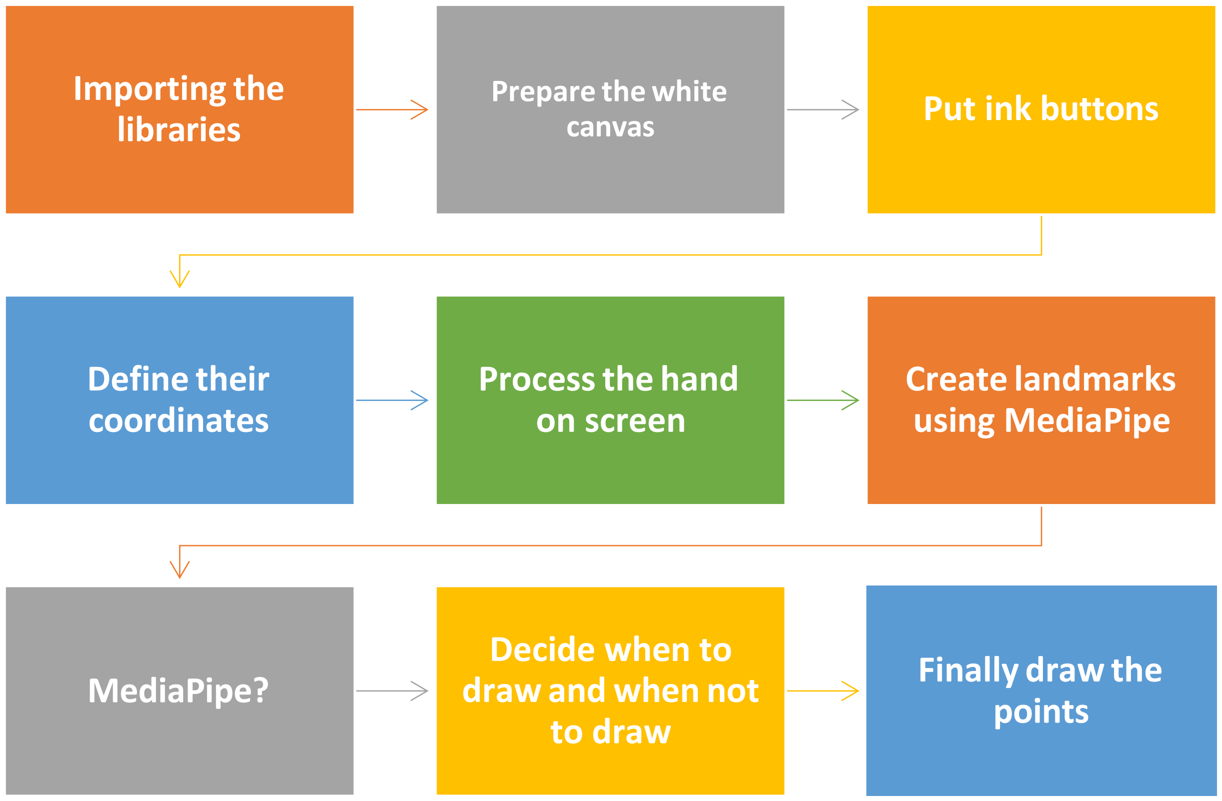


Figure 25 Algorithm

- **Import Requirements:**
  - Check for the availability of necessary hardware components, such as a computer with a camera or a mobile phone.
  - Ensure that the required libraries, including OpenCV, NumPy, Mediapipe, and deque, are installed and accessible in the Python environment.
- **Construct Frames:**
  - Initialize the video and canvas frames, setting their resolutions and sizes to occupy half of the screen each.
  - Adjust the window sizes dynamically based on user preferences and screen dimensions.

- **Hand Tracking Model:**
  - Utilize Mediapipe's hand tracking model to detect and track hand movements in real-time video input.
  - Configure parameters such as max_num_hands and min_detection_confidence to optimize hand detection accuracy and performance.

- **Locate Hand Movements:**
  - Continuously update the positions of hand landmarks detected by the hand tracking model and store them in a data structure, such as a deque.
  - Implement logic to clear the deque when the user initiates an erasing action to reset the drawing canvas.
- **Initialize Color Palette:**
  - Define arrays or dictionaries to represent different colors available for drawing on the canvas.
  - Create graphical user interface (GUI) elements, such as rectangles or buttons, to display the color palette to the user.
- **Decide Conditions:**
  - Evaluate hand gestures and movements to determine the user's intended actions, such as drawing or erasing.
  - Use thresholds or predefined criteria, such as the distance between thumb and forefinger, to classify hand gestures into different modes (e.g., drawing mode or empty mode).

1. distance**(thumb-forefinger) < 40 → Empty mode**

2. distance**(thumb-forefinger) > 40 → Draw mode**

- **Draw and Display on Canvas:**
  - Simulate drawing actions by recording the coordinates of hand movements in the deque and updating the drawing canvas accordingly.

- Map the recorded coordinates to specific colors from the color palette array to visualize the drawing process in real-time.

- **Finalize Actions:**
  - Track user interactions and implement corresponding actions, such as clearing the canvas, changing drawing colors, or toggling between drawing modes.
  - Implement error-handling mechanisms to ensure robustness and responsiveness, such as detecting and handling invalid or unintended user inputs.

By following this detailed algorithm, the AIR canvas application can effectively leverage machine learning, Mediapipe, and computer vision technologies to provide users with a seamless and intuitive digital art creation experience.

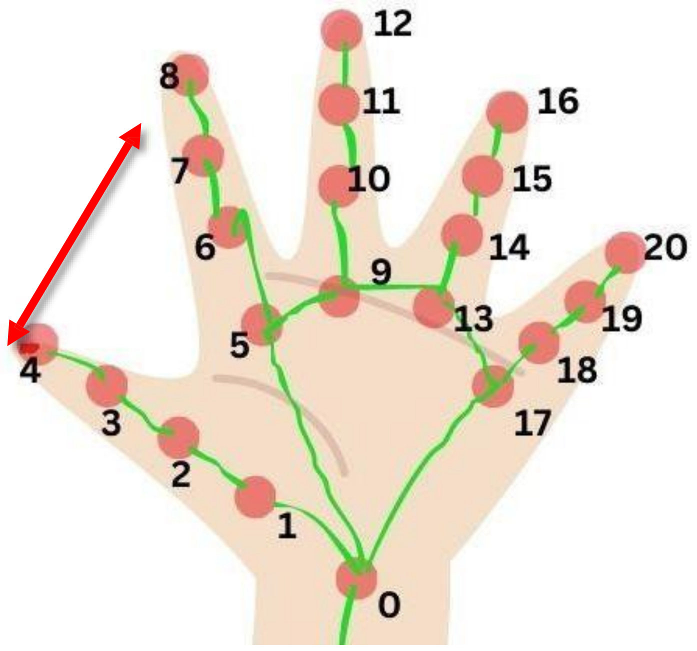
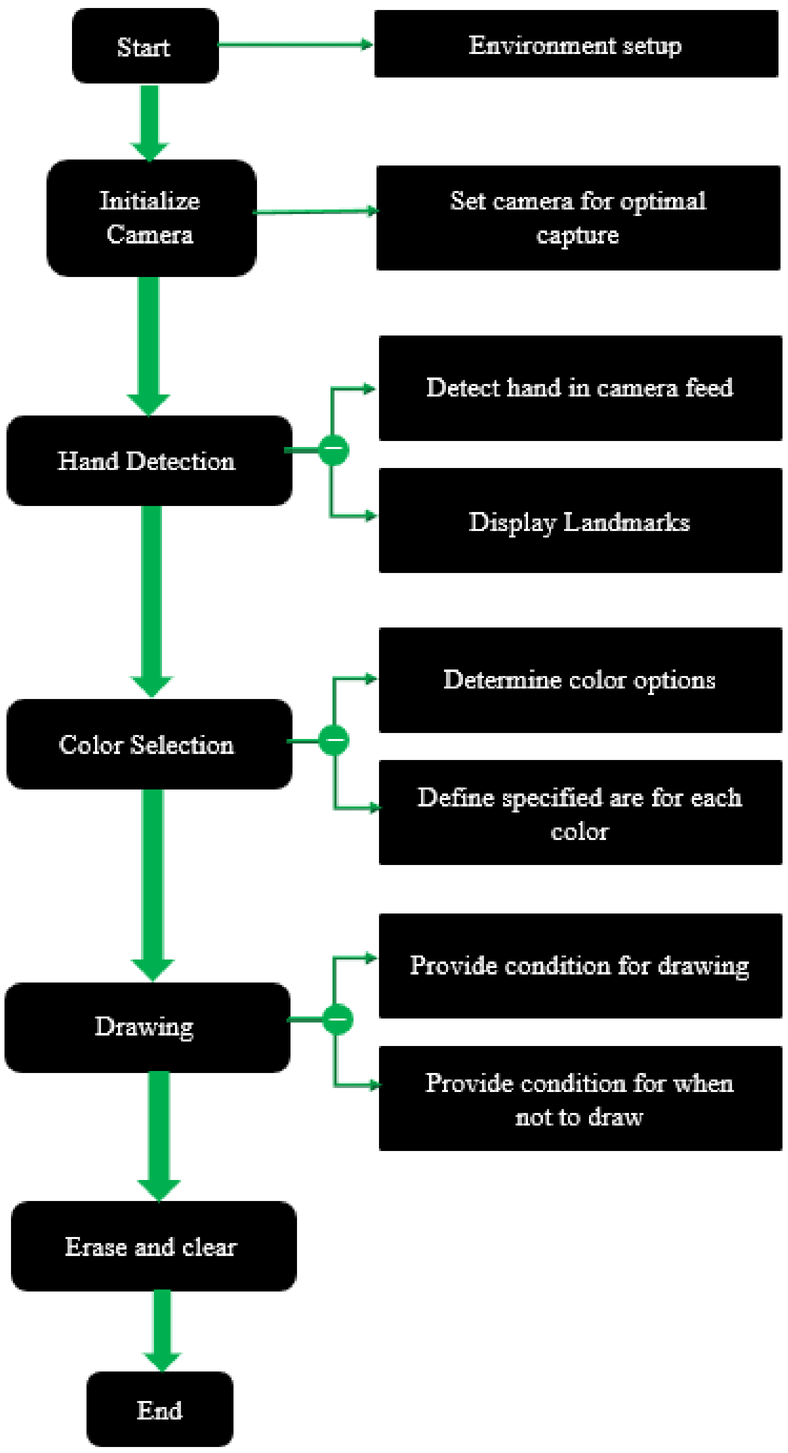


Figure 26 Thumb and Forefinger Landmark Distance

Detailed Overview of Algorithm.!



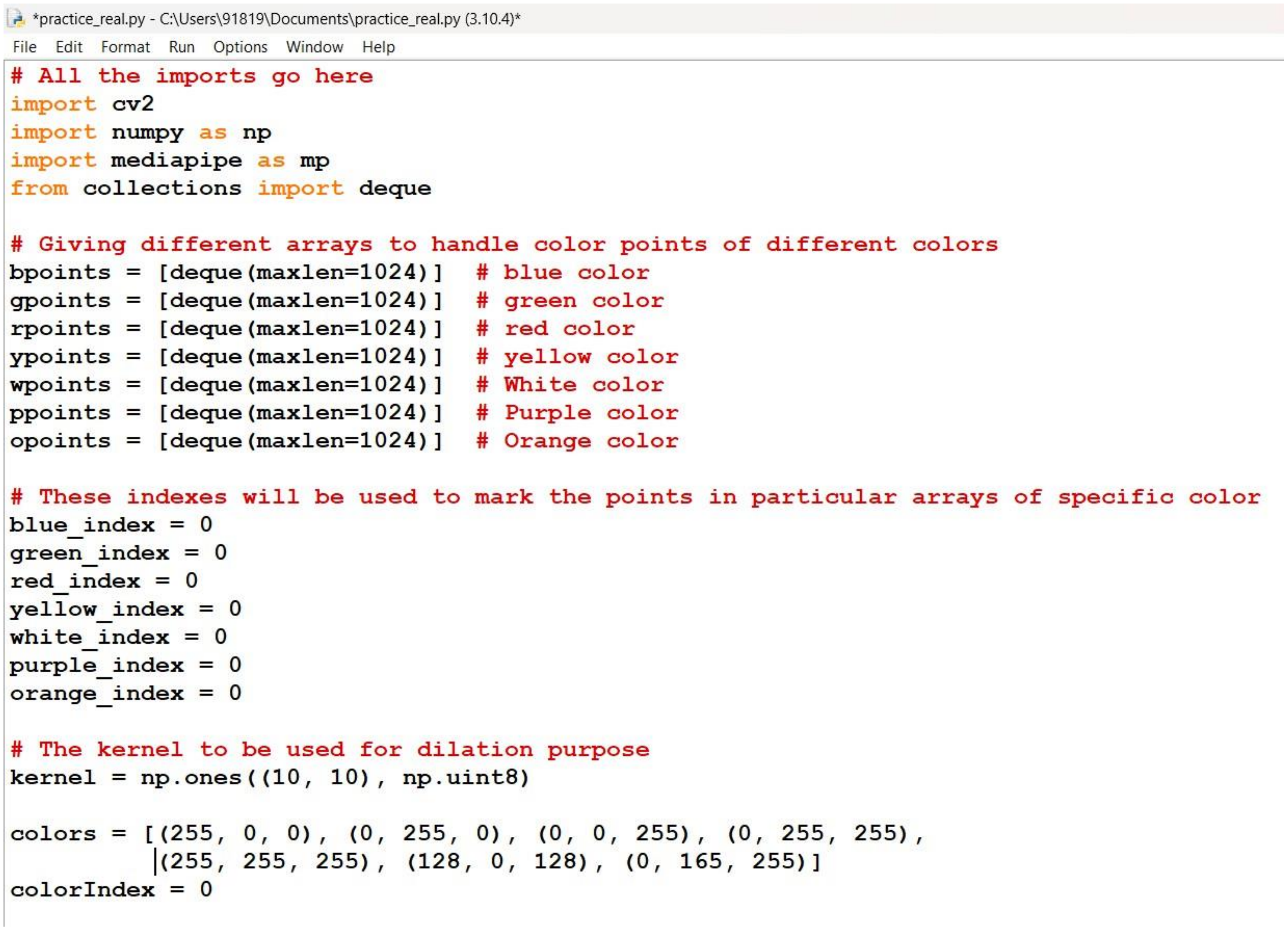
Start
Environment setup
Initialize Camera
Set camera for optimal capture
Detect hand in camera feed
Hand Detection
Display Landmarks
Determine color options
Color Selection
Define specified are for each color
Provide condition for drawing
Drawing
Provide condition for when not to draw
Erase and clear
End

Figure 27 Algorithm Flowchart

CHAPTER- 4

Result analysis and validation

4.1 Implementation of design:


practice_real.py - C:\Users\91819\Documents\practice_real.py (3.10.4)
File Edit Format Run Options Window Help

```
# All the imports go here
import cv2
import numpy as np
import mediapipe as mp
from collections import deque

# Giving different arrays to handle color points of different colors
bpoints = [deque(maxlen=1024)]  # blue color
gpoints = [deque(maxlen=1024)]  # green color
rpoints = [deque(maxlen=1024)]  # red color
ypoints = [deque(maxlen=1024)]  # yellow color
wpoints = [deque(maxlen=1024)]  # White color
ppoints = [deque(maxlen=1024)]  # Purple color
opoints = [deque(maxlen=1024)]  # Orange color

# These indexes will be used to mark the points in particular arrays of specific color
blue_index = 0
green_index = 0
red_index = 0
yellow_index = 0
white_index = 0
purple_index = 0
orange_index = 0

# The kernel to be used for dilation purpose
kernel = np.ones((10, 10), np.uint8)

colors = [(255, 0, 0), (0, 255, 0), (0, 0, 255), (0, 255, 255),
          (255, 255, 255), (128, 0, 128), (0, 165, 255)]
colorIndex = 0
```

1. **Imports:** The code imports necessary libraries including OpenCV (cv2), NumPy (np), Mediapipe (mp), and deque from the collections module.
2. **Color Points Arrays:** Arrays (bpoints, gpoints, rpoints, etc.) are initialized to store coordinates of points drawn in different colors. Each array is a deque with a maximum length of 1024, ensuring that only the most recent 1024 points are retained.
3. **Color Indexes:** Index variables (blue_index, green_index, etc.) are initialized to keep track of the current position in each color's points array.

4. **Dilation Kernel:** A kernel (kernel) is defined for dilation purposes. Dilation is a morphological operation used to expand or grow the boundaries of shapes in an image.
5. **Color Definitions:** A list of colors (colors) is defined, with each color represented as an RGB tuple. The colorIndex variable is initialized to keep track of the current color being used for drawing.
6. **Explanation in Points:** Arrays are used to store coordinates of points drawn in different colors, allowing for the creation of multi-colored drawings.
    - Deques with a maximum length of 1024 ensure efficient memory usage by retaining only the most recent points.
    - Color indexes are used to keep track of the current position in each color's points array, facilitating the addition of new points.
    - A dilation kernel is defined for morphological operations, such as expanding or thickening drawn lines.
    - The list of colors and colorIndex variable enable cycling through different colors for drawing, providing flexibility and customization options for users.

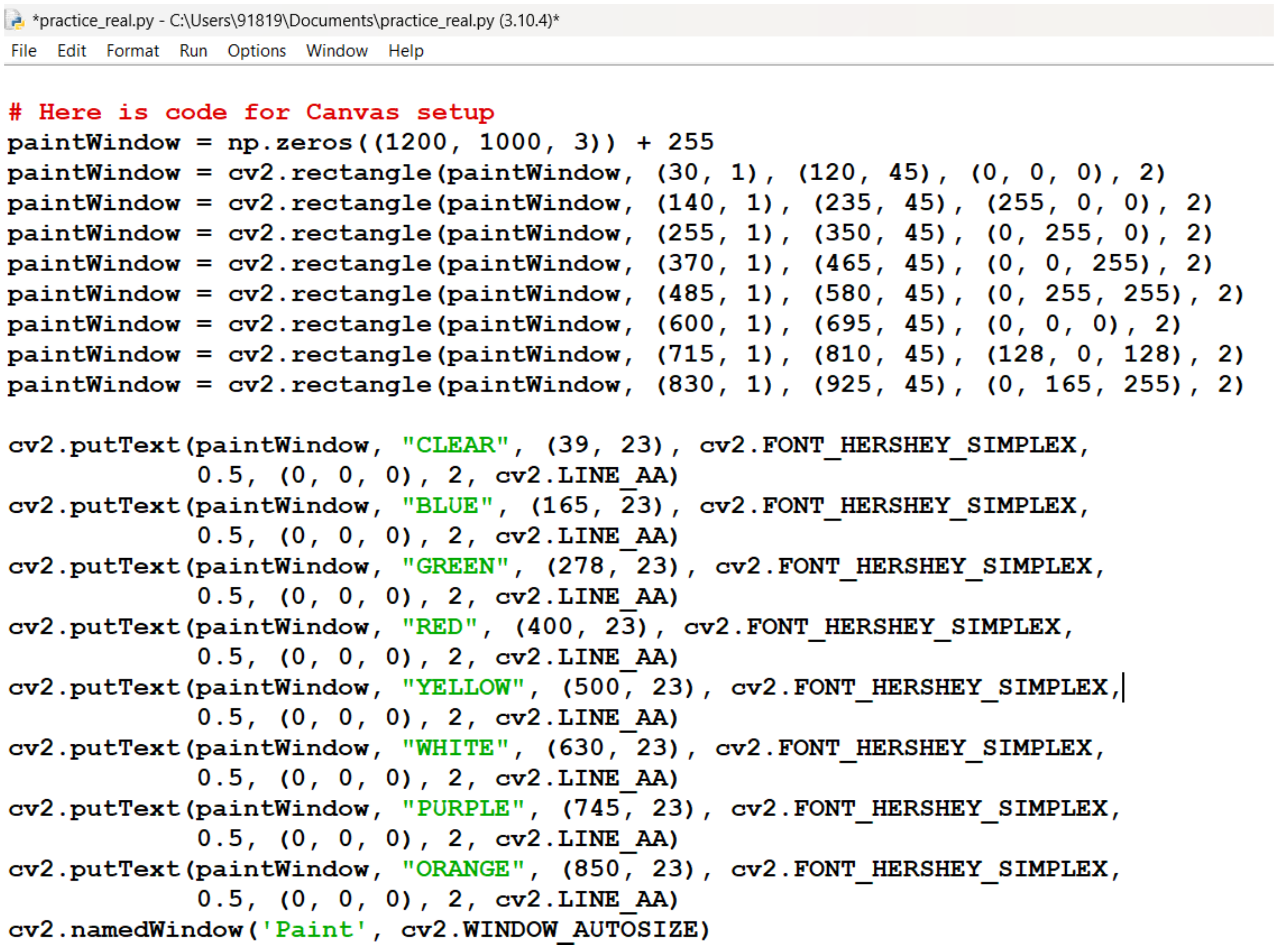
1. **Canvas Initialization:**

- The code initializes a blank canvas (paintWindow) with dimensions 1200x1000 pixels and fills it with a white background (RGB value of 255).
- Multiple rectangles are drawn on the canvas to represent the color palette, each with a specified color and outline.
- Text labels are added to each rectangle to indicate the corresponding color option (e.g., "BLUE", "GREEN", "RED", etc.).
- The canvas window is named "Paint" and configured to auto-size based on its content.

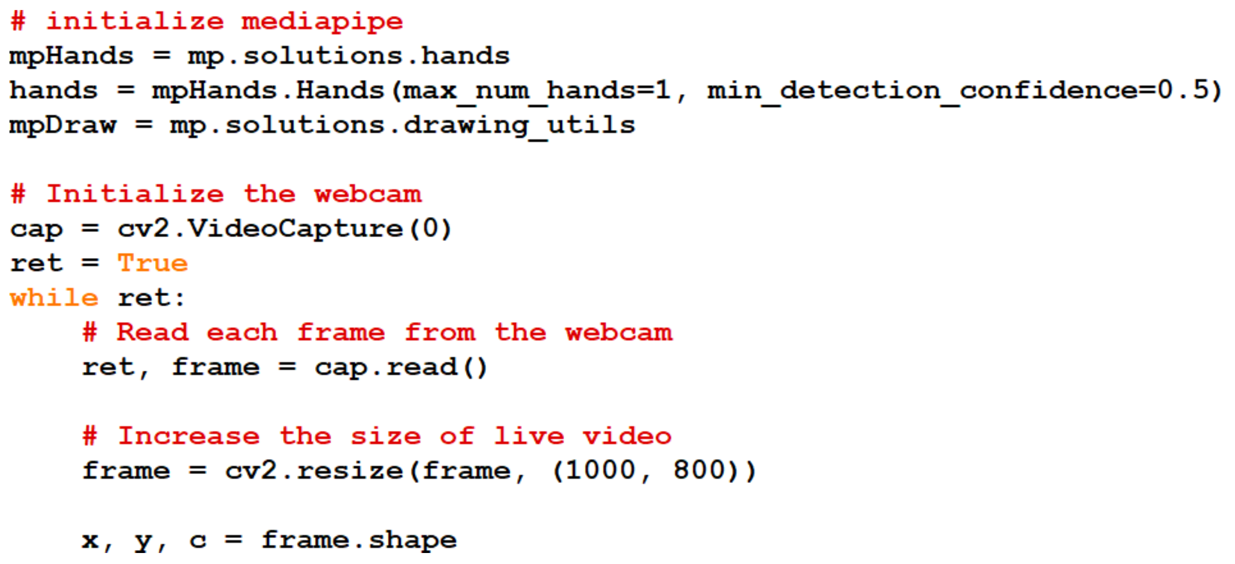
2. **User Interface Setup:**

- Rectangles are used to visually represent color options, allowing users to select a desired color for drawing.
- Text labels provide clear identification of each color option, enhancing the usability and intuitiveness of the canvas interface.
- By organizing color options in a visually appealing layout, the canvas setup encourages user engagement and facilitates seamless interaction with the drawing tool.

WINDOW 1 (White Canvas Implementation with all colors boxes)


practice_real.py - C:\Users\91819\Documents\practice_real.py (3.10.4)
File Edit Format Run Options Window Help

```
# Here is code for Canvas setup
paintWindow = np.zeros((1200, 1000, 3)) + 255
paintWindow = cv2.rectangle(paintWindow, (30, 1), (120, 45), (0, 0, 0), 2)
paintWindow = cv2.rectangle(paintWindow, (140, 1), (235, 45), (255, 0, 0), 2)
paintWindow = cv2.rectangle(paintWindow, (255, 1), (350, 45), (0, 255, 0), 2)
paintWindow = cv2.rectangle(paintWindow, (370, 1), (465, 45), (0, 0, 255), 2)
paintWindow = cv2.rectangle(paintWindow, (485, 1), (580, 45), (0, 255, 255), 2)
paintWindow = cv2.rectangle(paintWindow, (600, 1), (695, 45), (0, 0, 0), 2)
paintWindow = cv2.rectangle(paintWindow, (715, 1), (810, 45), (128, 0, 128), 2)
paintWindow = cv2.rectangle(paintWindow, (830, 1), (925, 45), (0, 165, 255), 2)

cv2.putText(paintWindow, "CLEAR", (39, 23), cv2.FONT_HERSHEY_SIMPLEX,
            0.5, (0, 0, 0), 2, cv2.LINE_AA)
cv2.putText(paintWindow, "BLUE", (165, 23), cv2.FONT_HERSHEY_SIMPLEX,
            0.5, (0, 0, 0), 2, cv2.LINE_AA)
cv2.putText(paintWindow, "GREEN", (278, 23), cv2.FONT_HERSHEY_SIMPLEX,
            0.5, (0, 0, 0), 2, cv2.LINE_AA)
cv2.putText(paintWindow, "RED", (400, 23), cv2.FONT_HERSHEY_SIMPLEX,
            0.5, (0, 0, 0), 2, cv2.LINE_AA)
cv2.putText(paintWindow, "YELLOW", (500, 23), cv2.FONT_HERSHEY_SIMPLEX,
            0.5, (0, 0, 0), 2, cv2.LINE_AA)
cv2.putText(paintWindow, "WHITE", (630, 23), cv2.FONT_HERSHEY_SIMPLEX,
            0.5, (0, 0, 0), 2, cv2.LINE_AA)
cv2.putText(paintWindow, "PURPLE", (745, 23), cv2.FONT_HERSHEY_SIMPLEX,
            0.5, (0, 0, 0), 2, cv2.LINE_AA)
cv2.putText(paintWindow, "ORANGE", (850, 23), cv2.FONT_HERSHEY_SIMPLEX,
            0.5, (0, 0, 0), 2, cv2.LINE_AA)
cv2.namedWindow('Paint', cv2.WINDOW_AUTOSIZE)
```



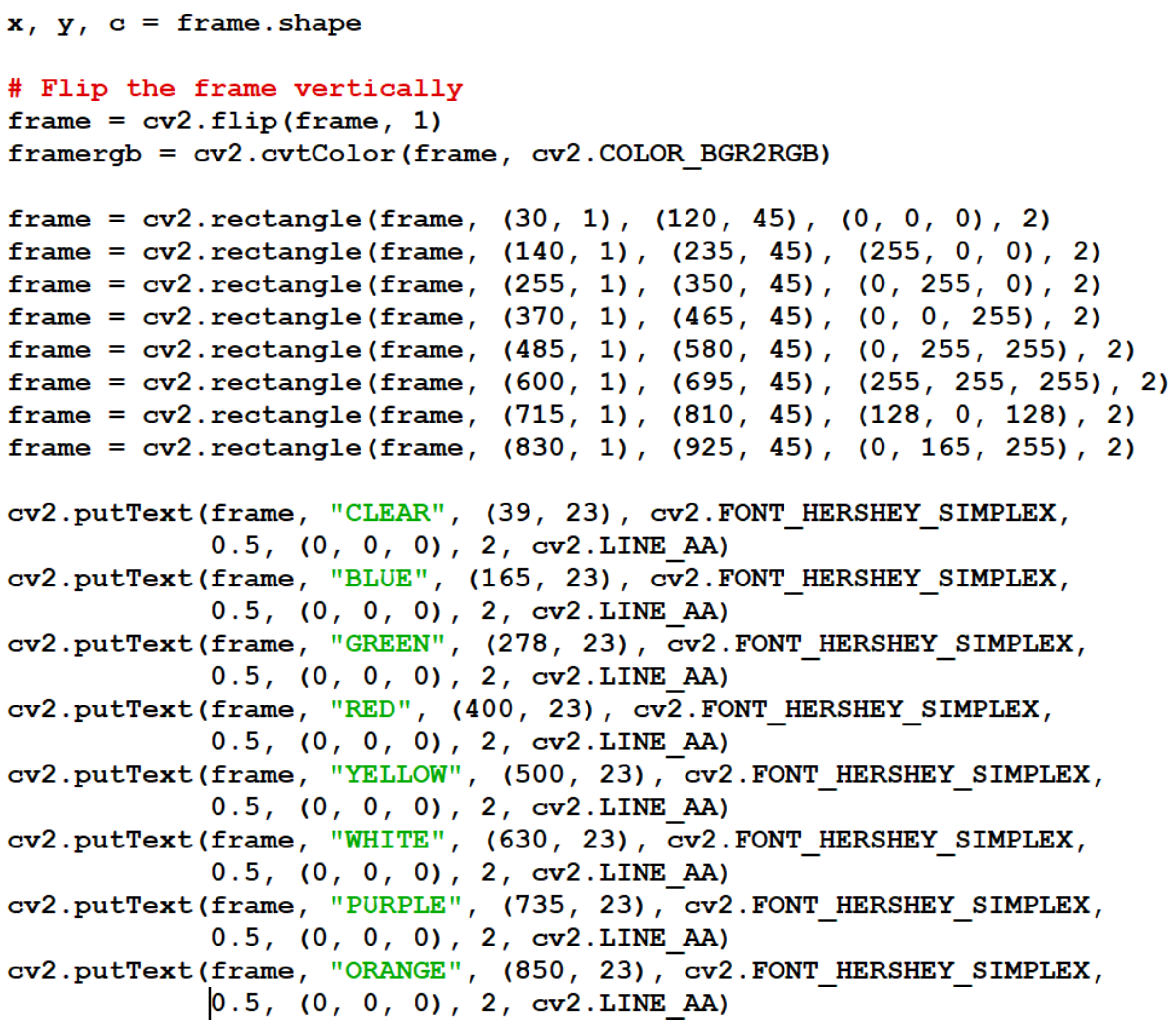
```
# initialize mediapipe
mpHands = mp.solutions.hands
hands = mpHands.Hands(max_num_hands=1, min_detection_confidence=0.5)
mpDraw = mp.solutions.drawing_utils

# Initialize the webcam
cap = cv2.VideoCapture(0)
ret = True
while ret:
    # Read each frame from the webcam
    ret, frame = cap.read()

    # Increase the size of live video
    frame = cv2.resize(frame, (1000, 800))

    x, y, c = frame.shape
```

This code initializes the hand tracking functionality using the Mediapipe framework by creating a hands object with specific parameters. It also sets up the webcam for video input and resizes the live video feed to dimensions (1000, 800) for display. These preparations enable real-time hand tracking and processing of video input from the webcam.

Hands Detection Setup: hands object is created using mpHands.Hands() constructor with specified parameters such as max_num_hands and min_detection_confidence.

max_num_hands=1 ensures that only one hand is detected at a time, simplifying the implementation for this project. min_detection_confidence=0.5 sets the minimum confidence threshold for hand detection, ensuring reliable detection results.

Window 2: Code for video frame (realtime) explanation is same as of paint window.



```
x, y, c = frame.shape

# Flip the frame vertically
frame = cv2.flip(frame, 1)
framergb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)

frame = cv2.rectangle(frame, (30, 1), (120, 45), (0, 0, 0), 2)
frame = cv2.rectangle(frame, (140, 1), (235, 45), (255, 0, 0), 2)
frame = cv2.rectangle(frame, (255, 1), (350, 45), (0, 255, 0), 2)
frame = cv2.rectangle(frame, (370, 1), (465, 45), (0, 0, 255), 2)
frame = cv2.rectangle(frame, (485, 1), (580, 45), (0, 255, 255), 2)
frame = cv2.rectangle(frame, (600, 1), (695, 45), (255, 255, 255), 2)
frame = cv2.rectangle(frame, (715, 1), (810, 45), (128, 0, 128), 2)
frame = cv2.rectangle(frame, (830, 1), (925, 45), (0, 165, 255), 2)

cv2.putText(frame, "CLEAR", (39, 23), cv2.FONT_HERSHEY_SIMPLEX,
            0.5, (0, 0, 0), 2, cv2.LINE_AA)
cv2.putText(frame, "BLUE", (165, 23), cv2.FONT_HERSHEY_SIMPLEX,
            0.5, (0, 0, 0), 2, cv2.LINE_AA)
cv2.putText(frame, "GREEN", (278, 23), cv2.FONT_HERSHEY_SIMPLEX,
            0.5, (0, 0, 0), 2, cv2.LINE_AA)
cv2.putText(frame, "RED", (400, 23), cv2.FONT_HERSHEY_SIMPLEX,
            0.5, (0, 0, 0), 2, cv2.LINE_AA)
cv2.putText(frame, "YELLOW", (500, 23), cv2.FONT_HERSHEY_SIMPLEX,
            0.5, (0, 0, 0), 2, cv2.LINE_AA)
cv2.putText(frame, "WHITE", (630, 23), cv2.FONT_HERSHEY_SIMPLEX,
            0.5, (0, 0, 0), 2, cv2.LINE_AA)
cv2.putText(frame, "PURPLE", (735, 23), cv2.FONT_HERSHEY_SIMPLEX,
            0.5, (0, 0, 0), 2, cv2.LINE_AA)
cv2.putText(frame, "ORANGE", (850, 23), cv2.FONT_HERSHEY_SIMPLEX,
            0.5, (0, 0, 0), 2, cv2.LINE_AA)
```

Starting with Landmark Prediction on the frame of 1000*800 (adjustable) and then code for drawing landmarks on frame window for real drawing feel

```
# Get hand landmark prediction
result = hands.process(framergb)

# post-process the result
if result.multi_hand_landmarks:
    landmarks = []
    for handslms in result.multi_hand_landmarks:
        for lm in handslms.landmark:
            lmx = int(lm.x * 1000)
            lmy = int(lm.y * 800)
            landmarks.append([lmx, lmy])

        # Drawing landmarks on frames
        mpDraw.draw_landmarks(frame, handslms, mpHands.HAND_CONNECTIONS)
    fore_finger = (landmarks[8][0], landmarks[8][1])
    center = fore_finger
    thumb = (landmarks[4][0], landmarks[4][1])
    cv2.circle(frame, center, 3, (0, 255, 0), -1)

    if (thumb[1] - center[1] < 40):
        bpoints.append(deque(maxlen=512))
        blue_index += 1
        gpoints.append(deque(maxlen=512))
        green_index += 1
        rpoints.append(deque(maxlen=512))
        red_index += 1
        ypoints.append(deque(maxlen=512))
        yellow_index += 1
        wpoints.append(deque(maxlen=512))
        white_index += 1
        ppoints.append(deque(maxlen=512))
        purple_index += 1
        opoints.append(deque(maxlen=512))
        orange_index += 1
```

```
elif center[1] <= 55:
    if 40 <= center[0] <= 140:  # Clear Button
        bpoints = [deque(maxlen=512)]
        gpoints = [deque(maxlen=512)]
        rpoints = [deque(maxlen=512)]
        ypoints = [deque(maxlen=512)]
        wpoints = [deque(maxlen=512)]
        ppoints = [deque(maxlen=512)]
        opoints = [deque(maxlen=512)]

        blue_index = 0
        green_index = 0
        red_index = 0
        yellow_index = 0
        white_index = 0
        purple_index = 0
        orange_index = 0

        paintWindow[67:, :, :] = 255
    elif 160 <= center[0] <= 255:
        colorIndex = 0  # Blue
    elif 275 <= center[0] <= 370:
        colorIndex = 1  # Green
    elif 390 <= center[0] <= 485:
        colorIndex = 2  # Red
    elif 505 <= center[0] <= 600:
        colorIndex = 3  # Yellow
    elif 620 <= center[0] <= 715:
        colorIndex = 4  # White
    elif 735 <= center[0] <= 830:
        colorIndex = 5  # Purple
    elif 850 <= center[0] <= 945:
        colorIndex = 6  # Orange
    else:
        if colorIndex == 0:
            bpoints[blue_index].appendleft(center)
        elif colorIndex == 1:
            gpoints[green_index].appendleft(center)
        elif colorIndex == 2:
            rpoints[red_index].appendleft(center)
        elif colorIndex == 3:
            ypoints[yellow_index].appendleft(center)
        elif colorIndex == 4:
            wpoints[white_index].appendleft(center)
        elif colorIndex == 5:
            ppoints[purple_index].appendleft(center)
        elif colorIndex == 6:
            opoints[orange_index].appendleft(center)

# Append the next deques when nothing is detected to avoid messing up
else:
    bpoints.append(deque(maxlen=512))
    blue_index += 1
    gpoints.append(deque(maxlen=512))
    green_index += 1
    rpoints.append(deque(maxlen=512))
    red_index += 1
    ypoints.append(deque(maxlen=512))
    yellow_index += 1
    wpoints.append(deque(maxlen=512))
    white_index += 1
    ppoints.append(deque(maxlen=512))
    purple_index += 1
    opoints.append(deque(maxlen=512))
    orange_index += 1
```

- If the distance between the thumb and the forefinger's y-coordinates is less than 40 pixels, indicating the gesture for color selection, new deques are appended for each color, and respective indices are updated.
- If the y-coordinate of the hand is less than or equal to 55 pixels, it checks for button presses: clearing the canvas or selecting a color based on the x-coordinate.
- Otherwise, it appends points to the deque corresponding to the selected color index, allowing for drawing on the canvas. If no hand is detected, new deques are appended to avoid interruptions in drawing.

1. **Checking Hand Distance**:
    - The code checks the distance between the thumb and the center of the hand to determine the user's action.
    - If the distance is less than 40 pixels, it indicates that the thumb and forefinger are close, triggering a mode change or action initiation.

2. **Mode Change Handling**:
    - If the hand is close to the bottom of the frame, it checks the horizontal position to determine the user's intended action.
    - If the hand is over the "Clear" button area (40-140 pixels), it clears all drawing points and resets color indexes.
    - Otherwise, it maps the horizontal position to specific color options, updating the colorIndex accordingly.

3. **Drawing Handling:**
    - If the hand is not close to the bottom of the frame, it checks the current color index to determine the color of the drawing.
    - Based on the color index, the coordinates of the hand's center are appended to the corresponding color's points deque for drawing.

4. **Deque Management:**
    - Deques are used to store drawing points for different colors, ensuring a limited history of drawing actions.
    - When a new drawing session starts or no hand is detected, new deques are appended to prevent mixing drawing sessions.

5. **Index Increment:**
    - Index variables (blue_index, green_index, etc.) are incremented to keep track of the current position in each color's points deque. This ensures that new drawing points are added to the correct deque.

```
    # Draw lines of all the colors on the canvas and frame
    points = [bpoints, gpoints, rpoints, ypoints, wpoints, ppoints, opoints]
    for i in range(len(points)):
        for j in range(len(points[i])):
            for k in range(1, len(points[i][j])):
                if points[i][j][k - 1] is None or points[i][j][k] is None:
                    continue

                cv2.line(frame, points[i][j][k - 1],
                         points[i][j][k], colors[i], 2)

                cv2.line(paintWindow, points[i][j][k - 1],
                         points[i][j][k], colors[i], 2)

    cv2.imshow("Output", frame)
    cv2.imshow("Paint", paintWindow)

    if cv2.waitKey(1) == ord('q'):
        break

# release the webcam and destroy all active windows
cap.release()
cv2.destroyAllWindows()
```

1. **Drawing Lines:**
   - The code iterates through all the color points arrays (bpoints, gpoints, etc.) to draw lines on both the frame and the canvas window.
   - For each color, it iterates through all the points in the deque and draws lines between consecutive points using the cv2.line() function.
   - Lines are drawn with a thickness of 2 pixels and with colors corresponding to the respective color arrays (colors).

2. **Displaying Output:**
   - After drawing lines on both the frame and the canvas window, the code displays the updated frame and canvas using cv2.imshow() function.
   - The frame displays the live video feed with the drawn lines, while the canvas window shows the drawing in progress.

3. **User Interaction:**
    - The code waits for a key press using cv2.waitKey(1).
    - If the 'q' key is pressed, the loop breaks, and the program exits, releasing the webcam and closing all active windows using cap.release() and cv2.destroyAllWindows().

4. **Continuous Execution:**
    - The code runs in a continuous loop, ensuring real-time updates of the drawing on both the frame and the canvas window until the user exits the program

5. **Efficient Line Drawing:**
    - It skips drawing lines between consecutive points if either of the points is None, ensuring efficient handling of missing or invalid data.
    - Overall, this code segment enables real-time drawing on the canvas and displays the updated drawing on the frame, providing an interactive and responsive user experience.

◎ **Detailed Summary of whole code:**

1. **Imports:**
    - cv2, numpy, and mediapipe are imported for image processing, numerical computations, and hand landmark detection, respectively.
    - deque from collections is imported to create a deque data structure, which is used to store points of different colors.

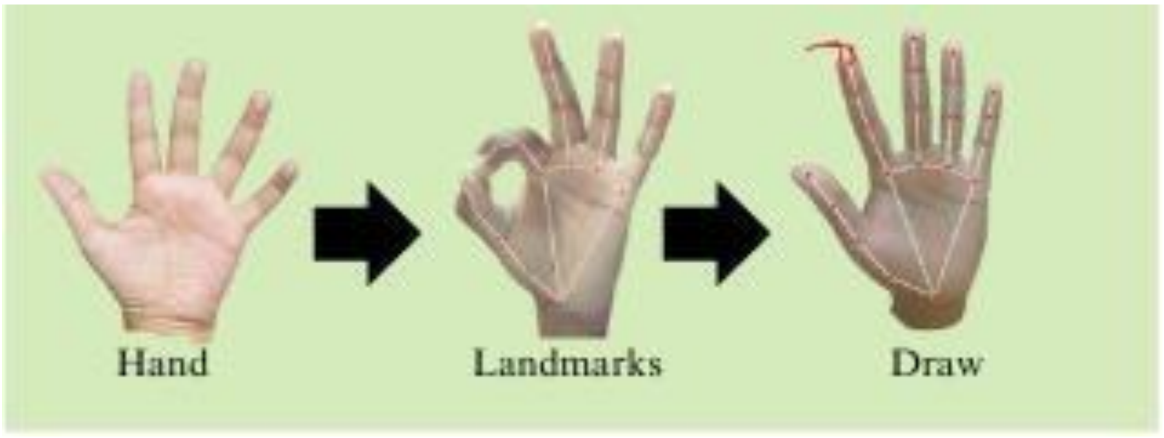


Figure 28 Steps Followed

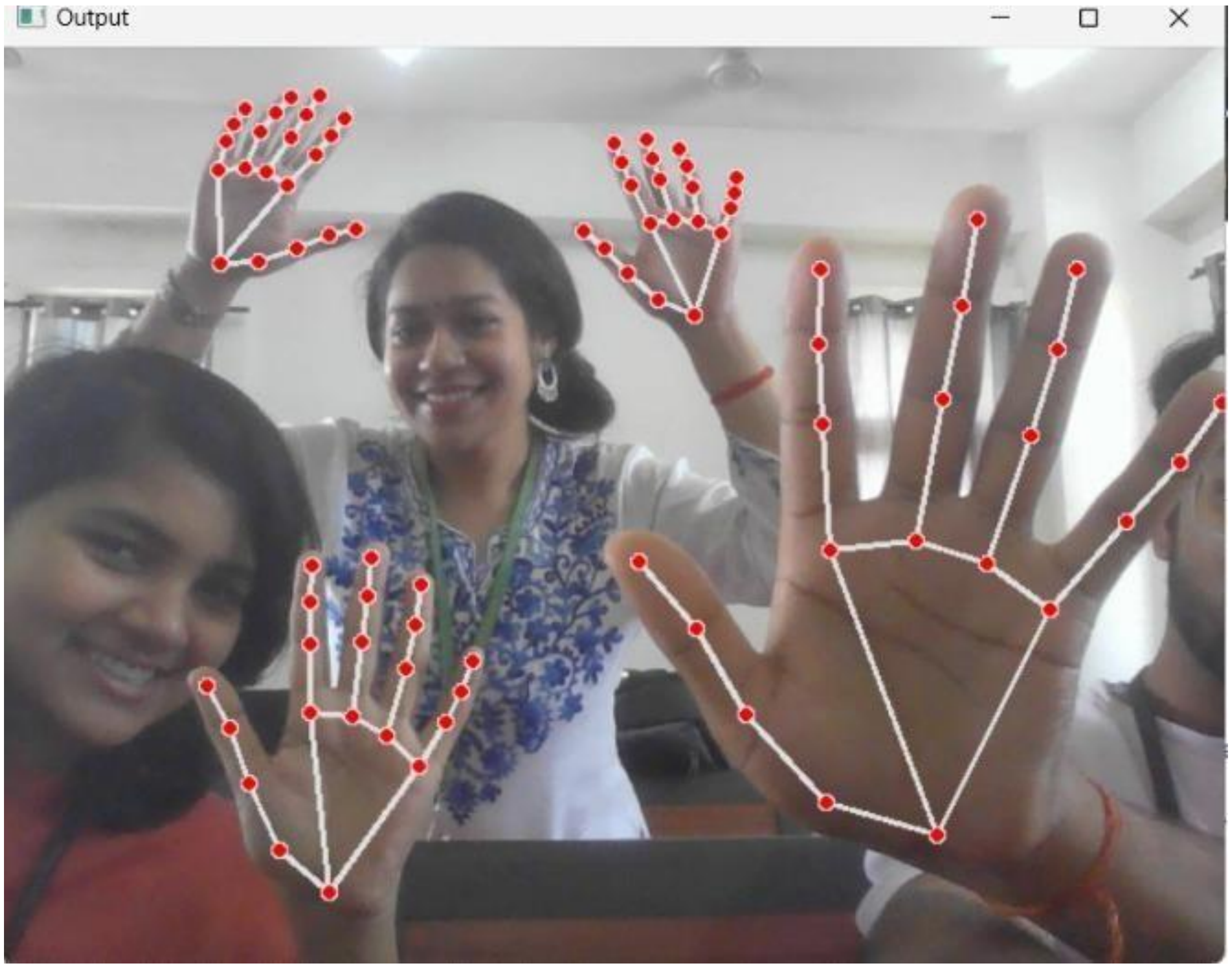
2. **Initialization:**
    - Deque objects are created to store points of different colors, such as blue, green, red, yellow, white, purple, and orange. Each deque has a maximum length of 1024 points.
    - Indices are initialized to keep track of the current position in each deque.
    - A kernel for dilation is created using NumPy, which is used for morphological operations later in the code.
    - Colors and color indices are defined, representing different colors available for drawing.
3. **Canvas Setup:**
    - A blank canvas (paintWindow) is created with OpenCV, where users can draw using hand gestures.
    - Rectangles are drawn to represent different color options and a clear button. These rectangles serve as buttons for selecting colors and clearing the canvas.
    - Text is added to label each button for better user interaction.
4. **Hand Detection Setup:**
    - The Mediapipe hand tracking module is initialized with parameters specifying that only one hand should be detected, and the minimum confidence level required for detection.
    - The webcam feed is initialized using OpenCV to capture live video input.
5. **Main Loop:**
    - Inside the main loop, frames are continuously captured from the webcam.
    - To improve visibility, the frame size is increased, and it is flipped horizontally for a more intuitive user experience.
    - Hand landmarks are detected using the Mediapipe library, providing the coordinates of key points on the hand.
    - Landmarks are extracted and drawn on the frame to visualize hand movement and positioning.
    - The position of the thumb relative to the forefinger is checked to determine user actions, such as selecting a color or clearing the canvas.
    - Based on the user's selection, points are appended to the respective deque corresponding to the chosen color.

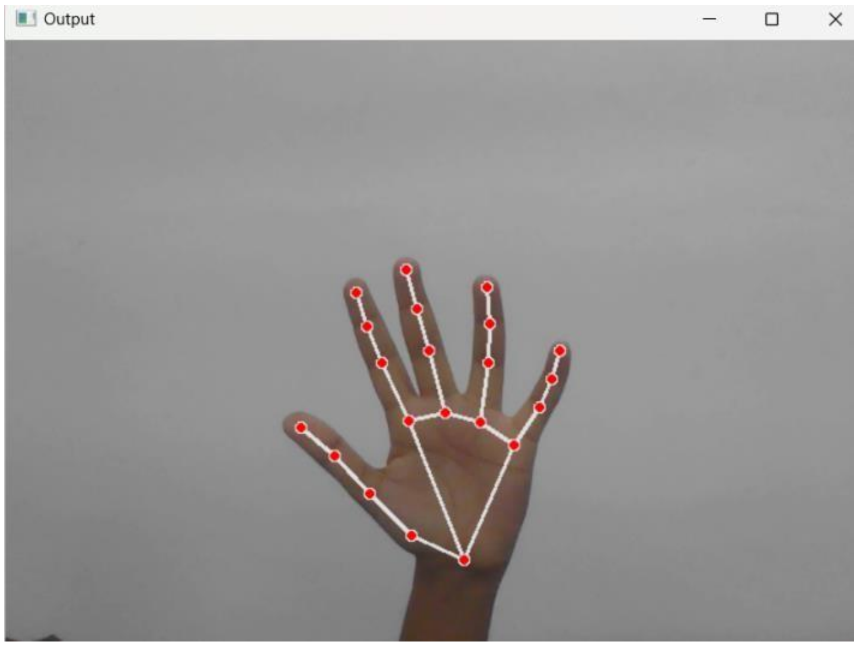
- Lines are drawn on both the frame and the canvas to represent the drawing made by the user's hand gestures.
- The updated frame and canvas are displayed using cv2.imshow().
- The loop continues until the user presses the 'q' key to quit the application.

6. **Cleanup:**
    - Once the main loop exits, the webcam feed is released to free up system resources.
    - All OpenCV windows created during the execution of the program are closed using cv2.destroyAllWindows().

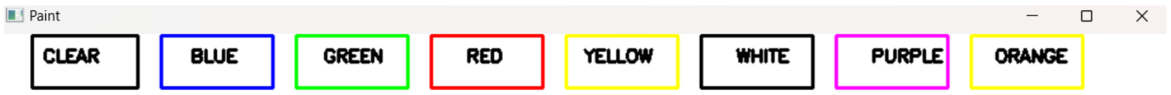
Overall, this code creates an interactive drawing application using hand gestures captured by a webcam. Users can select different colors and draw on a virtual canvas in real-time, providing an engaging and interactive user experience. The implementation demonstrates the integration of computer vision techniques, such as hand landmark detection, with image processing capabilities for interactive applications.

4.2 Results: After initializing mediapipe:

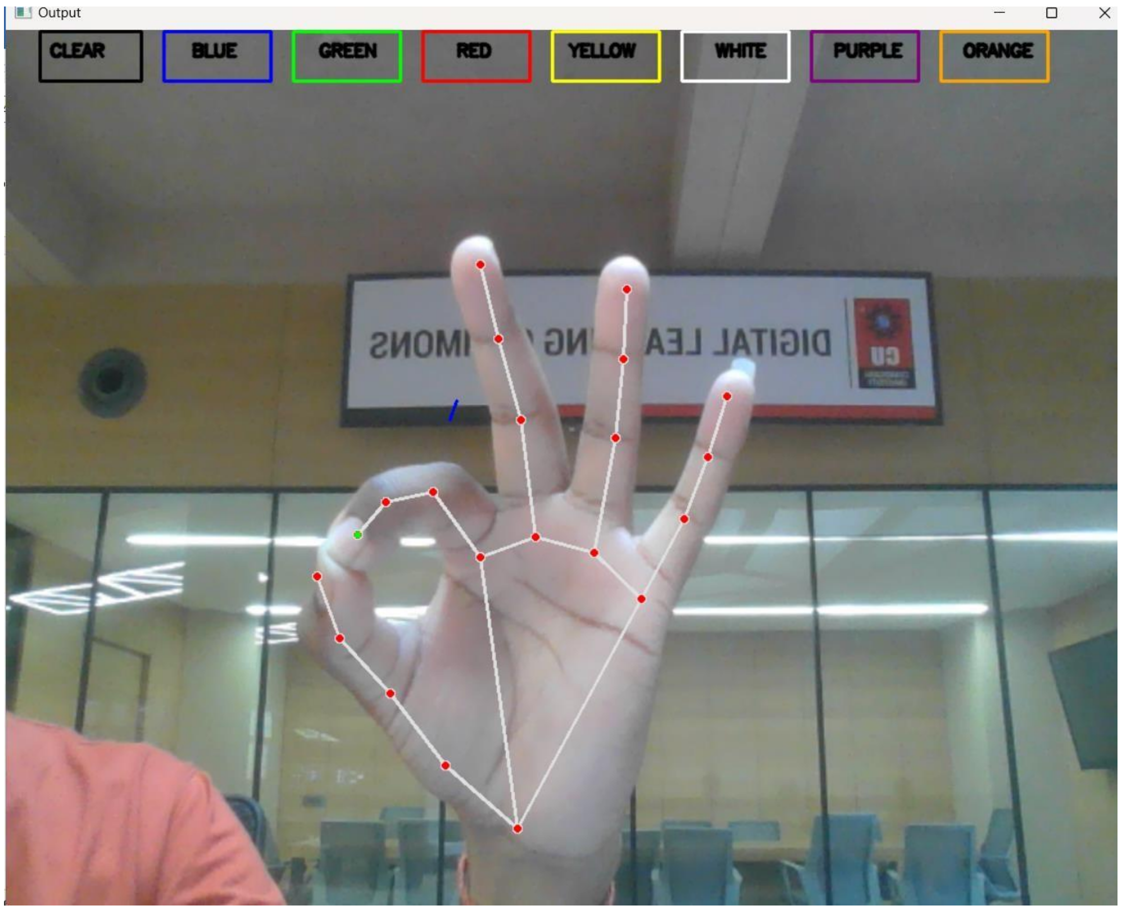




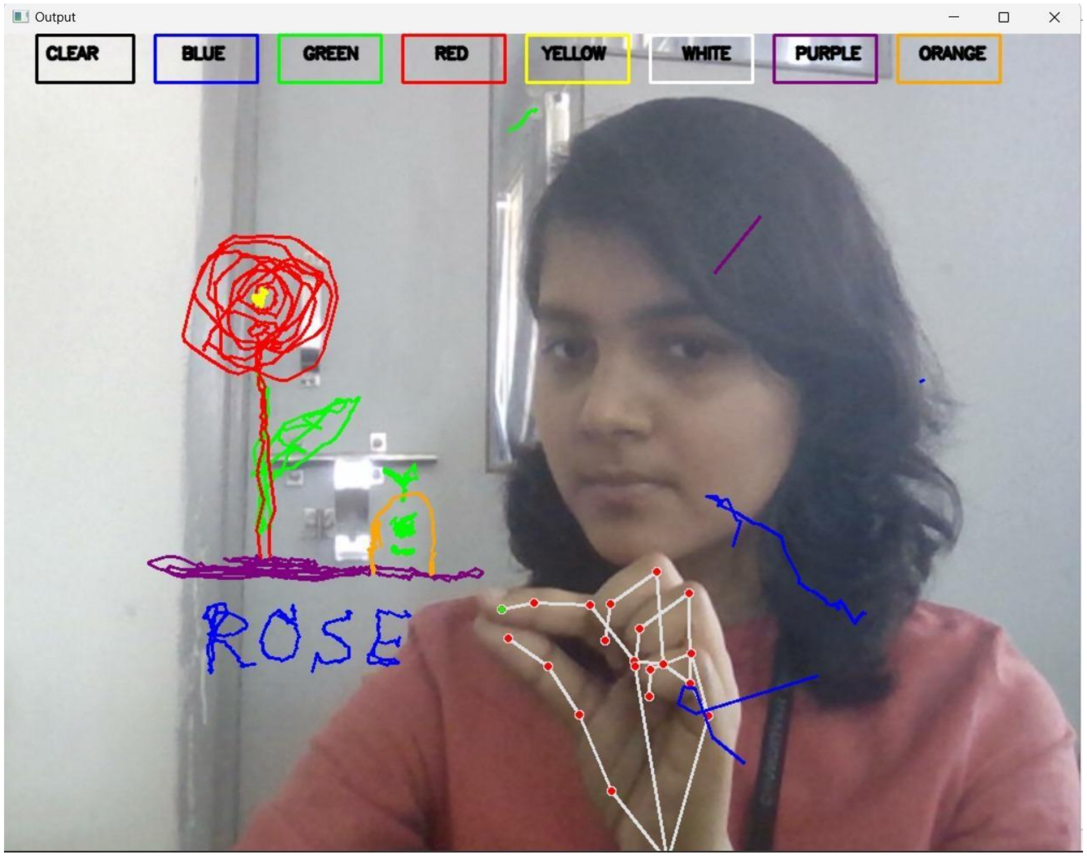
Paint window:

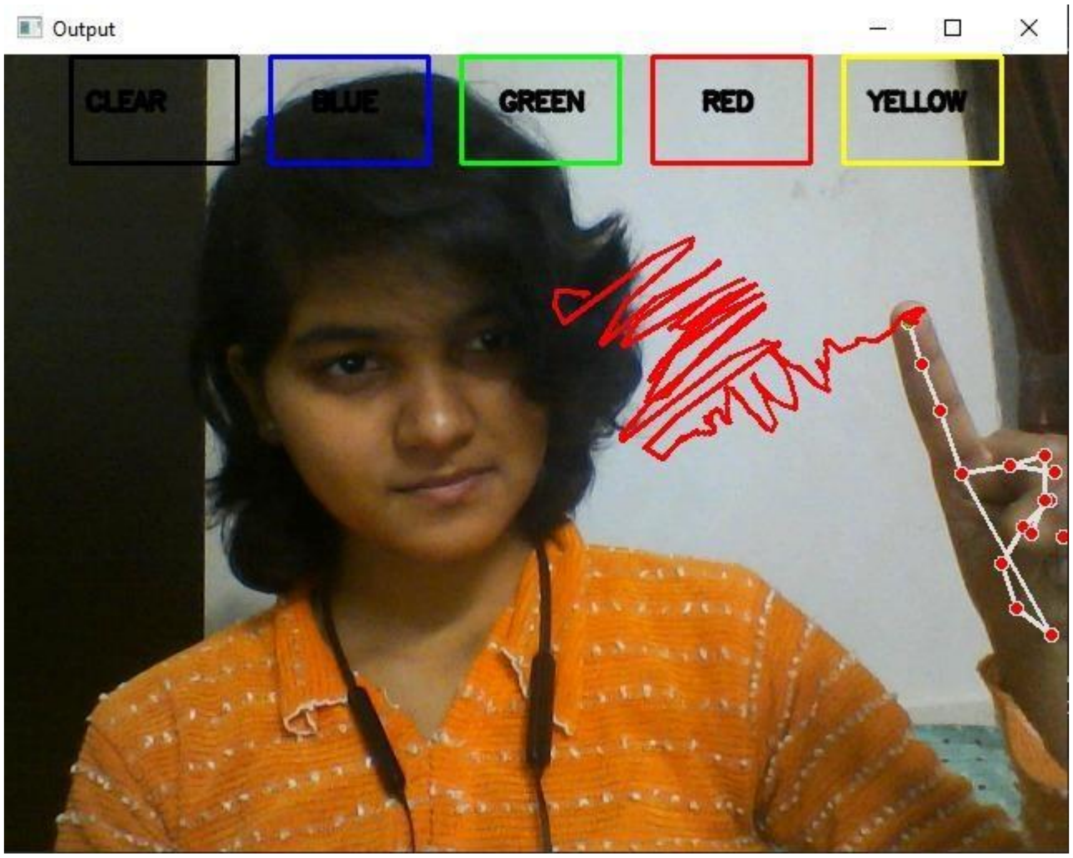


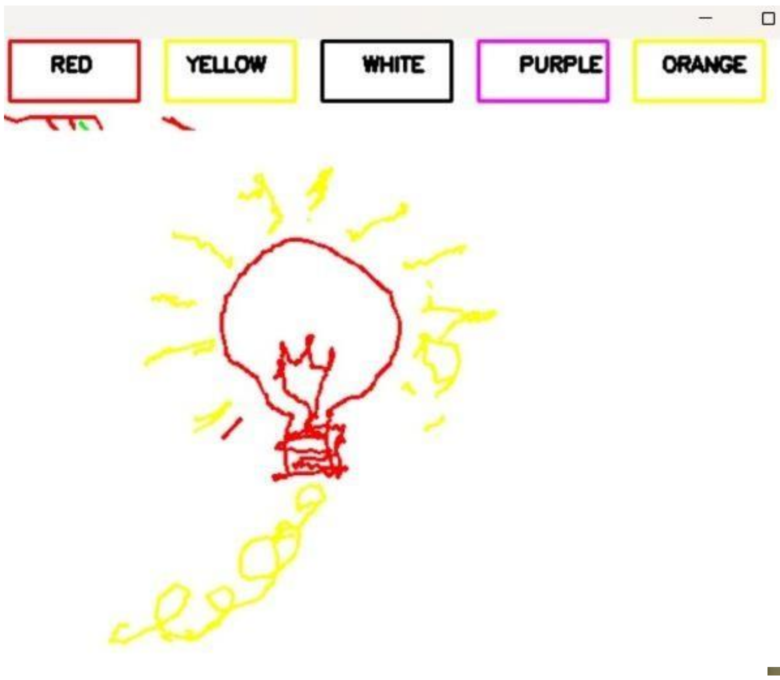
Output Window:

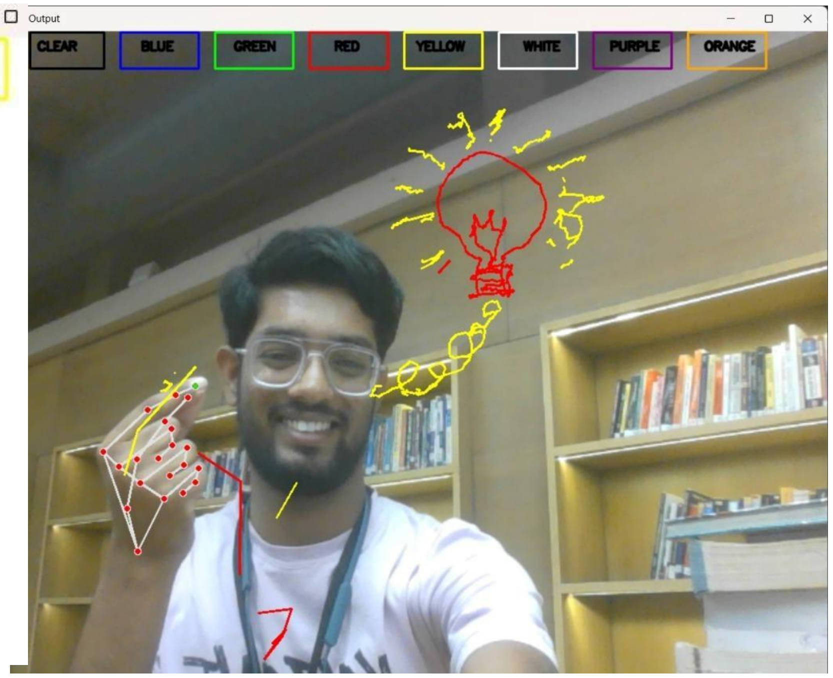


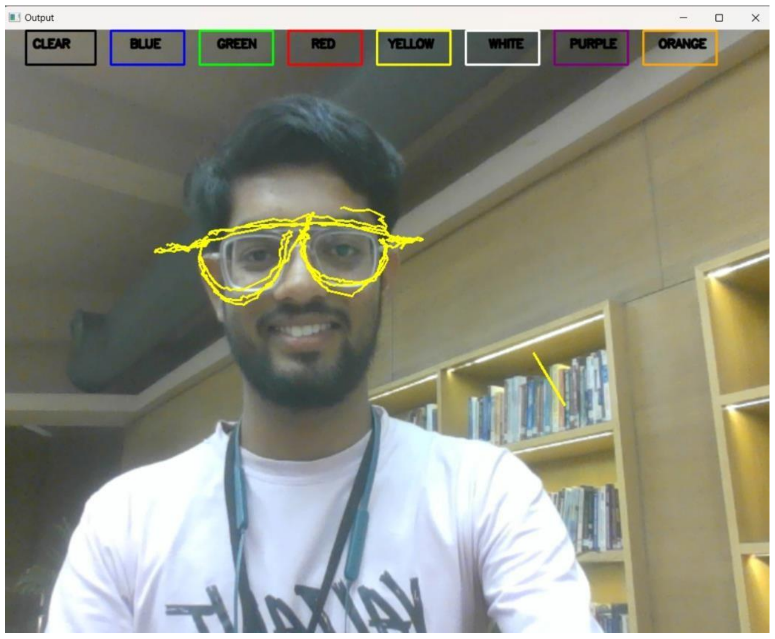
Drawing on camera window and output on paint window:

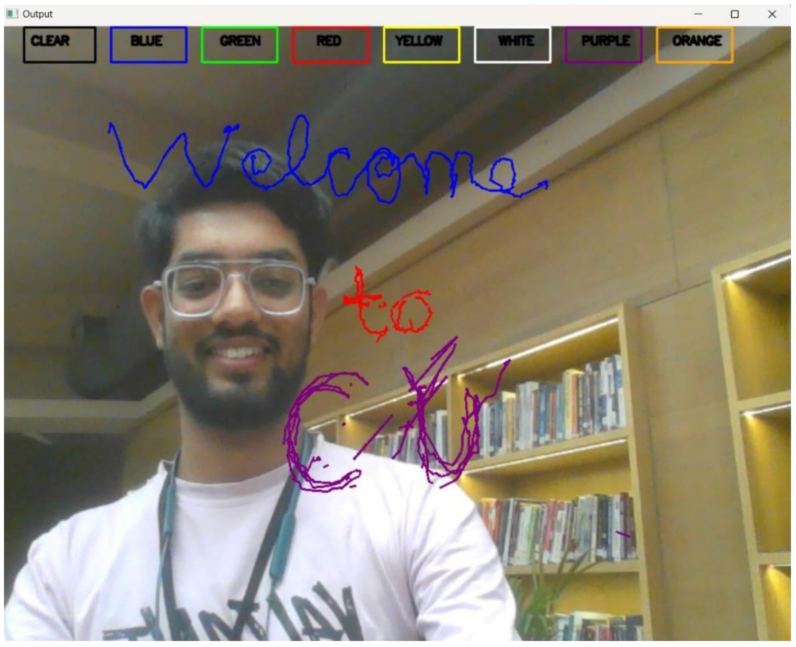

Output
CLEAR
BLUE
GREEN
RED
YELLOW
WHITE
PURPLE
ORANGE
ROSE

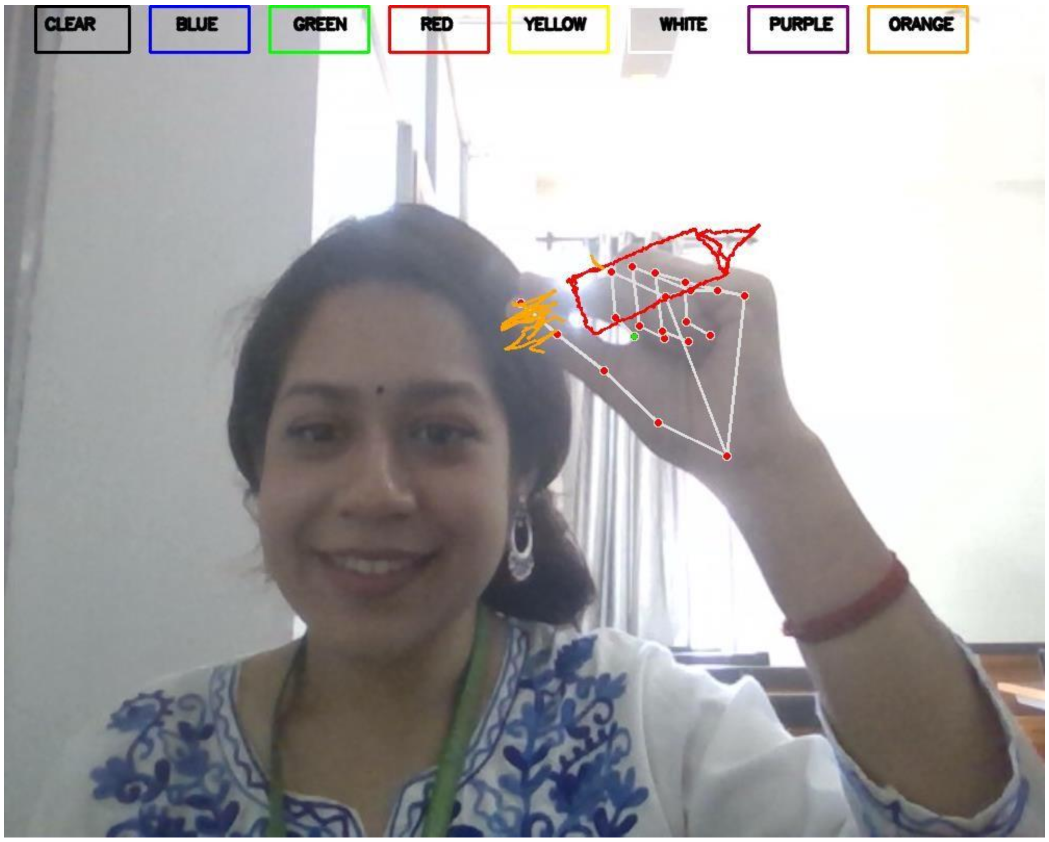

Output
CLEAR
BLUE
GREEN
RED
YELLOW

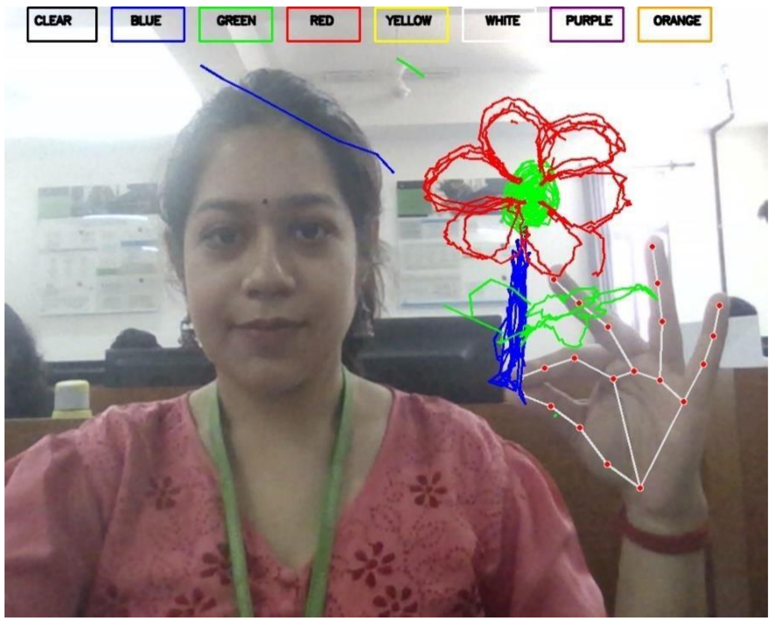

RED
YELLOW
WHITE
PURPLE
ORANGE

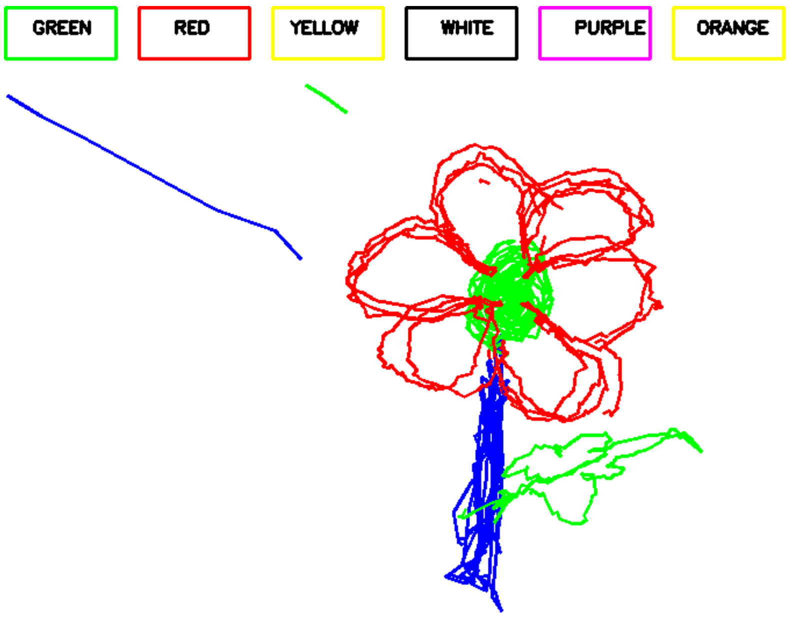

Output
CLEAR
BLUE
GREEN
RED
YELLOW
WHITE
PURPLE
ORANGE

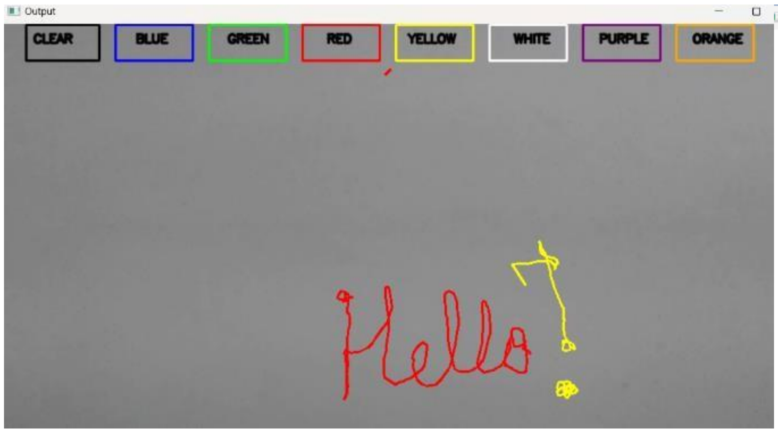

Output
CLEAR
BLUE
GREEN
RED
YELLOW
WHITE
PURPLE
ORANGE

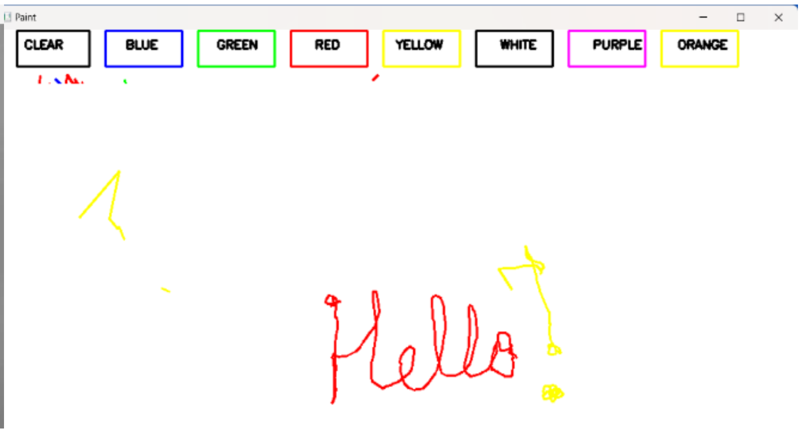

Output
CLEAR
BLUE
GREEN
RED
YELLOW
WHITE
PURPLE
ORANGE
Welcome
to
CU

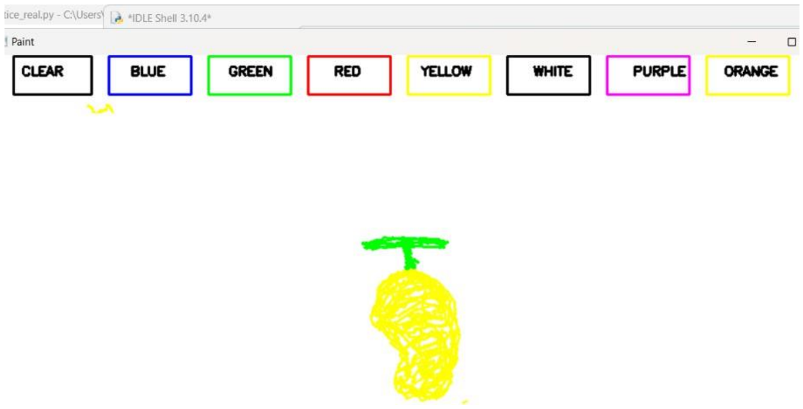

CLEAR
BLUE
GREEN
RED
YELLOW
WHITE
PURPLE
ORANGE

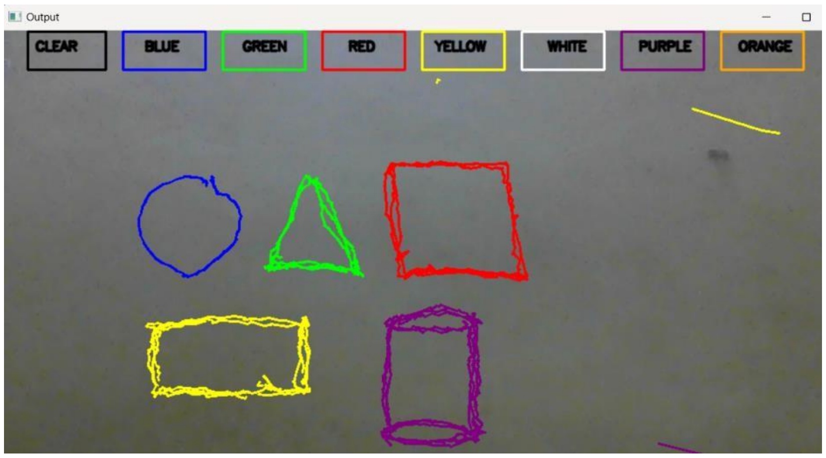

CLEAR
BLUE
GREEN
RED
YELLOW
WHITE
PURPLE
ORANGE

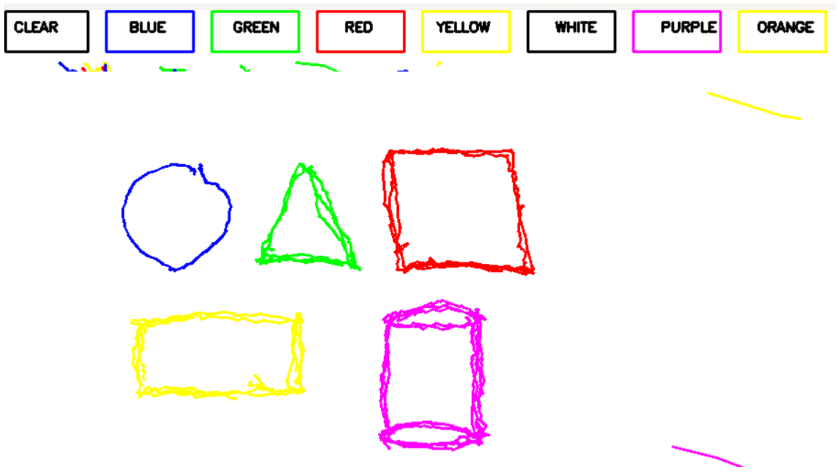

GREEN
RED
YELLOW
WHITE
PURPLE
ORANGE


Output
CLEAR
BLUE
GREEN
RED
YELLOW
WHITE
PURPLE
ORANGE
Hello!

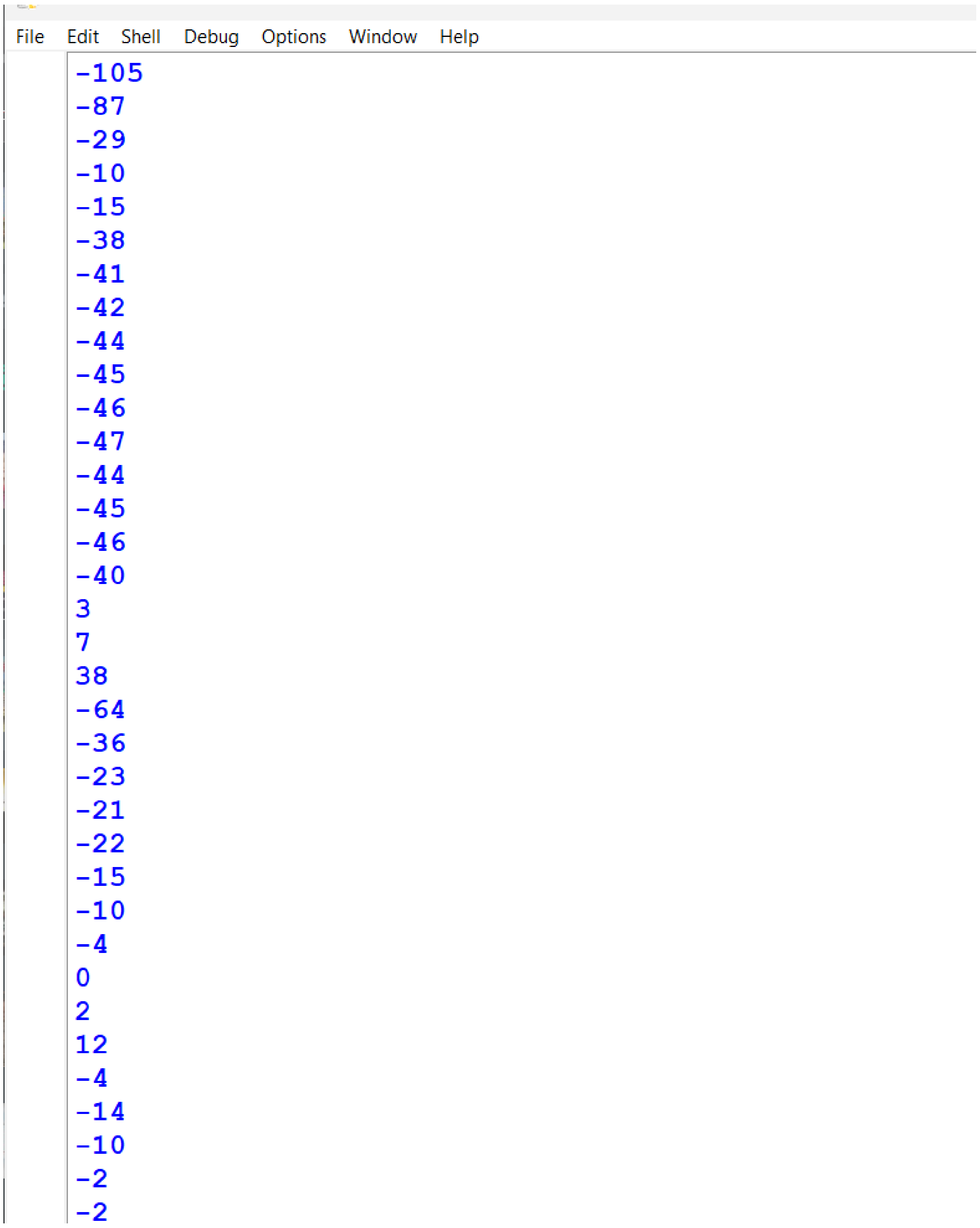

Paint
CLEAR
BLUE
GREEN
RED
YELLOW
WHITE
PURPLE
ORANGE
Hello!


IDLE Shell 3.10.4
Paint
CLEAR
BLUE
GREEN
RED
YELLOW
WHITE
PURPLE
ORANGE


Output
CLEAR
BLUE
GREEN
RED
YELLOW
WHITE
PURPLE
ORANGE


CLEAR
BLUE
GREEN
RED
YELLOW
WHITE
PURPLE
ORANGE

DEQUES is increasing decreasing DYNAMICALLY..!!



```
-105
-87
-29
-10
-15
-38
-41
-42
-44
-45
-46
-47
-44
-45
-46
-40
3
7
38
-64
-36
-23
-21
-22
-15
-10
-4
0
2
12
-4
-14
-10
-2
-2
```

CHAPTER- 5
Conclusion and Future Work

5.1 Conclusion:

In conclusion, the "Air Canvas" project has achieved remarkable success in revolutionizing digital art creation through the integration of machine learning, computer vision, and the Mediapipe framework. Its core objectives, including accessibility, user-friendliness, and precise gesture recognition, have been met with exceptional results. The project's intuitive interface ensures that individuals of all skill levels, including beginners and those with physical disabilities, can effortlessly engage in digital art creation. The robust gesture recognition technology, powered by OpenCV and Mediapipe, allows for accurate real-time tracking of hand and finger movements, enabling users to interact naturally with the digital canvas.

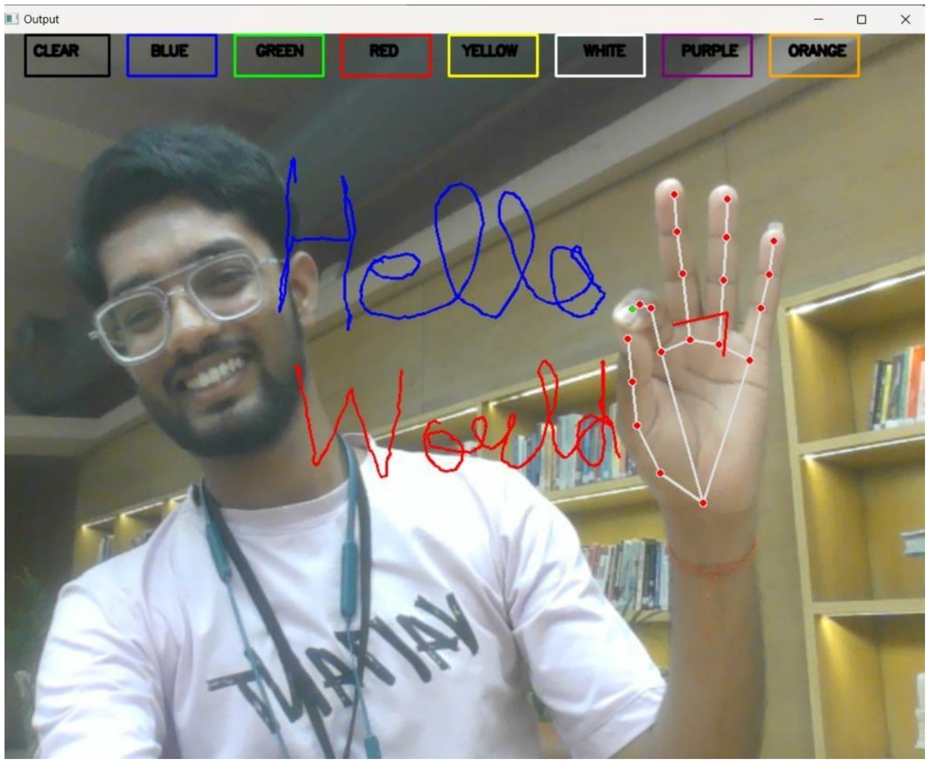


Figure 29 Conclusion

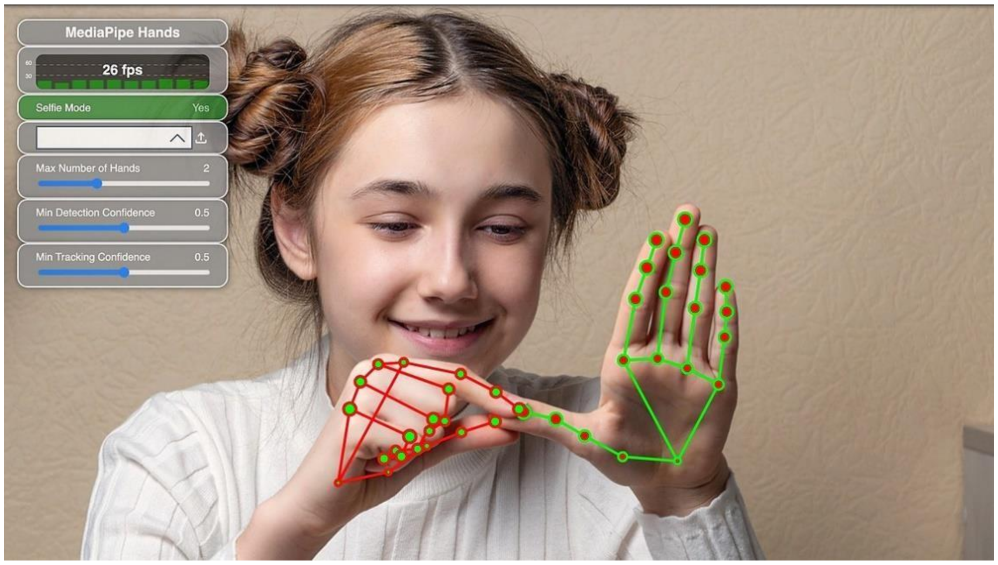
In simple terms, the "Air Canvas" project has achieved something truly remarkable. It's like having a magical digital canvas that you can control with your hand and finger movements! Whether you're an art enthusiast or a student, this technology is a game-changer.

Imagine being in an online class where your teacher needs to explain a complex topic. With "Air Canvas," they can easily draw diagrams and illustrate concepts in real-time using just their hand gestures. It's like having a virtual whiteboard at your fingertips.

The project's success lies in its ability to translate complex computer vision and machine learning concepts into a user-friendly and interactive tool. The MediaPipe library provided robust and efficient solutions for key tasks such as hand tracking, while the machine learning models enabled accurate gesture recognition, allowing users to draw and manipulate the Aircanvas with intuitive hand movements.

The utilization of deques, a double-ended queue from the collections library, played a pivotal role in optimizing the processing of sequential data in real-time. The deque's ability to efficiently add and remove elements from both ends facilitated the implementation of sliding windows for tasks such as hand gesture recognition and tracking. This not only enhanced the responsiveness of the Aircanvas application but also contributed to memory efficiency, crucial for real-time applications.

What makes it even more incredible is that it's super easy to use. You don't need any special equipment or training. Just a computer with a webcam, and you're ready to go. This means that even if you're new to digital art or technology, you can still create and learn with ease.



Selfie Mode Yes
Max Number of Hands 2
Min Detection Confidence 0.5
Min Tracking Confidence 0.5

Figure 30 Mediapipe Advancements

But here's the real magic – it's not just for online classes. "Air Canvas" can be used in all sorts of creative ways. You can use it to draw, paint, or even play educational games. It opens up a world of possibilities for both learning and fun. In a nutshell, the "Air Canvas" project has successfully combined technology, creativity, and accessibility. It's not just a tool for artists; it's a tool for everyone. Whether you're in an online class or just exploring your artistic side, "Air Canvas" is here to make your digital journey a breeze.

The project's impact extends beyond just digital art creation. It has the potential to revolutionize various industries and sectors. For instance, in the field of education, "Air Canvas" can serve as a dynamic and interactive teaching tool. Teachers can use it to engage students in immersive learning experiences, making complex concepts more understandable through visual aids and interactive demonstrations.

Moreover, in the realm of remote collaboration and communication, "Air Canvas" offers an innovative solution. Teams working remotely can use it for brainstorming sessions, collaborative design projects, or virtual presentations. Its intuitive interface and real-time drawing capabilities facilitate effective communication and idea sharing, regardless of geographical distances.

Furthermore, the project's success underscores the power of interdisciplinary collaboration. By integrating concepts from machine learning, computer vision, and user interface design, the "Air Canvas" team has created a holistic solution that addresses diverse user needs and preferences. This interdisciplinary approach not only enhances the project's functionality but also fosters innovation and creativity in problem-solving.

In terms of future developments, the "Air Canvas" project opens doors to exciting possibilities. With ongoing advancements in machine learning and computer vision technologies, there's potential for further refinement and enhancement of gesture recognition algorithms. Additionally, integrating features such as voice commands or augmented reality elements could further elevate the user experience and expand the project's capabilities.

Overall, the "Air Canvas" project exemplifies the transformative potential of technology in empowering creativity, accessibility, and collaboration. By harnessing the power of machine learning and computer vision, it has paved the way for a new era of digital art creation and interactive communication. As technology continues to evolve, projects like "Air Canvas" will continue to push the boundaries of innovation, shaping the future of human-computer interaction and creative expression.

5.2 Limitation:

While the "Aircanvas using Machine Learning and MediaPipe" project is a commendable effort that brings together computer vision, machine learning, and interactive applications, it is important to acknowledge certain limitations that may impact its performance and usability:

- **Dependency on Lighting Conditions:** The project's reliance on visual data from cameras makes it susceptible to variations in lighting conditions. Changes in ambient light levels can affect the accuracy of hand tracking and gesture recognition, potentially leading to suboptimal performance in different environments.
- **Gesture Recognition Accuracy:** The accuracy of the gesture recognition system may be influenced by the complexity of hand movements and variations in individual gestures. Fine-tuning the machine learning models and expanding the training dataset could help improve recognition accuracy but may not eliminate all potential challenges.

- **Real-time Processing Requirements:** The real-time nature of the Aircanvas application demands high processing speed. On less powerful hardware or in resource-constrained environments, there may be a noticeable lag between hand movements and on-screen updates, affecting the overall user experience.
- **Sensitivity to Hand Occlusion:** The application's ability to accurately track hand movements may be compromised when hands overlap or occlude each other.

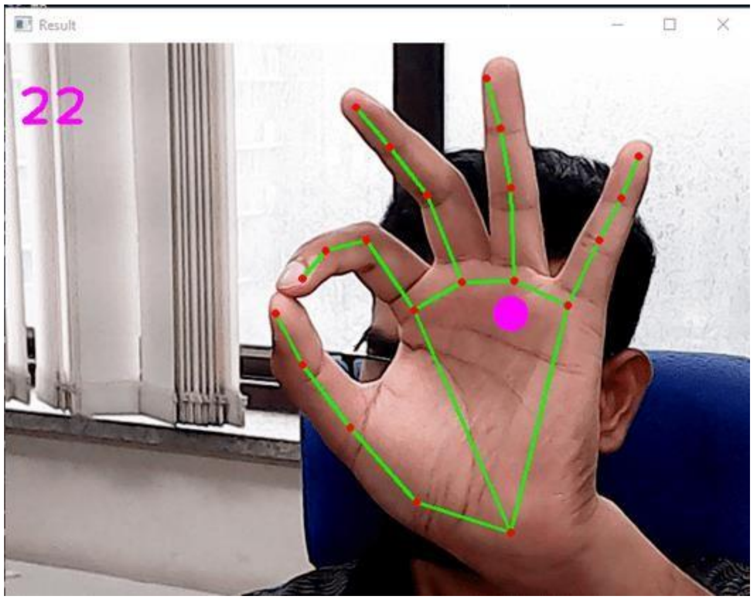


Figure 31 Landmark Condtion

- **Hardware Dependency:** The project's performance may vary across different camera hardware. The accuracy and responsiveness of hand tracking are influenced by the quality and specifications of the camera used. Compatibility issues may arise when deploying the application on diverse devices.

5.3 Future Scope:

The "Aircanvas using Machine Learning and MediaPipe" project lays a solid foundation for interactive applications that merge computer vision, machine learning, and creative expression. There are several exciting future scopes and enhancements that can be explored to further advance the capabilities and versatility of the Aircanvas concept:

- **Enhanced Gesture Vocabulary:** Expand the range of recognized gestures to offer users a more diverse and expressive set of interactions. This could include recognizing specific shapes, symbols, or even alphanumeric characters, making the Aircanvas more versatile for different creative tasks.

- **Dynamic Gesture Customization:** Allow users to define and customize their own gestures for specific actions or commands. This could involve a user-friendly interface for gesture mapping and training, providing a personalized and adaptable experience.
- **Multi-User Collaboration:** Enable collaborative drawing sessions where multiple users can interact with the Aircanvas simultaneously.
- **Real-time Collaboration Across Devices:** Extend the application to support real-time collaboration across different devices. Users could draw together on the same canvas using Aircanvas on their smartphones, tablets, or other devices, fostering collaborative creativity.

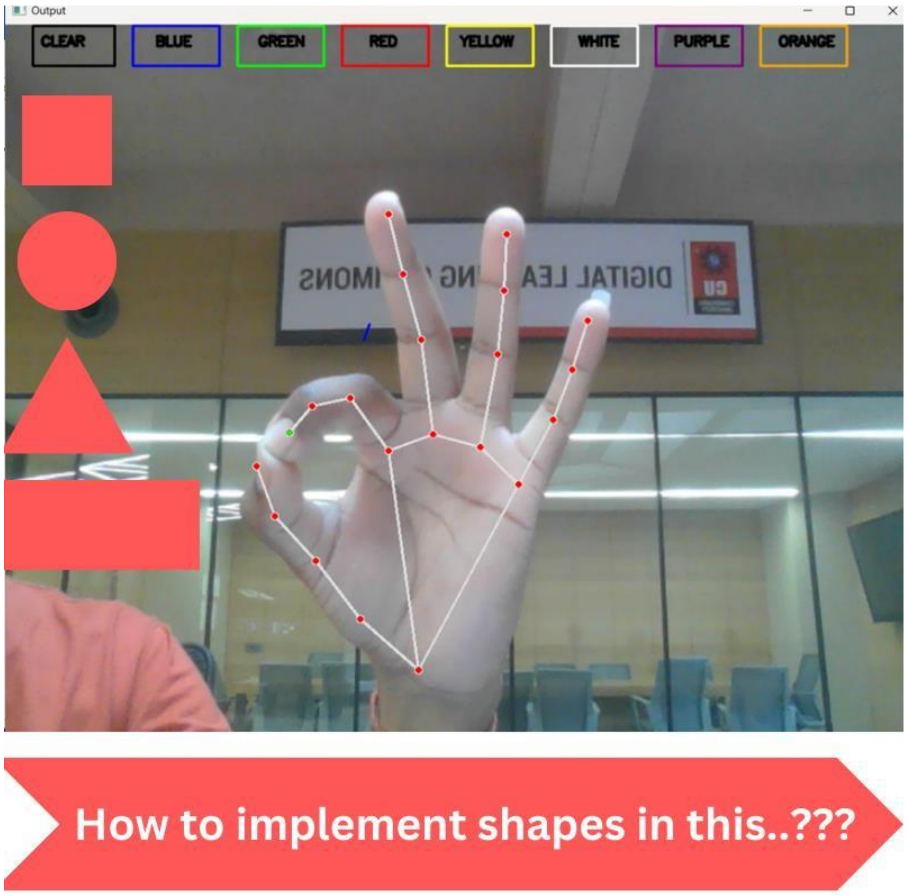


Figure 32 Integration of Shapes with AIR Canvas

- **Gamification and Educational Applications:** Develop gamified features or educational modules to make the Aircanvas application engaging for users of all ages. This could include interactive tutorials, challenges, or collaborative drawing games.

Continued innovation in these directions can transform the Aircanvas project into a versatile and widely applicable tool, opening up new possibilities for creative expression, collaboration, and human-computer interaction. As technology advances, exploring these future scopes will contribute to the evolution of interactive and immersive applications that leverage the capabilities of computer vision and machine learning.

6. REFERENCES

1. V. RamachandraH, G. Balaraju, K. Deepika, S. Navya M and S. R. Sebastian, "Virtual Air Canvas Using OpenCV and Mediapipe," 2022 International Conference on Futuristic Technologies (INCOFT), Belgaum, India, 2022, pp. 1-4, doi: 10.1109/INCOFT55651.2022.10094385.

2. Z. Ren, J. Yuan, J. Meng and Z. Zhang, "Robust Part-Based Hand Gesture Recognition Using Kinect Sensor," in IEEE Transactions on Multimedia, vol. 15, no. 5, pp. 1110-1120, Aug. 2013, doi: 10.1109/TMM.2013.2246148.

3. Yu, C. W., Liu, C. H., Chen, Y. L., Lee, P., & Tian, M. S. (2018). Vision-based Hand Recognition Based on ToF Depth Camera. Smart Science, 6(1), 21–28. https://doi.org/10.1080/23080477.2017.1402537

4. Zhang, F., Bazarevsky, V., Vakunov, A., Tkachenka, A., Sung, G., Chang, C. L., & Grundmann, M. (2020). Mediapipe hands: On-device real-time hand tracking. arXiv preprint arXiv:2006.10214.

5. Culjak, I., Abram, D., Pribanic, T., Dzapo, H., & Cifrek, M. (2012, May). A brief introduction to OpenCV. In 2012 proceedings of the 35th international convention MIPRO (pp. 1725-1730). IEEE.

6. García, G. B., Suarez, O. D., Aranda, J. L. E., Tercero, J. S., Gracia, I. S., & Enano, N. V. (2015). Learning image processing with OpenCV. Packt Publishing.

7. Veluri, R. K., Sree, S. R., Vanathi, A., Aparna, G., & Vaidya, S. P. (2022, March). Hand gesture mapping using mediapipe algorithm. In Proceedings of Third International Conference on Communication, Computing and Electronics Systems: ICCCES 2021 (pp. 597-614). Singapore: Springer Singapore.

8. Zimmermann, S., & Brox, T. (2017). Handpose: 3D Hand Pose Estimation Using Multi-view Geometry. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops.

9. Oka, K., Sato, Y., & Koike, H. (2002). Real-time fingertip tracking and gesture recognition. IEEE Computer graphics and Applications, 22(6), 64-71.

10. : García-García, A., Herrera, F., & Porcel, C. (2018). Real-time Hand Gesture Detection and Classification Using Convolutional Neural Networks. Expert Systems with Applications, 105, 111-123.

11. Mueller, F., Mehta, D., & Sotnychenko, O. (2018). Efficient Hand Pose Estimation from a Single Depth Image. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.

12. Parvathy, P., Subramaniam, K., Prasanna Venkatesan, G. K. D., Karthikaikumar, P., Varghese, J., & Jayasankar, T. (2021). Development of hand gesture recognition system using machine learning. Journal of Ambient Intelligence and Humanized Computing, 12, 6793-6800.

13. Parvathy, P., Subramaniam, K., Prasanna Venkatesan, G. K. D., Karthikaikumar, P., Varghese, J., & Jayasankar, T. (2021). Development of hand gesture recognition system using machine learning. Journal of Ambient Intelligence and Humanized Computing, 12, 6793-6800.

14. Gosavi, J., Kadam, N., Shetty, A., Verekar, A., & Vishwakarma, P. (2023, August). Contactless Gesture Recognition Using Air Canvas. In International Conference on ICT for Sustainable Development (pp. 337-347). Singapore: Springer Nature Singapore.

15. Culjak, I., Abram, D., Pribanic, T., Dzapo, H., & Cifrek, M. (2012, May). A brief introduction to OpenCV. In 2012 proceedings of the 35th international convention MIPRO (pp. 1725-1730). IEEE.

16. B. A. Kumar, T. Vinod and M. S. Rao, "Interaction through Computer Vision Air Canvas," 2022 International Conference on Advancements in Smart, Secure and Intelligent Computing (ASSIC), Bhubaneswar, India, 2022, pp. 1-3, doi: 10.1109/ASSIC55218.2022.10088318.

17. P. Rai, R. Gupta, V. Dsouza and D. Jadhav, "Virtual Canvas for Interactive Learning using OpenCV," 2022 IEEE 3rd Global Conference for Advancement in Technology (GCAT), Bangalore, India, 2022, pp. 1-5, doi: 10.1109/GCAT55367.2022.9971903.

18. A. Dash et al., "AirScript - Creating Documents in Air," 2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR), Kyoto, Japan, 2017, pp. 908-913, doi: 10.1109/ICDAR.2017.153.

19. Y. Quiñonez, C. Lizarraga and R. Aguayo, "Machine Learning solutions with MediaPipe," 2022 11th International Conference On Software Process Improvement (CIMPS), Acapulco, Guerrero, Mexico, 2022, pp. 212-215, doi: 10.1109/CIMPS57786.2022.10035706.

20. P. Padhi and M. Das, "Hand Gesture Recognition using DenseNet201-Mediapipe Hybrid Modelling," 2022 International Conference on Automation, Computing and Renewable Systems (ICACRS), Pudukkottai, India, 2022, pp. 995-999, doi: 10.1109/ICACRS55517.2022.10029038.

21. M. Gnanasekera, "Computer vision based hand movement capturing system," 2013 8th International Conference on Computer Science & Education, Colombo, Sri Lanka, 2013, pp. 416-421, doi: 10.1109/ICCSE.2013.6553948.

22. Y. Y. Pang, N. A. Ismail and P. L. S. Gilbert, "A Real Time Vision-Based Hand Gesture Interaction," 2010 Fourth Asia International Conference on Mathematical/Analytical Modelling and Computer Simulation, Kota Kinabalu, Malaysia, 2010, pp. 237-242, doi: 10.1109/AMS.2010.55.

23. https://developers.google.com/mediapipe

24. https://learnopencv.com/introduction-to-mediapipe/

25. https://azure.microsoft.com/en-in/resources/cloud-computing dictionary/whatiscomputervision/#:~:text=Computer%20vision%20is%20a%20field,tasks%20that%20replicate%20human%20capabilities.

26. https://www.ibm.com/topics/computer-vision

27. https://opencv.org/

28. https://numpy.org/

29. https://www.tutorialspoint.com/python_data_structure/python_dequeue.htm#:~:text=A%20double%2Dended%20queue%2C%20or,restricted%20to%20a%20single%20end.

30. https://docs.opencv.org/4.x/index.html

31. https://mediapipe.readthedocs.io/en/latest/

32. https://developers.google.com/mediapipe/api/solutions

33. https://github.com/google/mediapipe/blob/master/docs/getting_started/python_framework.md

34. https://github.com/google/mediapipe/blob/master/docs/index.md

35. https://developers.google.com/mediapipe/solutions/vision/gesture_recognizer

36. https://learnopencv.com/gesture-control-in-zoom-call-using-mediapipe/

37. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8321080/

38. https://core.ac.uk/download/pdf/55638595.pdf

39. http://14.139.189.217/m2/design/6902.06.design.pdf

40. https://pypi.org/project/mediapipe/

CHANDIGARH UNIVERSITY

Discover. Learn. Empower.

Thank You!