***A***

***Project report***

***On***

**VIRTUAL MOUSE USING HAND GESTURES AND EYE MOVEMENTS**

***submitted in partial fulfilment of the Requirements for the Award of the Degree of***

**BACHELOR OF TECHNOLOGY**

***In***

# COMPUTER SCIENCE AND ENGINEERING (Artificial Intelligence and Machine Learning)

***By***

|  |  |
| --- | --- |
| **B.SOUMYA** | **21VD5A6615** |
| **P.MAHESH BABU** | **20VD1A6619** |
| **V.AKSHAY KUMAR** | **20VD1A6605** |
| **K.NITHIN** | **21VD5A6608** |

**Under the Guidance of**

**Mr. A. NARAHARI,**

**Assistant professor,**

**Department of Computer Science and Engineering.**



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING(AI/ML)**

### JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY HYDERABAD

### UNIVERSITY COLLEGE OF ENGINEERING MANTHANI

**Pannur (Vil), Ramagiri (Mdl), Peddapally-505212, Telangana (India).**

**2023-2024**

**JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY HYDERABAD**

**UNIVERSITY COLLEGE OF ENGINEERING MANTHANI**

**Pannur (Vil), Ramagiri (Mdl), Peddapally-505212, Telangana (India).**

## DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

(Artificial Intelligence and Machine Learning)



# DECLARATION BY THE CANDIDATE

We,

|  |  |
| --- | --- |
| **B.SOUMYA** | **21VD5A6615** |
| **P. MAHESH BABU** | **20VD1A6619** |
| **V.AKSHAY KUMAR** | **20VD1A6605** |
| **K.NITHIN** | **21VD5A6608** |

hereby declare that the project report entitled ***“Virtual Mouse using Hand Gestures and Eye - Movements”*** under the guidance of **Mr. A. NARAHARI,** Assistant Professor, Department of Computer Science and Engineering, JNTUH University College of Engineering Manthani, submitted in partial fulfillment of the award of the Degree of Bachelor of Technology in Computer Science and Engineering (AI & ML).

This is a record of bonafide work carried out by us and the results embodied in this project report has not been reproduced or copied from any source. The results embodied in this project have not been submitted to any other University or Institute for the award of any degree.

|  |  |
| --- | --- |
| **B.SOUMYA** | **21VD5A6615** |
| **P.MAHESH BABU** | **20VD1A6619** |
| **V.AKSHAY KUMAR** | **20VD1A6605** |
| **K.NITHIN** | **21VD5A6608** |

## JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY HYDERABAD

## UNIVERSITY COLLEGE OF ENGINEERING MANTHANI

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**(AI/ML)**



# CERTIFICATE

This is to certify that the project report entitled ***“Virtual Mouse using Hand Gestures and Eye Movements”*** is a bonafide work carried out by

|  |  |
| --- | --- |
| **B.SOUMYA** | **21VD5A6615** |
| **P. MAHESH BABU** | **20VD1A6619** |
| **V.AKSHAY KUMAR** | **20VD1A6605** |
| **K.NITHIN** | **21VD5A6608** |

in partial fulfilment of the requirements for the degree of **BACHELOR OF TECHNOLOGY** in **COMPUTER SCIENCE AND ENGINEERING(Artificial Intelligence and Machine Learning)** by the **Jawaharlal Nehru Technological University Hyderabad University College of Engineering, Manthani** during the academic year 2023-24.

The results of the investigation enclosed in this report have been verified and found satisfactory. The results embodied in this project report have not been submitted to any other University or Institute for the award of any degree.

**INTERNAL GUIDE HEAD OF THE DEPARTMENT**

**(Mr. A. NARAHARI**

**Assistant Professor in CSE)**

**EXTERNAL EXAMINER**

**DATE:**

## ACKNOWLEDGMENT

We express our sincere gratitude to **Dr. CH. SRIDHAR REDDY, Professor of Mechanical Engineering, Principal & Head of the department of Computer Science and Engineering(AIML)**, JNTUH University College of Engineering Manthani for encouraging and giving permission to accomplish our project successfully.

We express our profound gratitude and thanks to our project guide **Mr. Narahari Ajmeera Assistant Professor CSE Department**, JNTUH University College of Engineering Manthani for his constant help, personal supervision, expert guidance and consistent encouragement throughout this project which enabled us to complete our project successfully in time. We are immensely obliged to other faculty members of the Department of CSE for their kind cooperation.

We also take this opportunity to thank other faculty members of CSE Department for their kind co-operation.

We wish to convey our thanks to one and all those who have extended their helping hands directly and indirectly in completion of our project..

|  |  |
| --- | --- |
| **B. SOUMYA** | **21VD5A6615** |
| **P. MAHESH BABU** | **20VD1A6619** |
| **V. AKSHAY KUMAR** | **20VD1A6605** |
| **K. NITHIN** | **21VD5A6608** |

## ABSTRACT

The concept of a virtual mouse has undergone significant evolution, transforming the way humans interact with digital devices. In the past, traditional computer mouse were essential peripherals for navigating graphical user interface(GUIs).

Since most of the physically disabled persons are finding it difficult to use traditional mouses on their devices as well as traditional mouse usage poses certain concerns in times like Covid-19 pandemic to be contact free. So there is a need for simplifying the operability and usability of a computer.

The implementation of virtual mouse using a combination of OpenCV, Mediapipe, PyAutoGUI, and PyCaw, enabling users to control cursor movement, clicks, and system volume through hand gestures, providing a novel and intuitive interaction experience.

In conclusion, the utilization of a virtual mouse as an alternative to the traditional physical mouse presents a compelling advancement in human-computer interaction. This innovation not only reduces the reliance on physical devices but also opens avenues for enhanced accessibility and user comfort. This shift in interaction paradigm aligns with the growing need for contactless and hygienic computing solutions, particularly in the context of the pandemic

## TABLE OF CONTENTS

|  |  |  |
| --- | --- | --- |
| **S.No** | **CONTENTS** | **PAGE NO** |
| **i.** | **TITLE PAGE** | i |
| **ii.** | **DECLARATION PAGE** | ii |
| **iii.** | **CERTIFICATE** | iii |
| **iv.** | **ACKNOWLEDGEMENT** | iv |
| **v.** | **ABSTRACT** | v |
| **1.** | **INTRODUCTION** | 1 |
|  | 1.1 Introduction | 2 |
|  | 1.2 Problem Statement | 3 |
|  | 1.3 Objectives | 3 |
| **2.** | **LITERATURE SURVEY** | 4 |
|  | 2.1 Existing System | 5 |
|  | 2.2 Limitations of Existing Systems | 7 |
| 3. | SYSTEM ANALYSIS | 8 |
|  | 3.1 Proposed System | 9 |
|  | 3.2 Methodology of Proposed System | 10 |
|  | 3.3 Advantages of Proposed System | 11 |
| **4**. | **SYSTEM REQUIREMENT ANALYSIS** | 12 |
|  | 4.1 Functional Requirements | 13 |
|  | 4.2 Non-Functional Requirements | 13 |
|  | 4.3 Hardware Requirements | 14 |
|  | 4.4 Software Requirements | 15 |
| **5.** | **SYSTEM DESIGN** | 20 |
|  | 5.1 Introduction | 21 |
|  | 5.2 UML Diagrams | 21 |
| **6.** | **IMPLEMENTATION** | 25 |
|  | 6.1 Sample code | 26 |
| **7.** | **TESTING** | 37 |
|  | 7.1 Introduction | 38 |
|  | 7.2 Types of Software Testing | 38 |
|  | 7.3 Testing on Our System | 39 |
| **8.** | **RESULTS** | 40 |
| **9.** | **CONCLUSION** | 45 |
| **10.** | **BIBLIOGRAPHY** | 47 |

## LIST OF FIGURES

|  |  |  |  |
| --- | --- | --- | --- |
| **FIG NO** | | **FIGURE NAME** | **PAGE NO** |
| Fig 3.1 | Land Marks Of Hand | 9 |
| Fig 3.2 | Land Marks of Eyes | 9 |
| Fig 5.1 | Use Case Diagram | 22 |
| Fig 5.2 | System Architecture | 23 |
| Fig 5.3 | Activity diagram | 24 |
| Fig 5.4 | Class Diagram | 25 |
| Fig 8.1 | Home Page of our Application | 41 |
| Fig 8.2 | Land Marks Of Hand | 41 |
| Fig 8.3 | Gesture to Move Cursor | 42 |
| Fig 8.4 | Gesture to Right Click | 42 |
| Fig 8.5 | Gesture to Left Click | 43 |
| Fig 8.6 | Gesture to Control Brighness and Volume | 43 |
| Fig 8.7 | Gesture to Scroll | 44 |
| Fig 8.8 | Land Marks Of Eyes | 44 |

# INTRODUCTION

**1. INTRODUCTION**

## 1.1 INTRODUCTION

In the dynamic landscape of human-computer interaction, technological advancements continuously strive to bridge the gap between users and machines. One such groundbreaking innovation that has emerged at the forefront is the development of a virtual mouse control system, seamlessly integrating hand gestures and eye movements. This transformative approach not only enhances user experience but also redefines the conventional boundaries of computing interfaces. Traditional input devices have long been dominated by mouse, restricting users to a predefined set of actions. The advent of virtual mouse control through hand gestures liberates users from the constraints of physical peripherals. By harnessing the potential of computer vision and gesture recognition technologies, users can now navigate, click, and interact with digital interfaces using natural hand movements. This intuitive mode of interaction adds a layer of fluidity and immediacy, breaking down barriers between the digital realm and the physical world.

Each recognized hand gesture is mapped to specific mouse functions, such as cursor movement, left-click, right-click, scrolling, and other actions. The system should allow for customization to cater to individual preferences. Moving the hand in space corresponds to the movement of the virtual cursor on the screen. The system translates the spatial information of hand movements into on-screen coordinates, allowing users to point and navigate.

Eye-tracking devices use infrared web camera to track the movement of the eyes. These sensors can be integrated into displays, monitors, or specialized glasses to capture and interpret the user's gaze accurately.­­­­­

A virtual mouse using hand gesture and eye movement recognition is an innovative approach to control a computer without the use of a physical mouse . This technology uses computer vision and python libraries to recognize hand gestures and translate them into mouse movements and clicks. The virtual mouse can be used in a variety of scenarios, including gaming, presentations, and accessibility for individuals with physical disabilities. This technology has the potential to revolutionize the way we interact with computers and make computing more accessible and intuitive for everyone

## 1.2 PROBLEM STATEMENT

* Physically disabled individuals encounter obstacles in computer interaction due to the unavailability of a technology that accommodates their needs.
* Using traditional mouse on various surfaces, causing unpredictable cursor movement

and disrupting the user's interaction.

* Amid contagious outbreaks, the lack of touchless computer interaction methods

poses a problem. Conventional methods involving physical contact can exacerbate disease spread.

## OBJECTIVES

* To create a virtual mouse using web camera to interact with the computer in a more

user friendly manner that can be alternative approach for traditional mouse.

* To develop a cursor control system virtually using hand gestures and eye movements

which performs operations such as left click, right click and cursor movement.

# LITERATURE SURVEY

## 2. LITERATURE SURVEY

### A literature survey or a literature review in a project report shows the various analyses and research made in the field of interest and the results already published, taking into account the various parameters of the project and the extent of the project. Literature survey is mainly carried out in order to analyze the background of the current project which helps to find out flaws in the existing system & guides on which unsolved problems we can work out. So, the following topics not only illustrate the background of the project but also uncover the problems and flaws which motivated to propose solutions and work on this project.

**2.1 EXISTING SYSTEMS**

**2.1.1 Virtual Mouse Control Using Coloured Finger Tips and Hand Gesture Recognition**.

**Authors: Vantukala VishnuTeja Reddy and Thumma Dhyanchand**,

In human-computer interaction, virtual mouse is implemented with fingertip recognition and hand gesture tracking based on image in a live video is one of the studies. The main objective is to find the solution for the finger tracking in the real world and the cursor control of a computer is still performed physically. The proposed system describes virtual mouse control using fingertip identification and hand gesture recognition. This study consists of two methods for tracking the fingers, one is by using coloured caps and other is by hand gesture detection. This includes three main steps that are finger detection using colour identification, hand gesture tracking and implementation on on-screen cursor. Hand gesture tracking is generated through the detection of contour and formation of a convex hull around it.

**2.1.2 Air Canvas Application Using Open CV and Numpy in Python**

**Authors: Prof. S.U. Saoji, Bharati Vidyapeeth**

In the recent years writing in air has been one of the most fascinating and challenging research areas in field of image processing and pattern recognition. It can improve the interface between man and machine in numerous applications. Object tracking is considered as a important task within the field of Computer Vision. The system describes about the use computer vision to trace the path of the finger. The generated text can also be used for various purposes, such as sending messages, emails, etc. The project focuses on developing a motion-to-text converter that can potentially serve as a software for intelligent wearable devices for writing from the air. The project takes an advantage of this gap and focuses on developing a motion-to-text converter that can potentially serve as a software for intelligent wearable devices for writing from the air.

**2.1.3 Real-Time Eye Blink Detection using Facial Landmarks**

**Authors** **: Čech and Soukupová**

A real-time algorithm to detect eye blinks in a video sequence from a standard camera is proposed. Recent landmark detectors, trained on in-the-wild datasets exhibit excellent robustness against face resolution, varying illumination and facial expressions. We show that the landmarks are detected precisely enough to reliably estimate the level of the eye openness. The proposed algorithm therefore estimates the facial landmark positions, extracts a single scalar quantity eye aspect ratio (EAR) – characterizing the eye openness in each frame. Finally, blinks are detected either by an SVM classifier detecting eye blinks as a pattern of EAR values in a short temporal window or by hidden Markov model that estimates the eye states followed by a simple state machine recognizing the blinks according to the eye closure lengths. The proposed algorithm has comparable results with the state-of- the-art methods on three standard datasets

**2.1.4 Hand Gesture Based Virtual Blackboard Using Webcam**

**Authors: Faria Soroni and Sakik al Sajid**

The proposed system is an application that will detect the numbers written by hand gestures. This will work as a virtual board where numbers and letters can be written by using gesture. The purpose of this project is to create a virtual board which can be used to make online confrontation by moving fingers as the hand gesture. The gesture is made by the user who is detected by the machine through the image processing and the operation unique to the machine is carried out, thereby eliminating the requirement of any hardware input device. The input picture from the camera is transformed first into the colour space of HSV, which detects the skin and removes the backdrop. The application is developed using Open CV and Pytorch.

### 2.2 LIMITATIONS OF EXISTING SYSTEMS

* **Limited Flexibility:** Colored finger tips might not capture the full range of natural

hand movements.

* **Ambiguity in Color Perception:** Users with color blindness or difficulties in color

perception might face challenges in accurately distinguishing and identifying colored finger tips.

* Multiple hand or object detection cannot be avoided.
* Identifying and characterizing an object such as a finger from an RGB image without

a depth sensor is a great challenge.

* Using real-time hand gestures to change the system from one state to another requires

a lot of code care.

* Angle between the finger may show effect on movemets.
* There are very limited features.

# SYSTEM ANALYSIS

# 3.SYSTEM ANALYSIS

# A System Analysis is a method for identifying and solving problems that looks at each component in the overall system for the purpose of achieving specific goals. The existing systems are using all algorithms which takes a lot of time to compute and there is no interactive tool for the users.

**3.1 PROPOSED SYSTEM**

This is a system developed which detects hand gestures without any external device in real time. The system accuracy depends the quality of the camera. This system provides a contact free environment to the user , thus eradicating the use of hardware devices. Using In-built python libraries to capture the image frames as input and detects landmarks on hand and performs mouse actions. It does not make use of any external hardware devices, An in-built webcam can be used to control the system.

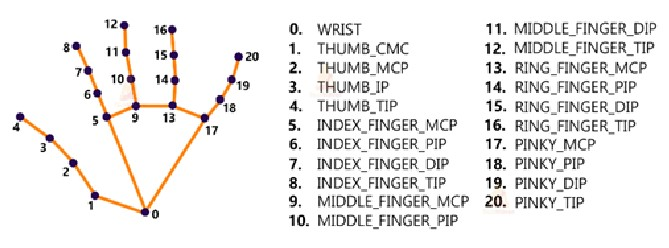


Fig 3.1 Land Marks of Hand

A drawing of a pair of eyes

Description automatically generated

Fig 3.2 Land Marks of Eyes

**3.2 METHODOLOGY OF PROPOSED SYSTEM**

Creating a virtual mouse using hand and eye movements in Python involves several steps. Here's a basic methodology:

**1. Setup:**

* Install necessary libraries such as OpenCV for computer vision and an eye-tracking

library like `dlib`.

* Connect and calibrate the eye-tracking device.

**2. Hand Movement Detection:**

* Use the computer vision library (e.g., OpenCV) to capture video frames from the

camera.

* Apply image processing techniques to identify and track the user's hand.
* Utilize techniques like background subtraction or color-based segmentation to

isolate the hand from the background

* .Implement a hand gesture recognition algorithm to interpret different gestures.

**3. Eye Movement Tracking:**

* Employ the eye-tracking library (e.g., `dlib`) to detect and track the user's eyes.
* Use the eye-tracking data to determine the point of gaze on the screen.

**4. Integration:**

* Combine the hand and eye movement data to control the cursor.
* Map specific hand gestures to cursor actions such as click, drag, or scroll.
* Use the eye-tracking data to enhance cursor precision and enable features like

dwell-based actions.

**5. User Interface:**

* Develop a simple GUI to visualize the virtual mouse movements.
* Display the real-time video feed with the tracked hand and eye movements.

**6. Testing and Calibration:**

* Implement a calibration process to ensure accurate tracking and alignment with the

screen

* Conduct thorough testing to refine the system and address any issues.

**7. Optimization:**

* Optimize the code for performance, considering factors like frame rate and

responsiveness.

* Fine-tune parameters for hand and eye-tracking to enhance accuracy.

**8. User Feedback and Iteration:**

* Gather user feedback to improve the system's usability.
* Iterate on the design and functionality based on user suggestions and testing results.

**9. Documentation:**

* Document the code thoroughly to aid future development and maintenance.

# 3.3 ADVANTAGES OF PROPOSED SYSTEM

* It consumes less or no training time.
* Overcomes the background noise.
* Many number of features.
* This technique will help the paralyzed person, physically challenged people especially person without hands to compute efficiently by using eye moments.

# SYSTEM REQUIREMENTS

**SPECIFICATIONS**

## 4. SYSTEM REQUIREMENTS SPECIFICATIONS

### 4.1 INTRODUCTION

Software Requirements Specification plays an important role in creating quality software solutions. Specification is basically a representation process. Requirements are represented in a manner that ultimately leads to successful software implementation. Requirements may be specified in a variety of ways. However, there are some guidelines worth following:

* Representation format and content should be relevant to the problem.
* Information contained within the specification should be nested
* Diagrams and other notational forms should be restricted in number and consistent
* in use.

**4.2 FUNCTIONAL REQUIREMENTS**

Functional requirements are product features or functions that developers must implement to enable users to accomplish their tasks. So, it’s important to make them clear both for the development team and the stakeholders. Generally, functional requirements describe system behaviour under specific conditions.

**4.3 NON-FUNCTIONAL REQUIREMENTS**

* + **Usability**:

Usability is the case of use and learns ability of a human-made object. The object of use can be a software application, website, book, tool, machine, process, or anything a human interacts with. A usability study may be conducted as a primary job function by a usability analyst or as a secondary job function by designers, technical writers, marketing personnel, and others.

* **Reliability**

The probability that a component part, equipment, or system will satisfactorily perform its intended function under given circumstances, such as environmental conditions, limitations as to operating time, and frequently and thoroughness of maintenance for a specified period of time.

* **Performance**

Accomplishment of a given task measured against present standards of accuracy.

* **Supportability**

To which the design characteristics of a stand by or support system meet the operational requirements of an organization.

* **Implementation**

Implementation is the realization of an application, or execution of a plan, idea, model, design, specification, standard, algorithm, or policy.

* **Interface**

An interface refers to a point of interaction between components and is applicable at the level of both hardware and software. This allows a component whether a piece of hardware such as a graphics card or a piece of software such as an internet browser to function independently while using interfaces to communicate with other components via an input/output system and an associated protocol.

* **Legal**

It is established by or founded upon law or official or accepted rules of or relating to jurisprudence: "legal loop hole". Having legal efficacy or force", "a sound title to the property" Relating to or characteristic of the profession of law, "the legal profession". Allowed by official rules; "a legal pass receiver".

### 4.4 HARDWARE REQUIREMENTS

Hardware interfaces specifies the logical characteristics of each interface between the software product and the hardware components of the system. The following are some hardware requirements.

|  |  |
| --- | --- |
| **System** | Intel I3 and above. |
| **Hard disk** | 120GB and above. |
| **RAM** | 4GB and above |
| **Input Devices** | Webcam |

### 4.5 SOFTWARE REQUIREMENTS

The programming languages, operating system and platform which we have used are as follows:

**4.5.1 PROGRAMMING LANGUAGES USED**

A programming language is a system of notation for writing computer programs Most programming languages are text-based formal languages, but they may also be graphical. They are a kind of computer language.

**4.5.1.1 PYTHON**

Python is a very popular general-purpose interpreted, interactive, object-oriented, and high-level programming language. Python is dynamically-typed and garbage-collected programming language. It was created by Guido van Rossum during 1980-1991.

Python supports multiple programming paradigms, including procedural, Object oriented and functional programming language. Python design philosophy emphasizes code readability with the use of significant indentation.

**4.5.1.2 KEY FEATURES OF PYTHON**

* **Interpreted language**

Python is an interpreted language (an interpreted language is a programming languagethat is generally interpreted, without compiling a program into machine instructions. It is one where the instructions are not directly executed by the target machine, but instead, read and executed by some other program known as the interpreter) and an IDLE (Interactive Development Environment) is packaged along with Python.

* **Dynamically Typed Language**

Python is a dynamically typed language. In other words, in Python, we do not need to declare the data types of the variables which we define. It is the job of the Python interpreter to determine the data types of the variables at runtime based on the types of the parts of the expression.

* **Open Source And Free**

Python is an open-source programming language and one can download it for free from Python’s official website

* **Large Standard Library**

One of the very important features because of which Python is so famous in today’s times is the huge standard library it offers to its users. The standard library of Python is extremely large with a diverse set of packages and modules like itertools, functools, operator, and many more with common and important functionalities in them. If the code of some functionality is already present in these modules and packages, the developers do not need to rewrite them from scratch.

* **Object Oriented Programming Language**

Python supports various programming paradigms like structured programming, functional programming, and object-oriented programming. However, the most important fact is that the Object-Oriented approach of Python allows its users to implement the concepts of Encapsulation, Inheritance, Polymorphism, etc. which is extremely important for the coding done in most Software Industries as objects map to entities in the real world and a lot of real-world problems can be solved using the Object-Oriented Approach.

* **Platform Independent**

Platform independence is yet another amazing feature of Python. In other words, it means that if we write a program in Python, it can run on a variety of platforms, for instance, Windows, Mac, Linux, etc.

**4.5.2 VISUAL STUDIO CODE**

Visual Studio Code (VS Code) is a versatile, free source-code editor by Microsoft. It supports multiple languages, offering smart code completion, debugging, and a vast extension library for tailored experiences. Its intuitive interface includes a sidebar for easy navigation and an integrated terminal. VS Code excels in debugging, featuring breakpoints and variable inspection. With Git integration and performance efficiency, it manages projects seamlessly. Its open-source nature fosters a thriving community, contributing to its continual enhancement. Offering a balance of robust features, ease of use, and adaptability, VS Code stands out as a top choice for developers across various platforms and languages**.**

**4.5.3 LIBRARIES USED**

**OpenCV**

OpenCV library in python is a computer vision library that is widely used for image analysis, image processing, detection, recognition, etc. OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. Being a BSD-licensed product, OpenCV makes it easy for businesses to utilize and modify the code.

The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high-resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, etc.

**MediaPipe**

Mediapipe is a cross-platform library developed by Google that provides amazing ready-touse ML solutions for computer vision tasks. MediaPipe Hands is a high-fidelity hand and finger tracking solution. It employs machine learning (ML) to infer 21 3D landmarks of a hand from just a single frame. Whereas current state-of-the-art approaches rely primarily on powerful desktop environments for inference, our method achieves real-time performance on a mobile phone, and even scales to multiple hands.

MediaPipe Hands utilizes an ML pipeline consisting of multiple models working together: A hand landmark model that operates on the cropped image region defined by the palm detector and returns high-fidelity 3D hand keypoints. This strategy is similar to that employed in our MediaPipe Face Mesh solution, which uses a face detector together with a face landmark model.

**PyAutoGUI**

PyAutoGUI is a Python library used for automating mouse movements, keyboard inputs, and screen interactions. It facilitates tasks like GUI automation, game playing, and automated testing. PyAutoGUI lets users control the mouse and keyboard, take screenshots, find images on the screen, and perform various actions programmatically. It's platform-independent and offers simple commands for complex interactions, making it accessible for tasks requiring repetitive actions or UI interactions. Despite its simplicity, PyAutoGUI empowers users to create sophisticated automation scripts, enhancing productivity across different applications and scenarios.

**Streamlit**

Streamlit is a Python library designed for building web applications with ease. It simplifies the creation of interactive, data-centric web apps by enabling developers to convert simple scripts into shareable web apps. With minimal code, Streamlit allows for the creation of intuitive and visually appealing interfaces for data visualization, machine learning models, and more. Its reactive framework automatically updates the interface when inputs change, providing a seamless user experience. Streamlit's simplicity and focus on rapid prototyping make it popular among data scientists and developers, enabling them to deploy and share their data-driven applications effortlessly.

# 

# SYSTEM DESIGN

## 5. SYSTEM DESIGN

### 5.1 INTRODUCTION

System design is the process of defining the elements of a system such as the architecture, modules, and components, the different interfaces of those components, and the data that goes through that system. It is meant to satisfy specific needs and requirements through the engineering of a coherent and well-running System. The most creative and challenging phase of the life cycle is System design. The term design describes a final system and the process by which it is developed. It refers to the technical specifications that will be applied in the implementation of the candidate system.

The system design may be defined as “The process of applying various techniques and principles defining a device, a process or a system with sufficient details to permit its physical realization”. The importance of software design can be stated in a single word “Quality”. The design provides us with representations of software that can be assessed for quality. Design is the only way where we can accurately translate a customer’s requirements into a complete software product or system. Without design, we risk building an unstable system that might fail if small changes are made. It may as well be difficult to test or could be one whose quality can’t be tested. So, it is an essential phase in the development of a software product.

**5.2 UML DIAGRAMS**

UML is a way of visualizing a software program using a collection of diagrams. The notation has evolved from the work of the Rational Software Corporation to be used for object oriented design, but it has since been extended to cover a wider variety of software engineering projects. The Object Management Group as the standard for modeling software development accepts UML.

UML stands for Unified Modelling Language. UML 2.0 helped extend the original UML specification to cover a wider portion of software development efforts including agile practices.

* Improved integration between structural models like class diagrams and behavior models like activity diagrams.
* Added to the ability to define a hierarchy and decompose a software system into components and subcomponents.
* The original UML specifies nine diagrams; UML 2. x brings that number up to 13. The four new diagrams are called: Communication diagram, a composite structure diagram, an interaction overview diagram, and a timing diagram.

**5.2.1 USE CASE DIAGRAM**

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. A use case diagram at its simplest is a representation of a user's interaction with the system that shows the relationship between the user and the different use cases in which the user is involved. A use case diagram can identify the different types of users of a system and the different use cases and will often be accompanied by other types of diagrams as well. The use cases are represented by either circles or ellipses.

**A diagram of a hand movement

Description automatically generated**

Fig 5.1 Use Case Diagram

**5.2. 2 SYSTEM ARCHITECTURE**

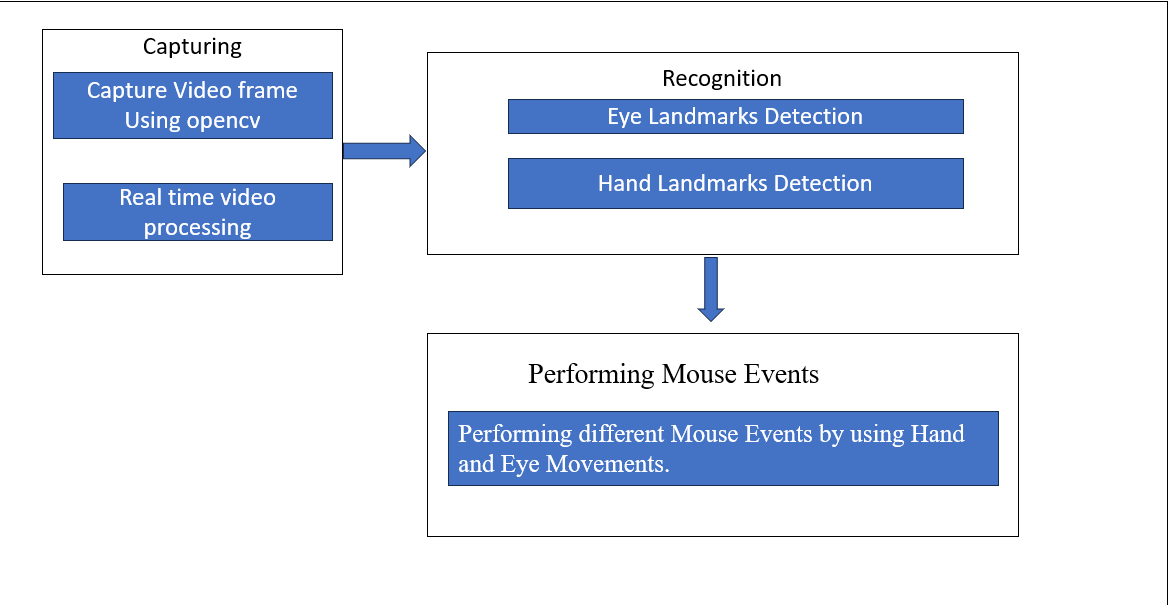
****

Fig 5.2 System Architecture

# 5.2.3 ACTIVITY DIAGRAM

Activity diagram is sometimes considered as the flowchart. Although the diagrams look like a flowchart, they are not. It shows different flows such as parallel, branched, concurrent, and single. Activity is a particular operation of the system. The below image represents the workflow or activity diagram of the Cursor and keyboard activity diagram.

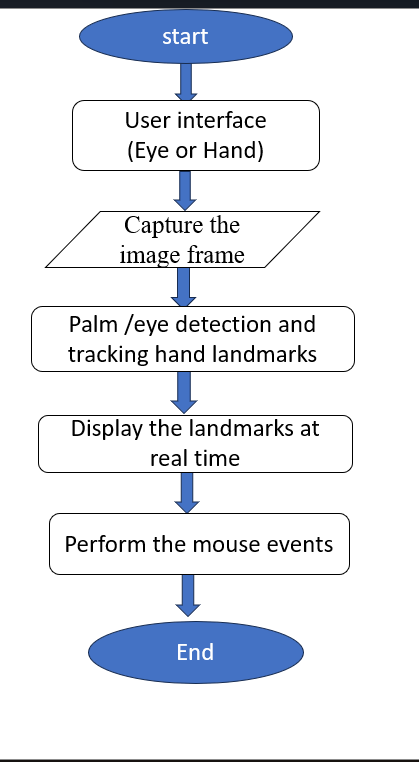


Fig 5.3 Activity Diagram

**5.2.4 CLASS DAIGRAM**

Class diagrams are used to model the static view of a system, focusing on the entities (classes) within that system and how they interact with each other. Classes are depicted as boxes with three sections: the top section contains the class name, the middle section includes the attributes (variables) of the class, and the bottom section displays the methods (functions or operations) that the class can perform.



# IMPLEMENTATION

# IMPLEMENTATION

**6.1 SAMPLE CODE**

Virtualmouse.py

import streamlit as st

import cv2

import mediapipe as mp

import pyautogui

import math

from enum import IntEnum

from ctypes import cast, POINTER

from comtypes import CLSCTX\_ALL

from pycaw.pycaw import AudioUtilities, IAudioEndpointVolume

from google.protobuf.json\_format import MessageToDict

import screen\_brightness\_control as sbcontrol

st.title("VIRTUAL MOUSE USING HAND GESTURES AND EYE MOVEMENTS")

st.title("Choose an Operation")

running=False

if st.button("USING EYE MOVEMENTS"):

st.write("PRESS ENTER TO EXIT WINDOW")

import cv2

import mediapipe as mp

import pyautogui

cam = cv2.VideoCapture(0)

face\_mesh = mp.solutions.face\_mesh.FaceMesh(refine\_landmarks=True)

screen\_w, screen\_h = pyautogui.size()

while True:

\_, frame = cam.read()

frame = cv2.flip(frame, 1)

rgb\_frame = cv2.cvtColor(frame, cv2.COLOR\_BGR2RGB)

output = face\_mesh.process(rgb\_frame)

landmark\_points = output.multi\_face\_landmarks

frame\_h, frame\_w, \_ = frame.shape

if landmark\_points:

landmarks = landmark\_points[0].landmark

for id, landmark in enumerate(landmarks[474:478]):

x = int(landmark.x \* frame\_w)

y = int(landmark.y \* frame\_h)

cv2.circle(frame, (x, y), 3, (0, 255, 0))

if id == 1:

screen\_x = screen\_w \* landmark.x

screen\_y = screen\_h \* landmark.y

pyautogui.moveTo(screen\_x, screen\_y)

left = [landmarks[145], landmarks[159]]

for landmark in left:

x = int(landmark.x \* frame\_w)

y = int(landmark.y \* frame\_h)

cv2.circle(frame, (x, y), 3, (0, 255, 255))

if (left[0].y - left[1].y) < 0.01:

pyautogui.click()

pyautogui.sleep(1)

cv2.imshow('Eye Controlled Mouse', frame)

if cv2.waitKey(5) & 0xFF == 13:

break

if st.button("USING HAND GESTURES"):

st.write("PRESS ENTER TO EXIT WINDOW")

running=True

pyautogui.FAILSAFE = False

mp\_drawing = mp.solutions.drawing\_utils

mp\_hands = mp.solutions.hands

# Gesture Encodings

class Gest(IntEnum):

# Binary Encoded

FIST = 0

PINKY = 1

RING = 2

MID = 4

LAST3 = 7

INDEX = 8

FIRST2 = 12

LAST4 = 15

THUMB = 16

PALM = 31

# Extra Mappings

V\_GEST = 33

TWO\_FINGER\_CLOSED = 34

PINCH\_MAJOR = 35

PINCH\_MINOR = 36

# Multi-handedness Labels

class HLabel(IntEnum):

MINOR = 0

MAJOR = 1

# Convert Mediapipe Landmarks to recognizable Gestures

class HandRecog:

def \_\_init\_\_(self, hand\_label):

self.finger = 0

self.ori\_gesture = Gest.PALM

self.prev\_gesture = Gest.PALM

self.frame\_count = 0

self.hand\_result = None

self.hand\_label = hand\_label

def update\_hand\_result(self, hand\_result):

self.hand\_result = hand\_result

def get\_signed\_dist(self, point):

sign = -1

if self.hand\_result.landmark[point[0]].y < self.hand\_result.landmark[point[1]].y:

sign = 1

dist = (self.hand\_result.landmark[point[0]].x - self.hand\_result.landmark[point[1]].x)\*\*2

dist += (self.hand\_result.landmark[point[0]].y - self.hand\_result.landmark[point[1]].y)\*\*2

dist = math.sqrt(dist)

return dist\*sign

def get\_dist(self, point):

dist = (self.hand\_result.landmark[point[0]].x - self.hand\_result.landmark[point[1]].x)\*\*2

dist += (self.hand\_result.landmark[point[0]].y - self.hand\_result.landmark[point[1]].y)\*\*2

dist = math.sqrt(dist)

return dist

def get\_dz(self,point):

return abs(self.hand\_result.landmark[point[0]].z -self.hand\_result.landmark[point[1]].z)

def set\_finger\_state(self):

if self.hand\_result == None:

return

points = [[8,5,0],[12,9,0],[16,13,0],[20,17,0]]

self.finger = 0

self.finger = self.finger | 0 #thumb

for idx,point in enumerate(points):

dist = self.get\_signed\_dist(point[:2])

dist2 = self.get\_signed\_dist(point[1:])

try:

ratio = round(dist/dist2,1)

except:

ratio = round(dist/0.01,1)

self.finger = self.finger << 1

if ratio > 0.5 :

self.finger = self.finger | 1

# Handling Fluctations due to noise

def get\_gesture(self):

if self.hand\_result == None:

return Gest.PALM

current\_gesture = Gest.PALM

if self.finger in [Gest.LAST3,Gest.LAST4] and self.get\_dist([8,4]) < 0.05:

if self.hand\_label == HLabel.MINOR :

current\_gesture = Gest.PINCH\_MINOR

else:

current\_gesture = Gest.PINCH\_MAJOR

elif Gest.FIRST2 == self.finger :

point = [[8,12],[5,9]]

dist1 = self.get\_dist(point[0])

dist2 = self.get\_dist(point[1])

ratio = dist1/dist2

if ratio > 1.7:

current\_gesture = Gest.V\_GEST

else:

if self.get\_dz([8,12]) < 0.1:

current\_gesture = Gest.TWO\_FINGER\_CLOSED

else:

current\_gesture = Gest.MID

else:

current\_gesture = self.finger

if current\_gesture == self.prev\_gesture:

self.frame\_count += 1

else:

self.frame\_count = 0

self.prev\_gesture = current\_gesture

if self.frame\_count > 4 :

self.ori\_gesture = current\_gesture

return self.ori\_gesture

# Executes commands according to detected gestures

class Controller:

tx\_old = 0

ty\_old = 0

trial = True

flag = False

grabflag = False

pinchmajorflag = False

pinchminorflag = False

pinchstartxcoord = None

pinchstartycoord = None

pinchdirectionflag = None

prevpinchlv = 0

pinchlv = 0

framecount = 0

prev\_hand = None

pinch\_threshold = 0.3

def getpinchylv(hand\_result):

dist = round((Controller.pinchstartycoord - hand\_result.landmark[8].y)\*10,1)

return dist

def getpinchxlv(hand\_result):

dist = round((hand\_result.landmark[8].x - Controller.pinchstartxcoord)\*10,1)

return dist

def changesystembrightness():

brightness\_value = sbcontrol.get\_brightness(display=0)

if isinstance(brightness\_value, list) and len(brightness\_value) > 0:

currentBrightnessLv = brightness\_value[0] / 100.0

currentBrightnessLv += Controller.pinchlv / 50.0

currentBrightnessLv = max(0.0, min(1.0, currentBrightnessLv)) # Clamp between 0.0 and 1.0

sbcontrol.fade\_brightness(int(100 \* currentBrightnessLv), start=brightness\_value[0])

else:

pass

def changesystemvolume():

devices = AudioUtilities.GetSpeakers()

interface = devices.Activate(IAudioEndpointVolume.\_iid\_, CLSCTX\_ALL, None)

volume = cast(interface, POINTER(IAudioEndpointVolume))

currentVolumeLv = volume.GetMasterVolumeLevelScalar()

currentVolumeLv += Controller.pinchlv/50.0

if currentVolumeLv > 1.0:

currentVolumeLv = 1.0

elif currentVolumeLv < 0.0:

currentVolumeLv = 0.0

volume.SetMasterVolumeLevelScalar(currentVolumeLv, None)

def scrollVertical():

pyautogui.scroll(120 if Controller.pinchlv>0.0 else -120)

def scrollHorizontal():

pyautogui.keyDown('shift')

pyautogui.keyDown('ctrl')

pyautogui.scroll(-120 if Controller.pinchlv>0.0 else 120)

pyautogui.keyUp('ctrl')

pyautogui.keyUp('shift')

def get\_position(hand\_result):

point = 9

position = [hand\_result.landmark[point].x ,hand\_result.landmark[point].y]

sx,sy = pyautogui.size()

x\_old,y\_old = pyautogui.position()

x = int(position[0]\*sx)

y = int(position[1]\*sy)

if Controller.prev\_hand is None:

Controller.prev\_hand = x,y

delta\_x = x - Controller.prev\_hand[0]

delta\_y = y - Controller.prev\_hand[1]

distsq = delta\_x\*\*2 + delta\_y\*\*2

ratio = 1

Controller.prev\_hand = [x,y]

if distsq <= 25:

ratio = 0

elif distsq <= 900:

ratio = 0.07 \* (distsq \*\* (1/2))

else:

ratio = 2.1

x , y = x\_old + delta\_x\*ratio , y\_old + delta\_y\*ratio

return (x,y)

def pinch\_control\_init(hand\_result):

Controller.pinchstartxcoord = hand\_result.landmark[8].x

Controller.pinchstartycoord = hand\_result.landmark[8].y

Controller.pinchlv = 0

Controller.prevpinchlv = 0

Controller.framecount = 0

# Hold final position for 5 frames to change status

def pinch\_control(hand\_result, controlHorizontal, controlVertical):

if Controller.framecount == 5:

Controller.framecount = 0

Controller.pinchlv = Controller.prevpinchlv

if Controller.pinchdirectionflag == True:

controlHorizontal() #x

elif Controller.pinchdirectionflag == False:

controlVertical() #y

lvx = Controller.getpinchxlv(hand\_result)

lvy = Controller.getpinchylv(hand\_result)

if abs(lvy) > abs(lvx) and abs(lvy) > Controller.pinch\_threshold:

Controller.pinchdirectionflag = False

if abs(Controller.prevpinchlv - lvy) < Controller.pinch\_threshold:

Controller.framecount += 1

else:

Controller.prevpinchlv = lvy

Controller.framecount = 0

elif abs(lvx) > Controller.pinch\_threshold:

Controller.pinchdirectionflag = True

if abs(Controller.prevpinchlv - lvx) < Controller.pinch\_threshold:

Controller.framecount += 1

else:

Controller.prevpinchlv = lvx

Controller.framecount = 0

def handle\_controls(gesture, hand\_result):

x,y = None,None

if gesture != Gest.PALM :

x,y = Controller.get\_position(hand\_result)

# flag reset

if gesture != Gest.FIST and Controller.grabflag:

Controller.grabflag = False

pyautogui.mouseUp(button = "left")

if gesture != Gest.PINCH\_MAJOR and Controller.pinchmajorflag:

Controller.pinchmajorflag = False

if gesture != Gest.PINCH\_MINOR and Controller.pinchminorflag:

Controller.pinchminorflag = False

# implementation

if gesture == Gest.V\_GEST:

Controller.flag = True

pyautogui.moveTo(x, y, duration = 0.1)

elif gesture == Gest.FIST:

if not Controller.grabflag :

Controller.grabflag = True

pyautogui.mouseDown(button = "left")

pyautogui.moveTo(x, y, duration = 0.1)

elif gesture == Gest.MID and Controller.flag:

pyautogui.click()

Controller.flag = False

elif gesture == Gest.INDEX and Controller.flag:

pyautogui.click(button='right')

Controller.flag = False

elif gesture == Gest.TWO\_FINGER\_CLOSED and Controller.flag:

pyautogui.doubleClick()

Controller.flag = False

elif gesture == Gest.PINCH\_MINOR:

if Controller.pinchminorflag == False:

Controller.pinch\_control\_init(hand\_result)

Controller.pinchminorflag = True

Controller.pinch\_control(hand\_result,Controller.scrollHorizontal, Controller.scrollVertical)

elif gesture == Gest.PINCH\_MAJOR:

if Controller.pinchmajorflag == False:

Controller.pinch\_control\_init(hand\_result)

Controller.pinchmajorflag = True

Controller.pinch\_control(hand\_result,Controller.changesystembrightness, Controller.changesystemvolume)

class GestureController:

gc\_mode = 0

cap = None

CAM\_HEIGHT = None

CAM\_WIDTH = None

hr\_major = None # Right Hand by default

hr\_minor = None # Left hand by default

dom\_hand = True

def \_\_init\_\_(self):

GestureController.gc\_mode = 1

GestureController.cap = cv2.VideoCapture(0)

GestureController.CAM\_HEIGHT = GestureController.cap.get(cv2.CAP\_PROP\_FRAME\_HEIGHT)

GestureController.CAM\_WIDTH = GestureController.cap.get(cv2.CAP\_PROP\_FRAME\_WIDTH)

def classify\_hands(results):

left , right = None,None

try:

handedness\_dict = MessageToDict(results.multi\_handedness[0])

if handedness\_dict['classification'][0]['label'] == 'Right':

right = results.multi\_hand\_landmarks[0]

else :

left = results.multi\_hand\_landmarks[0]

except:

pass

try:

handedness\_dict = MessageToDict(results.multi\_handedness[1])

if handedness\_dict['classification'][0]['label'] == 'Right':

right = results.multi\_hand\_landmarks[1]

else :

left = results.multi\_hand\_landmarks[1]

except:

pass

if GestureController.dom\_hand == True:

GestureController.hr\_major = right

GestureController.hr\_minor = left

else :

GestureController.hr\_major = left

GestureController.hr\_minor = right

def start(self):

handmajor = HandRecog(HLabel.MAJOR)

handminor = HandRecog(HLabel.MINOR)

with mp\_hands.Hands(max\_num\_hands = 2,min\_detection\_confidence=0.5, min\_tracking\_confidence=0.5) as hands:

while GestureController.cap.isOpened() and GestureController.gc\_mode:

success, image = GestureController.cap.read()

if not success:

print("Ignoring empty camera frame.")

continue

image = cv2.cvtColor(cv2.flip(image, 1), cv2.COLOR\_BGR2RGB)

image.flags.writeable = False

results = hands.process(image)

image.flags.writeable = True

image = cv2.cvtColor(image, cv2.COLOR\_RGB2BGR)

if results.multi\_hand\_landmarks:

GestureController.classify\_hands(results)

handmajor.update\_hand\_result(GestureController.hr\_major)

handminor.update\_hand\_result(GestureController.hr\_minor)

handmajor.set\_finger\_state()

handminor.set\_finger\_state()

gest\_name = handminor.get\_gesture()

if gest\_name == Gest.PINCH\_MINOR:

Controller.handle\_controls(gest\_name, handminor.hand\_result)

else:

gest\_name = handmajor.get\_gesture()

Controller.handle\_controls(gest\_name, handmajor.hand\_result)

for hand\_landmarks in results.multi\_hand\_landmarks:

mp\_drawing.draw\_landmarks(image, hand\_landmarks, mp\_hands.HAND\_CONNECTIONS)

else:

Controller.prev\_hand = None

cv2.imshow('Gesture Controller', image)

if cv2.waitKey(5) & 0xFF == 13:

break

GestureController.cap.release()

cv2.destroyAllWindows()

gc1 = GestureController()

gc1.start()

# TESTING

## 7. TESTING

### 7.1 INTRODUCTION

Software Testing is a method to check whether the actual software product matches expected requirements and to ensure that the software product is Defect free. It involves the execution of software/system components using manual or automated tools to evaluate one or more properties of interest. The purpose of software testing is to identify errors, gaps, or missing requirements in contrast to actual requirements.. Properly tested software product ensures reliability, security, and high performance which further results in time-saving, cost-effectiveness, and customer satisfaction. Testing is important because software bugs could be expensive or even dangerous. Software bugs can potentially cause monetary and human loss, and history is full of such examples. i)Starbucks was forced to close about 60 percent of stores in the U.S. and Canada due to software failure in its POS system. At one point, the store served coffee for free as they were unable to process the transaction. ii) Some of Amazon’s third-party retailers saw their product price reduced to 1p due to a software glitch. They were left with heavy losses. iii) Vulnerability in Windows 10. This bug enables users to escape from security sandboxes through a flaw in the win32k system. iv) In April of 1999, a software bug caused the failure of a $1.2 billion military satellite launch, the costliest accident in history

**7.2 TYPES OF SOFTWARE TESTING**

Here are the software testing types:

Typically Testing is classified into two categories.

1. Functional Testing
2. Non-Functional Testing or Performance Testing.
   * 1. **FUNCTIONAL TESTING**

Functional testing deals with :

* **Unit Testing**: This software testing basic approach is followed by the programmer to test the unit of the program. It helps developers to know whether the individual unit of the code is working properly or not.
  + **Integration testing**: It focuses on the construction and design of the software. You

need to see whether the integrated units are working without errors or not.

* + **System testing**: In this method, your software is compiled as a whole and then tested

as a whole. This testing strategy checks functionality, security, and portability, among others.

**7.2.2 NON-FUNCTIONAL TESTING**

Non-functional Testing deals with

* + **Performance**: Performance refers to the capability of a system to provide a certain

response time. server a defined number of users or process a certain amount of data.

* + **Endurance**: Endurance Testing is a type of Software Testing that is performed to

observe whether an application can resist the processing load it is expected to have to endure for a long period. During endurance testing, memory consumption is considered to determine potential failures.

* + **Scalability**: Scalability refers to the characteristic of a system to increase

performance by adding additional resources.

**7.3 TESTING ON OUR SYSTEM:**

Every project involves testing and coding part. These two are important for completion of a project. In our project ,it should detect a person having diabetes or not.In this we will give the input to the code itself and to get desired output it involves different kinds of testing in our system. Integration testing will be there for checking the software requirements of our system, it sees whether our system has a capacity to give the desired output. It will test for RAM,OS, libraries, required process the code, checking runtime errors and compile time errors. After completion of the testing, if it has no errors in the entire code, it will generate the desired output

# RESULTS

**8.RESULT**

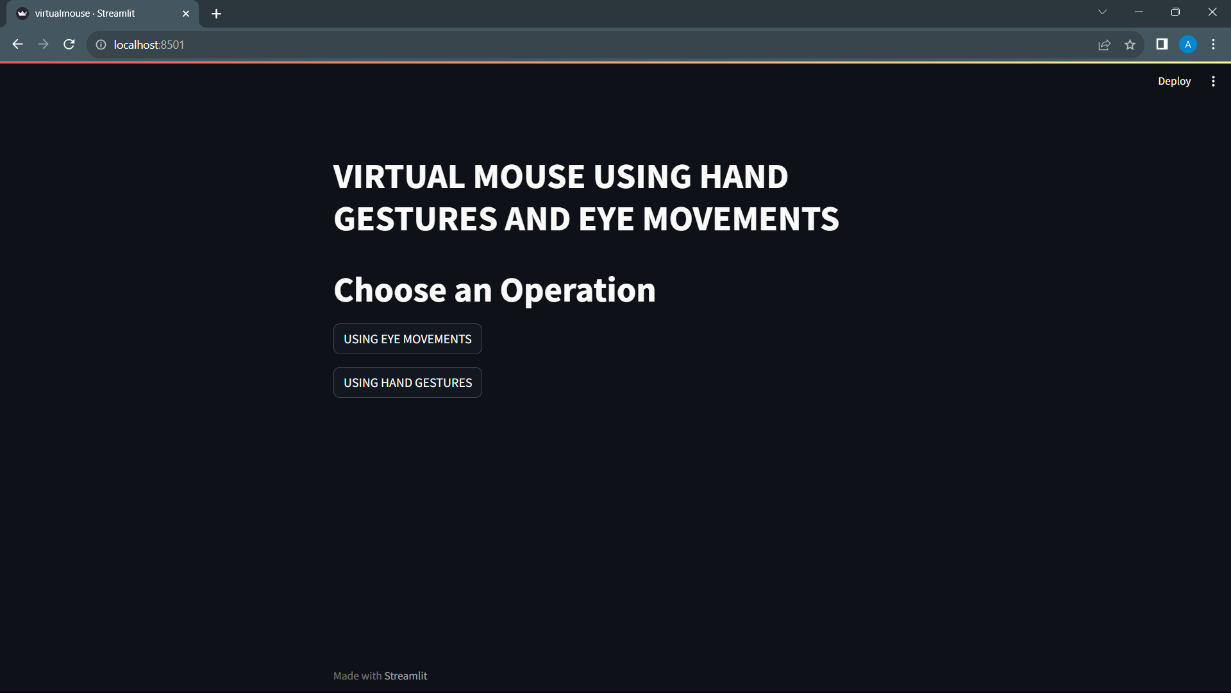


Fig 8.1 Home Page of our application

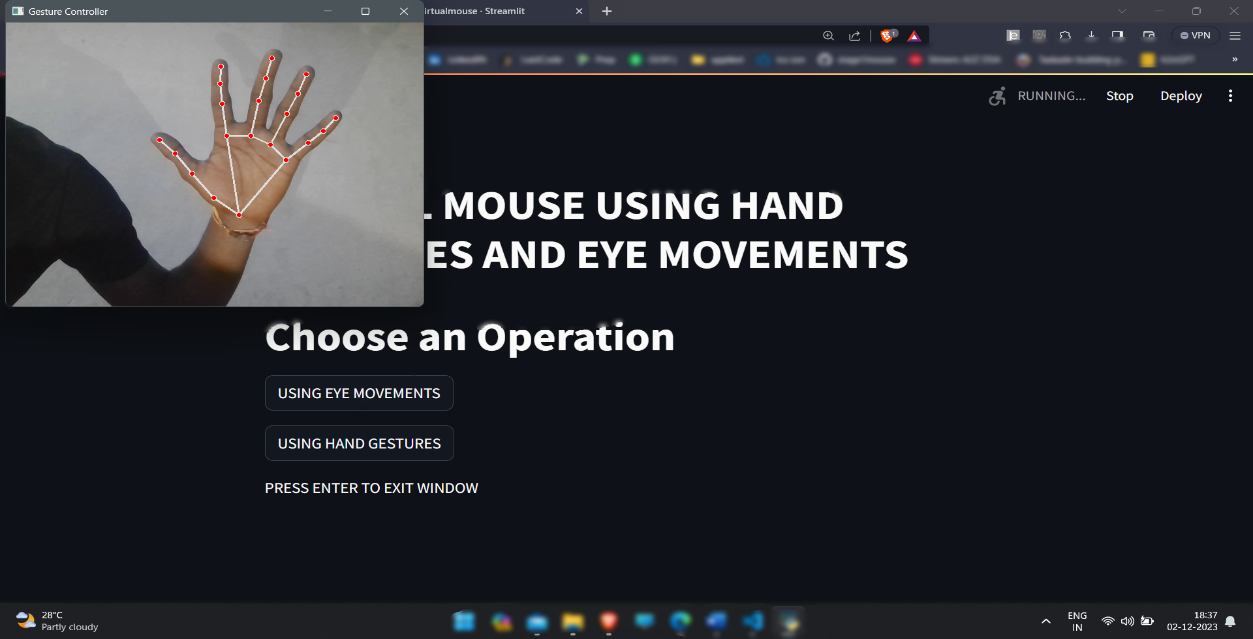
****

Fig 8.2 Landmarks of Hand

**A screenshot of a computer

Description automatically generated**

Fig 8.3 Gesture to move cursor

**A screenshot of a computer

Description automatically generated**

Fig 8.4 Gesture to Right Click

**A screenshot of a computer

Description automatically generated**

Fig 8.5 Gesture to Left Click

**A screenshot of a computer

Description automatically generated**

Fig 8.6 Gesture to control Volume and Brightness

**A screenshot of a computer

Description automatically generated**

Fig 8.7 Gesture to Scroll

**A close up of a person's eye

Description automatically generated**

Fig 8.8 Land Marks of Eyes

# CONCLUSION

## 9. CONCLUSION

Virtual mouse technology marks a pivotal leap in Human-Computer Interaction (HCI), virtually controlled system focusing on cursor operations and interactions using hand gestures and eye movements. This system is developed in such a way that a simple Webcam is sufficient enough to identify hand gestures and perform operations.

The system also acts as a user-friendly device for physically challenged and aged people by providing them with an interface for communication. This innovative approach significantly reduces hardware dependency, eliminating the need for a mouse

# BIBLIOGRAPHY

## REFERENCES

[1]. F. Soroni, S. a. Sajid, M. N. H. Bhuiyan, J. Iqbal and M. M. Khan, "Hand Gesture Based Virtual Blackboard Using Webcam," 2021 IEEE 12th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON), 2021, pp. 0134-0140, doi: 10.1109/IEMCON53756.2021.9623181.

[2]. Prof. S.U. Saoji, Nishtha Dua, Akash Kumar Choudhary, Bharat Phogat “Air Canvas application using OpenCV and NumPy in python”International Research Journal of Engineering and Technology, Volume: 08, Issue 08 Aug 2021

[3]. K. V. V. Reddy, T. Dhyanchand, G. V. Krishna and S. Maheshwaram, "Virtual Mouse Control Using Colored Finger Tips and Hand Gesture Recognition ", 2020, doi:10.1109/HYDCON48903.2020.9242677.

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

[5]. C. D. Sai Nikhil, C. U. Someswara Rao, E. Brumancia, K. Indira, T. Anandhi and P. Ajitha, "Finger Recognition and Gesture based Virtual Keyboard," 2020 5th International Conference on Communication and Electronics Systems (ICCES), 2020, pp. 1321-1324, doi: 10.1109/ICCES48766.2020.9137889.

[6] Ioana Bacivarov, Mircea Ionita, Peter Corcoran, “Statistical models of appearance for eye tracking and eye blink detection and measurement”, IEEE transactions on consumer electronics, Vol.54 , No.3, pp. 1312‐1320 August2009.

[7] Mohamed Nasor, K KMujeeb Rahman, Maryam Mohamed Zubair, Haya Ansari, Farida Mohamed. "Eye- controlled mouse cursor for physically disabled individual" , 2018 Advances in Science and Engineering Technology International Conferences (ASET), 2018.

[8]. S. R. Chowdhury, S. Pathak and M. D. A. Praveena, "Gesture Recognition Based Virtual Mouse and Keyboard," 2020 4th International Conference on Trends in Electronics and Informatics (ICOEI)(48184), 2020, doi: 10.1109/ICOEI48184.2020.9143016.