# Hand Gesture Virtual Air Keyboard

Submitted Mini Project work in partial fulfillment of the requirements

For the degree of

**Under Graduate of Engineering**

**(Computer Enginnering)**

by

**Vaibhav Chavan (22CE1261)**

**Purvesh Desai (22CE1187)**

**Pratthamesh Baviskar(22CE1225)**

**Aditya Andhale (22CE1255)**

Supervisor

**Ms. Smitha Raveendran**


Department of Information Technology
Ramrao Adik Institute of Technology,
Sector 7, Nerul , Navi Mumbai
(Affiliated to D.Y.Patil Deemed to be University) October 2023

**Ramrao Adik Institute of Technology**

(Affiliated to D.Y.Patil Deemed to be University) Dr. D. Y. Patil Vidyanagar,Sector 7, Nerul, Navi Mumbai 400706.

# Certificate

This is to certify that, the Mini-Project titled

## "AI Hand Gesture Virtual Air Keyboard"

is a bonafide work done by

**Vaibhav Chavan (22CE1261)**

**Purvesh Desai (22CE1187)**

**Pratthamesh Baviskar(22CE1225)**

**Aditya Andhale (22CE1255)**

and is submitted in the partial fulfillment of the requirement for the degree of

**Bachelor of Engineering**

**( Information Technology)** to the

**D.Y.Patil Deemed to be University**.

Supervisor (**Ms. Smitha Raveendran**)

| Mini-Project Coordinator | Head of Department | Principal |
|---|---|---|
| (Dr.Gautam Borkar) | (Dr.Sangita Chaudhari) | (Dr.Mukesh D.Patil) |

# Mini Project Approval for S. E.

This is to certify that the Mini project entitled "**AI Hand Gesture Virtual Air Keyboard**" is a bonafide work done by **Vaibhav Chavan , Purvesh Desai ,Pratthamesh Baviskar and Aditya Andhale** under the supervision of **Ms. Smitha Raveendran** This report has been approved.

**Examiners:**

1: _____

2: _____

**Supervisors:**

1: _____

2: _____

**Principal:**

_____

Date: _____

Place: _____

# Declaration

We declare that this written submission represents my ideas in my own words and where others' ideas or words have been included, We have adequately cited and referenced the original sources. We also declare that We have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

_____

**(Signature)**

_____

**Vaibhav Chavan (22CE1261)**

_____

**Purvesh Desai (22CE1187)**

_____

**Pratthamesh Baviskar(22CE1225)**

_____

**Aditya Andhale (22CE1255)**

Date: _____

# ABSTRACT

Traditional keyboard input methods, such as typing or swiping, can be cumbersome and time-consuming. A hand gesture keyboard is a novel input method that uses computer vision to recognize hand gestures and convert them to text. This can be a more efficient and natural way to interact with computers, especially for people with disabilities or who are in situations where traditional input methods are difficult to use.This project proposes a hand gesture keyboard system that uses a webcam to track the user's hand and recognize gestures. The system uses a machine learning model to classify the hand gestures and map them to corresponding characters. The output text is then displayed on the screen. The system is designed to be user-friendly and efficient. It can be used to type text, enter commands, and control applications. The system is also customizable, so users can create their own gestures and map them to any desired characters or functions.The potential applications of a hand gesture keyboard are wide-ranging. It can be used in a variety of settings, such as education, healthcare, entertainment, and gaming. It can also be used by people with disabilities to improve their communication and independence.

Here are some of the specific benefits of using a hand gesture keyboard:

- Increased efficiency: Hand gesture typing can be faster and more efficient than traditional typing methods, especially for repetitive tasks.
- Reduced fatigue: Hand gesture typing can reduce fatigue associated with traditional typing methods, which can be beneficial for people with long-term computer use.
- Improved accessibility: Hand gesture keyboards can make computers more accessible to people with disabilities, such as those with limited mobility or vision impairments.
- Increased engagement: Hand gesture keyboards can make computer interactions more engaging and immersive, which can be beneficial for education and entertainment applications.

Overall, a hand gesture keyboard is a promising new input method with the potential to revolutionize the way we interact with computers.

# CONTENTS

# Chapter 1

# *Introduction*

In today's digital world, we are constantly interacting with screens and devices. One of the most common ways we interact with these devices is through typing. However, typing on a physical keyboard can be difficult and uncomfortable, especially for people with disabilities. Virtual keyboards offer a more accessible and inclusive way to type. They can be used on a variety of devices, including computers, tablets, and smartphones. Virtual keyboards can also be customized to meet the needs of individual users. One of the most promising ways to create virtual keyboards is to use computer vision. Computer vision can be used to track the movements of the user's hand and fingers, and to identify the keys that the user is trying to press. This type of virtual keyboard is known as a hand-tracked keyboard. Hand-tracked keyboards have several advantages over traditional virtual keyboards. First, they are more accessible to people with disabilities. Second, they are more accurate and efficient, as the user does not have to worry about accidentally pressing the wrong key. Third, they can be used on a variety of devices, without the need for any special hardware.

## 1.1 Motivation

We was motivated to work on this project by my own experiences with hand-tracked keyboards. We have a friend who has cerebral palsy, and he finds it very difficult to use a traditional keyboard. He told me that a hand-tracked keyboard would be a life-changer for him. We also believe that hand-tracked keyboards have the potential to revolutionize the way we interact with computers. With hand-tracked keyboards, we could type on any surface, without the need for a physical keyboard. This would make computing more accessible and inclusive for everyone.
Practical Application: Developing a virtual keyboard offers a practical and real-world application of software development skills. It's a project that can be used on a daily basis, making it relevant and valuable.

Accessibility and Inclusivity: By creating an accessible virtual keyboard, you can make a positive impact on the lives of individuals with disabilities who rely on alternative input methods for communication.

Learning Opportunity: Building a virtual keyboard involves a variety of technologies, including UI/UX design, machine learning, and mobile development, providing an opportunity to learn and master new skills.

Innovation and Customization: You have the freedom to innovate and add unique features to your virtual keyboard, making it stand out in the market. Personalization options can cater to specific user preferences.

Global User Base:

Virtual keyboards have a vast user base worldwide. Developing one can potentially reach a global audience and make a meaningful contribution to the tech community.

Problem Solving:

Virtual keyboards often have limitations that can be addressed through your project. This provides the opportunity to solve real-world problems and improve user experiences.

Portfolio Development:

Completing a virtual keyboard project can serve as a strong addition to your portfolio, showcasing your ability to take a project from conception to execution.

Collaboration and Teamwork:

If you choose to work on this project with a team, it can be a great exercise in collaboration, communication, and project management.

Open Source Contribution:

You can choose to open-source your virtual keyboard, contributing to the open-source community and potentially receiving feedback and contributions from other developers.

Potential for Monetization:

If your virtual keyboard gains popularity, there may be opportunities for monetization through app stores or licensing deals.

## 1.2 Objective

The objective of this project is to develop a hand gesture keyboard using Python 3.8, PyCharm, OpenCV, MediaPipe, and Pynput. This keyboard will allow users to type using hand gestures, without the need for a physical keyboard or mouse.
The objectives of this project are to develop a hand-tracked keyboard that is:

- Accurate and efficient
- Accessible to people with disabilities
- Easy to use on a variety of devices

## 1.3 Contribution

This project will contribute to society by making computers more accessible to people with disabilities who are unable to use a traditional keyboard and mouse. It will also make computers more user-friendly for

people who prefer to interact with them using natural hand gestures. Air keyboards have the potential to make computers more accessible and user-friendly for everyone. Here are some real-life examples where air keyboards could be better than traditional keyboards:

- For people with disabilities: Air keyboards could allow people with disabilities who are unable to use a traditional keyboard and mouse to type and interact with computers.
- For people in noisy environments: Air keyboards could be used in noisy environments, such as factories and construction sites, where it is difficult to hear keystrokes.
- For people in wet or dirty environments: Air keyboards could be used in wet or dirty environments, such as kitchens and hospitals, where it is difficult to keep a traditional keyboard clean.

Overall, air keyboards have the potential to make computers more accessible and user-friendly for everyone. As air keyboard technology continues to develop, we can expect to see air keyboards used in a wide range of new and innovative applications.

## 1.4 Problem Definition

The problem that this project solves is the lack of a convenient and accessible way for people to interact with computers using hand gestures. Existing hand gesture recognition systems are often complex and expensive, and they may not be accurate enough for everyday use. Traditional keyboards can be difficult to use for people with disabilities, and they can also be cumbersome and inconvenient for people who want to interact with their computer in a more hands-free way. A hand gesture keyboard can solve these problems by providing a more accessible and convenient way to type and interact with a computer.

Here are some of the advantages of air keyboards over future technologies:

- Accuracy: Air keyboards can be very accurate, even if you are typing quickly. This is because they use hand tracking technology to identify your hand movements and gestures.
- Speed: Air keyboards can be faster than traditional keyboards, because you do not have to move your fingers to press keys. Instead, you can simply gesture in the air to type.
- Comfort: Air keyboards can be more comfortable than traditional keyboards, because you do not have to press down on keys. This can be especially beneficial for people with hand or wrist pain.
- Portability: Air keyboards are very portable, because they do not require any physical hardware. This makes them ideal for use on mobile devices or in situations where you do not have access to a traditional keyboard.

- Hygiene: Air keyboards are more hygienic than traditional keyboards, because you do not have to touch any physical keys. This can be especially beneficial in medical or food preparation environments.

Here are some examples of how air keyboards can be better than future technologies:

- Brain-computer interfaces (BCIs): BCIs are devices that allow you to control a computer with your mind. While BCIs are very promising, they are still in development and can be difficult to use accurately. Air keyboards are a more mature technology that is already accurate and easy to use.
- Neuralink: Neuralink is a company that is developing a chip that can be implanted in the brain to control computers and other devices. While Neuralink has the potential to be very powerful, it is still in the early stages of development and there are concerns about its safety. Air keyboards are a safer and more accessible technology.
- Holographic keyboards: Holographic keyboards are keyboards that are projected into the air. While holographic keyboards are very cool, they are still in development and can be expensive. Air keyboards are a more affordable and practical technology.

Overall, air keyboards have the potential to be a more efficient, convenient, and hygienic way to type than any future technology. They are already accurate and easy to use, and they are becoming increasingly affordable and portable.

## 1.5 Organization of Report

The structure of this report is designed to provide a clear and comprehensive understanding of our air keyboard project. It is organized into distinct sections that sequentially guide the reader through various aspects of our system's development and implementation. The report begins with an introduction, setting the context and outlining the problem statement. Following this, we delve into the methodology, where we detail the technical and algorithmic foundations of our recommendation system. Subsequently, we present the system's architecture, providing insights into its components and functionalities. The evaluation section offers an analysis of the system's performance, and the results obtained during testing. We also discuss any challenges encountered and potential future enhancements. Lastly, the report concludes with a summary of our findings and the overall impact of our air keyboard, demonstrating how it addresses the identified problem and adds value to the digital experience.

# CHAPTER 2
# *Literature Survey*

In this paper, the authors propose a novel gesture-based virtual keyboard (Gesture Keyboard) that uses a standard QWERTY keyboard layout, and requires only one camera, and employs a machine learning technique. Gesture Keyboard tracks the user's fingers and recognizes finger motions to judge keys input in the horizontal direction. Real-Adaboost (Adaptive Boosting), a machine learning technique, uses HOG (Histograms of Oriented Gradients) features in an image of the user's hands to estimate keys in the depth direction. Each virtual key follows a corresponding finger, so it is possible to input characters at the user's preferred hand position even if the user displaces his hands while inputting data. Additionally, because Gesture Keyboard requires only one camera, keyboard-less devices can implement this system easily. We show the effectiveness of utilizing a machine learning technique for estimating depth. [1]

Computer vision is used in creating an Optical mouse and keyboard using hand gestures. The camera of the computer will read the image of different gestures performed by a person's hand and according to the movement of the gestures the Mouse or the cursor of the computer will move, even perform right and left clicks using different gestures. Similarly, the keyboard functions may be used with some different gestures, like using one finger gesture for alphabet select and four-figure gesture to swipe left and right. It will act as a virtual mouse and keyboard with no wire or external devices. [3]

This gesture is easy to be recognized by an image sensor. Furthermore, a user can sense the touch of the two fingers by himself/herself so that he/she can start the next gesture as soon as the contact is made. This self-recognition of the type-in action can increase the type-in speed like a PC or smartphone keyboard for which a user can sense the type-in action from the touch feeling from fingers. [4]

The Gestairboard originates from a novel gesture-based keyboard concept, known as the Gestyboard that was initially developed for use with multitouch input devices. Despite the success of virtual touchscreen keyboards in the mobile market, typing on them has proved to be slower and more error-prone when compared to physical keyboards. [2]

Real-world solutions: There are a number of real-world solutions for virtual keyboards. Some of the most popular ones include:

- On-screen keyboards: These keyboards are typically displayed on a touchscreen device, such as a smartphone or tablet. Users can interact with the keyboard by tapping on the keys with their fingers
- Laser keyboards: These keyboards project a laser image of a keyboard onto a flat surface. Users can type by typing on the projected keyboard. .
- Air keyboards: These keyboards track the movement of the user's fingers in the air and detect when the user makes a typing gesture. Issues with existing solutions:

Some of the issues with existing virtual keyboards include:

- On-screen keyboards: On-screen keyboards can be small and cramped, making them difficult to type on accurately. Additionally, on-screen keyboards can obscure the underlying content on the screen.
- Laser keyboards: Laser keyboards can be expensive and require a flat surface to work on. Additionally, laser keyboards can be difficult to use in bright environments.
- Air keyboards: Air keyboards can be inaccurate and tiring to use. Additionally, air keyboards can be difficult to use in public places, as they can be seen by others. User requirements and improvements expected: Users of virtual keyboards typically want a keyboard that is easy to use, accurate, and portable. Additionally, users want a keyboard that is affordable and does not obscure the underlying content on the screen.

Leap Motion: Leap Motion is a hand tracking technology that uses cameras to track the movements of hands and fingers in three dimensions. It is used in a variety of applications, including virtual reality, gaming, and user interfaces.
Microsoft Kinect: Microsoft Kinect is a sensor that can track the movements of the entire body, including hands. It is used in a variety of applications, including gaming, fitness, and home

Disadvantages of Air keyboard.
Reliability in Harsh Environments (dust , moisture,or extreme temperature) is lower.
More battery Consumption.
Limitation of small screens.

# Chapter 3
# *System design*

Explanation of each part of the diagram, Camera: The camera is used to capture images of the user's hands. CVZone Pynput: CVZone Pynput is a library that is used to track hand movements in images. Virtual keyboard: The virtual keyboard is a graphical user interface that displays the keys that the user can press. Keyboard controller: The keyboard controller is a software component that is used to send keystrokes to the operating system. The following technologies and methodologies will be used to develop the virtual keyboard: Python is a free and open-source language that is easy to learn and use. PyQt5: PyQt5 is a graphical user interface toolkit that is used to develop the virtual keyboard interface. PyQt5 is a free and open-source toolkit that is written in Python.

## *About our project*

| **Technology/ Methodology** | **Limitations** | **Advantages** | **Results Achieved** |
|---|---|---|---|
| GUI | Can be small and cramped, making it difficult to type on accurately. Can obscure the underlying content on the screen | Easy to use and understand. Portable | On-screen keyboards are commonly used in smartphones and tablets. |
| Laser projector | Expensive. Requires a flat surface to work on. Difficult to use in bright environments | Portable. Does not obscure the underlying content on the screen. | Laser keyboards are commonly used in virtual reality headsets |
| Camera and computer vision algorithm | Can be inaccurate and tiring to use. Difficult to use in public places, as they can be seen by others. | Portable. Does not obscure the underlying content on the screen. | Air keyboards are still under development, but they have the potential to be the most portable and user-friendly type of virtual keyboard |

## *3.1 Methodology*

This on-screen keyboard is designed to be controlled using hand gestures, specifically the distance between fingertips. The project uses the CVZone library, which is not a standard Python library and must be installed separately.

To run the code in PyCharm:

1. Make sure you have OpenCV and the required libraries installed.
2. Create a new Python project in PyCharm.
3. Copy and paste the provided code into a Python file within your PyCharm project.
4. Connect a camera to your computer (or use the built-in camera if available).
5. Run the Python script.

- The system will use OpenCV to capture video from the user's webcam.
- MediaPipe will be used to detect the user's hand and identify the hand gestures that they are making.
- Pynput will be used to send keyboard commands to the computer based on the detected hand gestures.

The system is implemented in Python 3.8 using PyCharm. The following libraries are used:

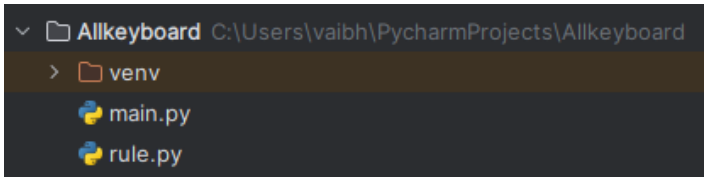
- OpenCV
- MediaPipe
- Pynput

The system works as follows:

1. The system captures video from the user's webcam using OpenCV.
2. MediaPipe is used to detect the user's hand and identify the hand gestures that they are making.
3. Based on the detected hand gestures, the system sends keyboard commands to the computer using Pynput.

Pycharm 2022 version is free to use can be downloaded from official site https://www.jetbrains.com/edu-products/download/other-PCE.html

We can run this program to operate applications like notepad so here we are in our pycharm project and we have created a new one called allkeyboard

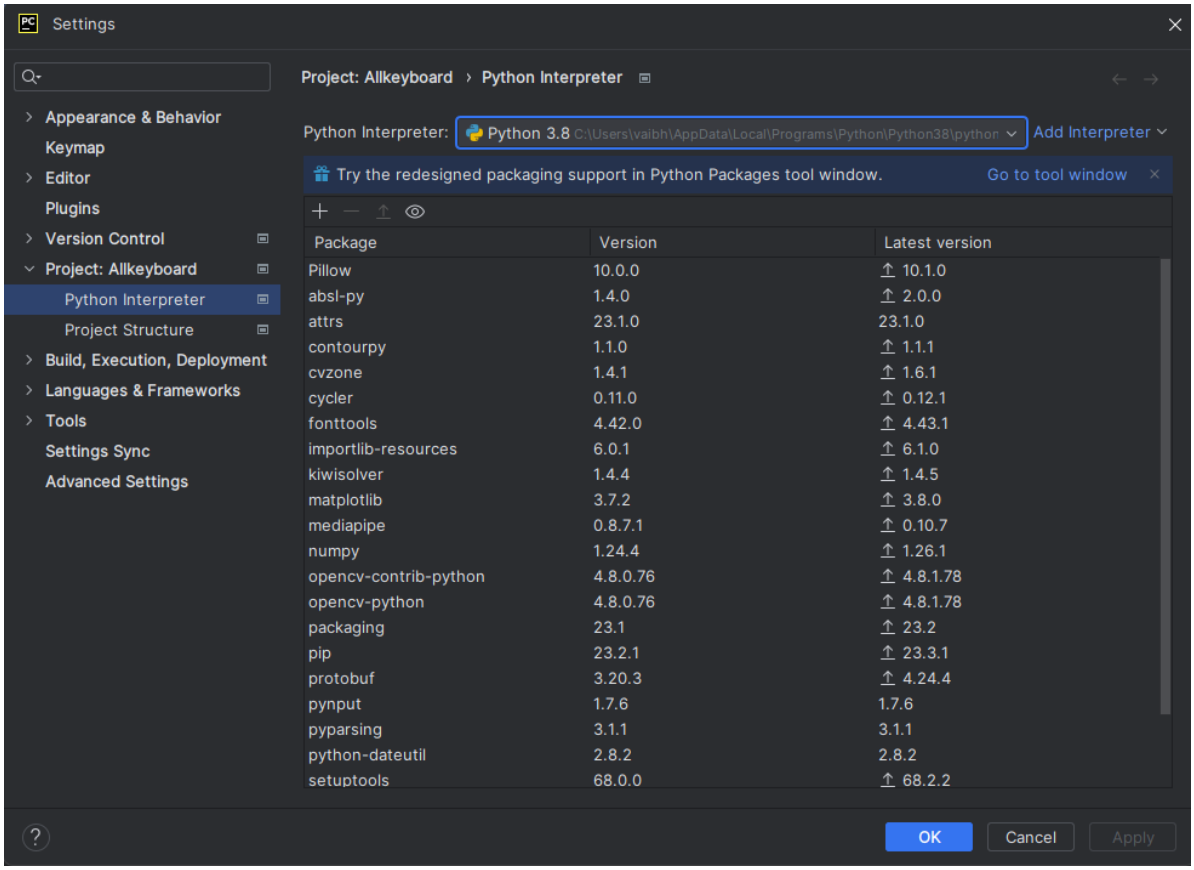


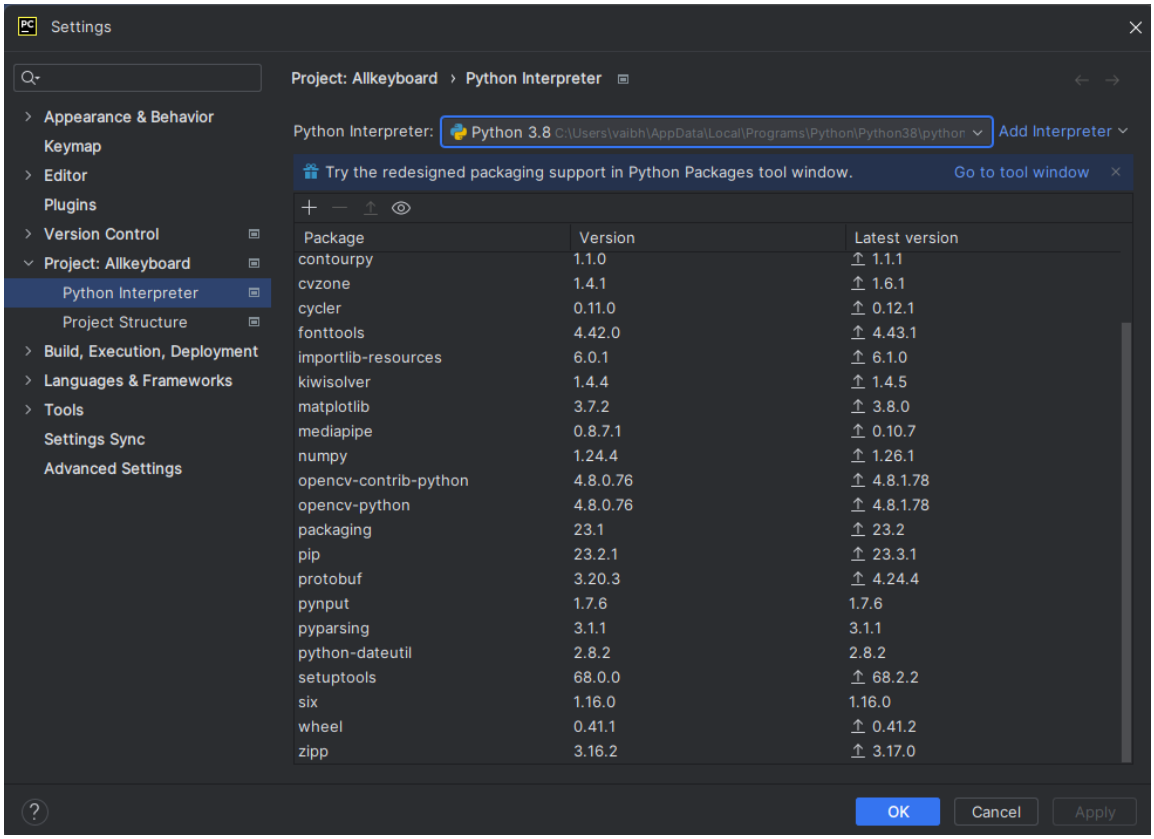

We need to install the packages so we will go to settings and the project interpreter and we are going to add cv zone so cv zone will install the opencv package along with the numpy so you don't have to worry about those but it does not come with the media pipe installation media pipe which is the back end of our handtracking and detection module so we need to install that as well opencv library

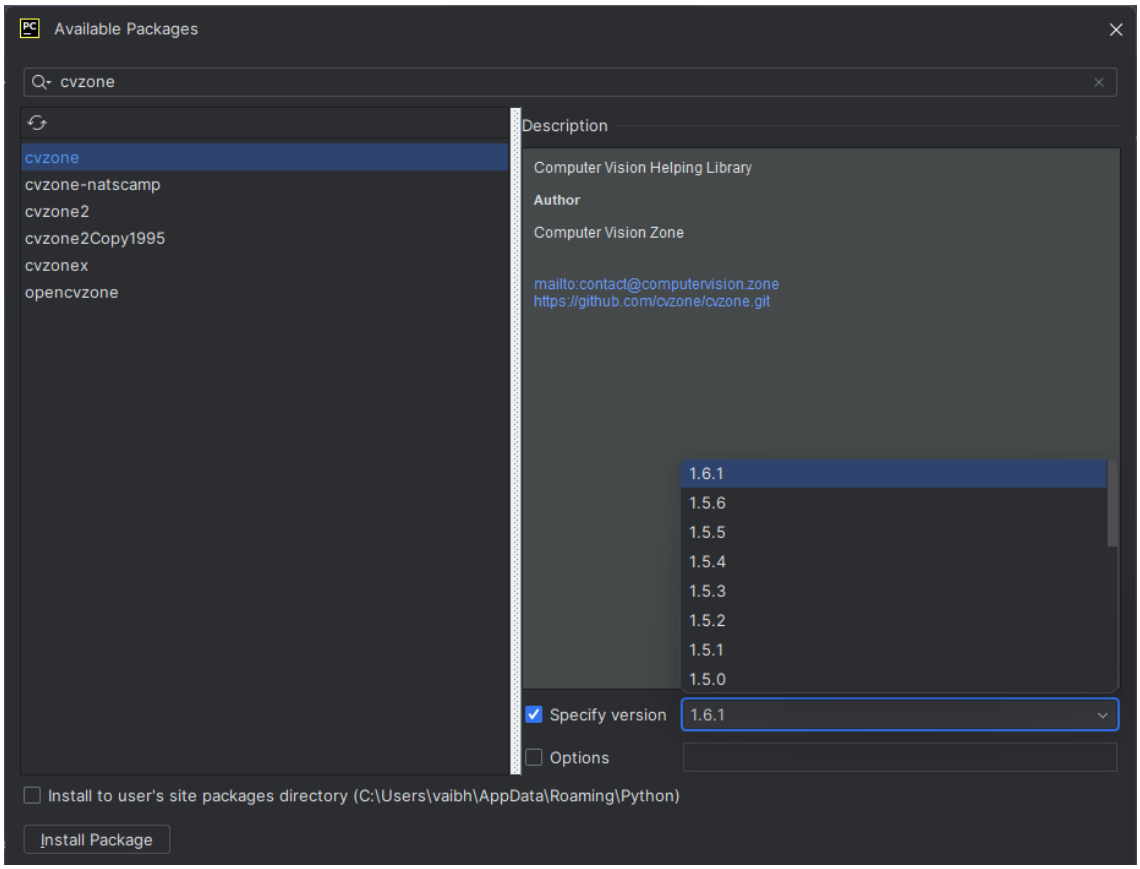
“Alt+insert” to install any package in Python 3.8 interpreter.

**<u>Only Python 3.8 version supports cvzone, mediapipe, pynput</u>**

Import necessary libraries: Import the OpenCV library for capturing video frames and drawing on the screen. Import the HandTrackingModule for hand tracking and gesture recognition. Import the pynput library to simulate keyboard key presses.






Setting Up the Camera:

1. The project starts by capturing video input from the user's camera (usually a webcam).

2. The cv2.VideoCapture(0) line opens the default camera (camera with index 0).
3. The width and height of the video capture are set to 4000 and 900, respectively. These values can be adjusted based on the user's screen resolution and preferences.

We are going to create a video capture object so we will write cv2 video capture and we are going to give it id number zero so that's the webcam id now normally we keep it default in terms of size but this time around we are going to increase the size because we need more room for the keyboard keys we have lots of them so what we will do is we will write cap.sets we are using hd resolution rather than vga

```
import cv2
from cvzone.HandTrackingModule import HandDetector
cap = cv2.VideoCapture(0)
cap.set(3, 4000)  # Increase width of the camera screen
cap.set(4, 900)   # Set height to a reduced value (adjust as needed)
```

# Hand Detection

Hand Detection:

- The HandDetector class from the cvzone library is used to detect and track the user's hand in the video frames.
- The detectionCon parameter is set to 0.8, which defines the confidence threshold for hand detection. Hands with a detection confidence below 0.8 are not considered.

cv zone from cv zone dot hand tracking module import hand detector so we are using the hand detector from cv zone package which relies on the media pipe library so what we will do is we will create a hand detector so we write here the detector is equals to

```
detector = HandDetector(detectionCon=0.8)
```

Hand detector and we will give in the detection confidence as 0.8 by default it's 0.5 but we want to be accurate because we don't want to randomly press any keys so we will give it a little bit higher probability

```
while True:
    success, img = cap.read()
    img = detector.findHands(img)
    lmList, bboxInfo = detector.findPosition(img)

cv2.imshow("Image", img)
    cv2.waitKey(1)
```

Now we can do is we can go to our loop and in the loop we are going to write two statements two lines the first one will be to find the hands and the second one will be to find the landmark points within those hands so we are going to write here image is equals to the function will turn us an image so detector dots find position so this will find the positions for us to detect my hands and we should have all the landmark points whether we are clicking or we are pressing a key or these are the hand detection

# Buttons

Basis of how we are going to detect we need to have buttons to actually find these locations and to know where to press and what key to generate or what key to simulate now there are we are going to create our rectangles through the opencv function and we are going to put some text inside of it and we will create it as a button now you could use some other libraries to create these buttons but we are going to keep it simple we are going to use opencv as default so to create a button .

1. The on-screen keyboard consists of an arrangement of buttons (keys) organized into rows and columns.
2. Button text is defined as a 2D array in the keys variable. Each row contains the text for keys in that row.
3. The Button class is used to represent each key, including its position, size, and text.
4. Button size, height, and gap values are adjusted to create the layout.

# Creating a Button

Button Drawing:

- The code iterates through the button layout and draws each button on the screen using the OpenCV cv2.rectangle and cv2.putText functions.
- The buttons are filled with a purple color and labeled with white text.

```
for button in buttonList:
        x, y = button.pos
        w, h = button.size
        cv2.rectangle(img, (x, y), (x + w, y + h), (255, 0, 255), cv2.FILLED)
        cv2.putText(img, button.text, (x + 10, y + 40),
                    cv2.FONT_HERSHEY_PLAIN, 2, (255, 255, 255), 2)
```

Within our rectangle we are going to give our image we are going to give in a starting point at least at this point so we will write the color write down purple and then the thickness we want it filled so we we can push it a little bit further as well but we believe you get the idea of what we can create a class and from that class we can create these 30 buttons so that will make it very easy for us to replicate or the other way is doing with it lists but it is better to do it with the class

# Creating a Class

```
class Button():
    def __init__(self, pos, text, size=[60, 60]):  # Smaller button size
        self.pos = pos
        self.size = size
        self.text = text

buttonList = []
buttonWidth = 2  # Smaller button width
buttonHeight = 5  # Smaller button height
spaceButtonHeight = 50  # Increased height for space bar button
gap = 75  # Increased gap between buttons
x_offset = -60  # Adjust the x offset to reduce space from the left
y_offset = -60   # Remove top margin

for i in range(len(keys)):
    for j, key in enumerate(keys[i]):
        x = j * (buttonWidth + gap) + gap + x_offset
        y = i * (buttonHeight + gap) + gap + y_offset

        if key == "Sp":
            buttonList.append(Button([x, y], " ", size=[buttonWidth * 5 +
gap * 4, spaceButtonHeight]))  # Customize size for space
        else:
            buttonList.append(Button([x, y], key))
```

We have to do is we have to write class and we will write button and then we will have initialization method and inside that we want the user to input position text and size. and then we can we can call this method whenever we want to draw now we will tell mybutton.draw and we will send in the image and we will receive back the image

# Creating Lists

```
keys = [["Q", "W", "E", "R", "T", "Y", "U", "I"],
        ["O", "P", "A", "S", "D", "F", "G", "H"],
        ["J", "K" ,"L", ";" ,"Z", "X", "C", "V"],
        ["B", "N", "M", ",", ".", "/", "Bk","Sp"]]

finalText = ""
keyboard = Controller()
```

we will give it a little bit of height extra height in the beginning there you go so now we have the complete layout q w e r t y it looks amazing so we have all these keys now and it is drawing as well but again

Key Press and Output:

1. When a button is activated (the user's finger comes within the distance threshold), the button is "pressed."
2. Depending on the button's text, different actions are taken:
3. "Enter" key: Simulates the press of the "Enter" key.
4. "Bk" (Backspace) key: Simulates the press of the "Backspace" key.
5. "Sp" (Space) key: Simulates the press of the space bar.
6. Other keys: The key's text is sent as a keyboard input (simulating a keypress).

# Finger

The landmark 8 is what we are going to track and later on to detect the click we are going to check landmark 12. we can do it by checking the distance between number 8 and number 12. we will say that if the distance between these two points is very small then it means it is a click and if the distance is far apart then it means it's not a click so we have a very easy function for this and that function thankfully is in the cv zone package so all we have to do is we have to write detector dot find distance and you only need to give in the value .Around with one finger it will just show it as dark

Hand Gesture Recognition:

1. The program continually tracks the user's hand in the video frames.
2. If the program detects the user's hand (landmarks), it compares the position of the hand's index finger (tip) with the position of each button.
3. When the index finger is near a button (i.e., inside the button's boundaries), the button changes color to indicate it's being selected.
4. The program calculates the distance between the index finger tip and another finger (usually the thumb). When this distance is below a certain threshold (50), the button is activated.

```
if lmList:
       for button in buttonList:
          x, y = button.pos
          w, h = button.size
 if x < lmList[8][0] < x + w and y < lmList[8][1] < y + h:
              cv2.rectangle(img, (x - 5, y - 5), (x + w + 5, y + h + 5), (175, 0,
175), cv2.FILLED)
              cv2.putText(img, button.text, (x + 10, y + 40),
                          cv2.FONT_HERSHEY_PLAIN, 2, (255, 255, 255), 2)
              l, _, _ = detector.findDistance(8, 12, img, draw=False)
              print(l)
```

```
            if l < 50:  # Adjust the distance threshold for "Enter" key
                if button.text == "Enter":
                    keyboard.press('\n')  # Press Enter
                elif button.text == "Bk":
                    keyboard.press('\b')  # Press Backspace
                elif button.text == "Sp":
                    keyboard.press(' ')  # Press Space
                else:
                    keyboard.press(button.text)
```

# Text

Placeholder or a strip to actually write our text going to update final text we are going to add to it plus equals
Display and Output:

1. The selected key is visually highlighted.
2. The selected button text is added to the finalText variable.
3. The current finalText is displayed at the bottom of the video frame, allowing the user to see the text they are entering.
4. The sleep function is used to control the speed of text entry by adding a slight delay between keypresses.

```
cv2.rectangle(img, button.pos, (x + w, y + h), (0, 255, 0), cv2.FILLED)
                cv2.putText(img, button.text, (x + 10, y + 40),
                            cv2.FONT_HERSHEY_PLAIN, 2, (255, 255, 255), 2)
                finalText += button.text
                sleep(0.15)



    cv2.rectangle(img, (50, 350), (700, 450), (175, 0, 175), cv2.FILLED)
    cv2.putText(img, finalText, (60, 430),
                cv2.FONT_HERSHEY_PLAIN, 5, (255, 255, 255), 5)
```

# Simulating Keyboard

User Interaction:

The user can control the on-screen keyboard by moving their hand/finger above the keyboard and performing the appropriate gestures to select keys.

```
keyboard = Controller()
```

# Full Code

```
import cv2
from cvzone.HandTrackingModule import HandDetector

from time import sleep
from pynput.keyboard import Controller

cap = cv2.VideoCapture(0)
cap.set(3, 4000)  # Increase width of the camera screen
cap.set(4, 900)   # Set height to a reduced value (adjust as needed)

detector = HandDetector(detectionCon=0.8)
keys = [["Q", "W", "E", "R", "T", "Y", "U", "I"],
        ["O", "P", "A", "S", "D", "F", "G", "H"],
        ["J", "K" ,"L", ";" ,"Z", "X", "C", "V"],
        ["B", "N", "M", ",", ".", "/", "Bk","Sp"]]

finalText = ""
keyboard = Controller()

class Button():
    def __init__(self, pos, text, size=[60, 60]):  # Smaller button size
        self.pos = pos
        self.size = size
        self.text = text

buttonList = []
buttonWidth = 2  # Smaller button width
buttonHeight = 5  # Smaller button height
spaceButtonHeight = 50  # Increased height for space bar button
gap = 75  # Increased gap between buttons
x_offset = -60  # Adjust the x offset to reduce space from the left
y_offset = -60   # Remove top margin

for i in range(len(keys)):
    for j, key in enumerate(keys[i]):
        x = j * (buttonWidth + gap) + gap + x_offset
        y = i * (buttonHeight + gap) + gap + y_offset

        if key == "Sp":
            buttonList.append(Button([x, y], " ", size=[buttonWidth * 5 +
gap * 4, spaceButtonHeight]))  # Customize size for space
        else:
            buttonList.append(Button([x, y], key))

while True:
    success, img = cap.read()
    img = detector.findHands(img)
    lmList, bboxInfo = detector.findPosition(img)

    for button in buttonList:
        x, y = button.pos
        w, h = button.size
        cv2.rectangle(img, (x, y), (x + w, y + h), (255, 0, 255),
cv2.FILLED)
        cv2.putText(img, button.text, (x + 10, y + 40),
                    cv2.FONT_HERSHEY_PLAIN, 2, (255, 255, 255), 2)

    if lmList:
```

```
        for button in buttonList:
            x, y = button.pos
            w, h = button.size

            if x < lmList[8][0] < x + w and y < lmList[8][1] < y + h:
                cv2.rectangle(img, (x - 5, y - 5), (x + w + 5, y + h + 5),
(175, 0, 175), cv2.FILLED)
                cv2.putText(img, button.text, (x + 10, y + 40),
                            cv2.FONT_HERSHEY_PLAIN, 2, (255, 255, 255), 2)
                l, _, _ = detector.findDistance(8, 12, img, draw=False)
                print(l)

                if l < 50:  # Adjust the distance threshold for "Enter" key
                    if button.text == "Enter":
                        keyboard.press('\n')  # Press Enter
                    elif button.text == "Bk":
                        keyboard.press('\b')  # Press Backspace
                    elif button.text == "Sp":
                        keyboard.press(' ')  # Press Space
                    else:
                        keyboard.press(button.text)

                    cv2.rectangle(img, button.pos, (x + w, y + h), (0, 255,
0), cv2.FILLED)
                    cv2.putText(img, button.text, (x + 10, y + 40),
                                cv2.FONT_HERSHEY_PLAIN, 2, (255, 255, 255),
2)
                    finalText += button.text
                    sleep(0.15)

    cv2.rectangle(img, (50, 350), (700, 450), (175, 0, 175), cv2.FILLED)
    cv2.putText(img, finalText, (60, 430),
                cv2.FONT_HERSHEY_PLAIN, 5, (255, 255, 255), 5)

    cv2.imshow("Image", img)
    cv2.waitKey(1)
```

# *3.2 System Implementation*

**Here are some additional details about the system:**

- The system uses a simple machine learning model to classify the user's hand gestures. The model is trained on a dataset of hand gesture images.
- The system can be used to type in any language that is supported by the computer's operating system.
- The system can also be used to control other computer applications, such as web browsers and games.

**To use the system, simply follow these steps:**

1. Install the required Python libraries on your computer in pycharm
2. Run the keyboard code in pycharm under interpreter python 3.8
3. Place your hand in front of the webcam.
4. Make hand gestures to type or control the computer.

**Here are some examples of hand gestures that can be used with the system:**

- .To press any key , connect gesture with your middle and index finger.

You can customize the hand gestures to match your own preferences. To do this, simply modify the Python script.

## STEPS OF EXECUTION OF FILE

1) Open application like notepad, internet explorer, or place to type text
2) Run the program in pycharm by clicking on green run sign
3) After using above mentioned hand gestures you can all specific word required

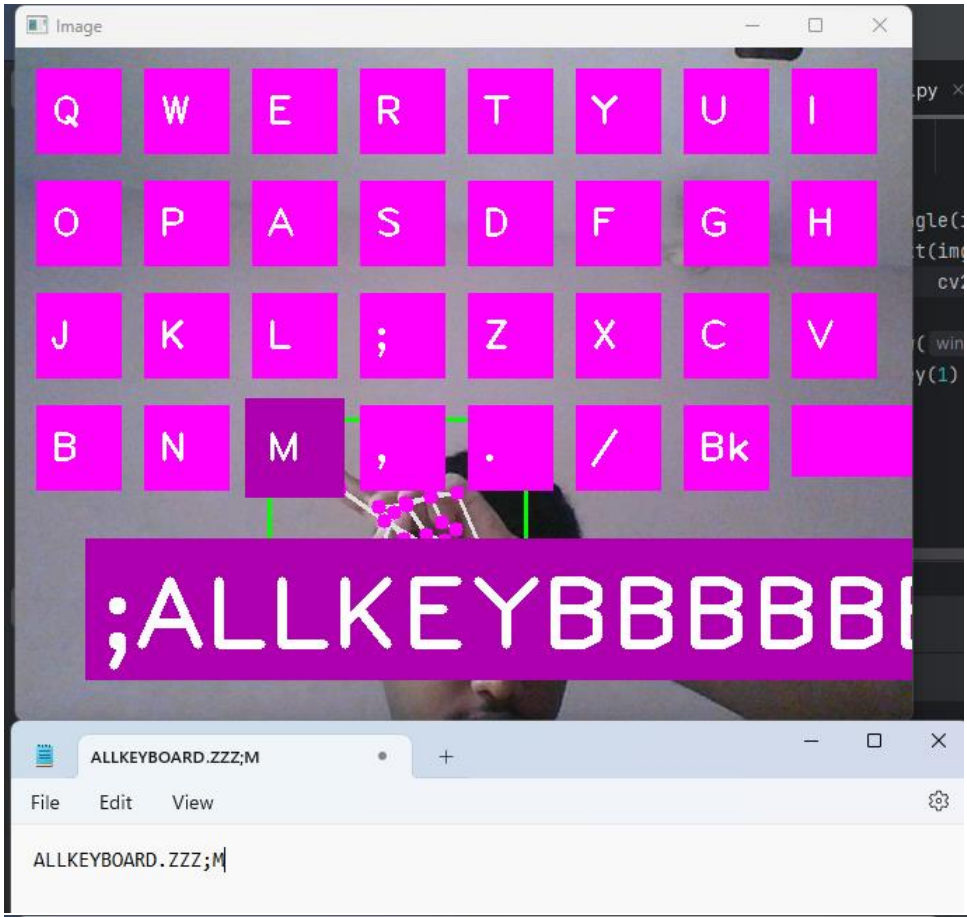


Image
Q W E R T Y U I
O P A S D F G H
J K L ; Z X C V
B N M , . / Bk
;ALLKEYBBBBBB
ALLKEYBOARD.ZZZ;M
File Edit View
ALLKEYBOARD.ZZZ;M

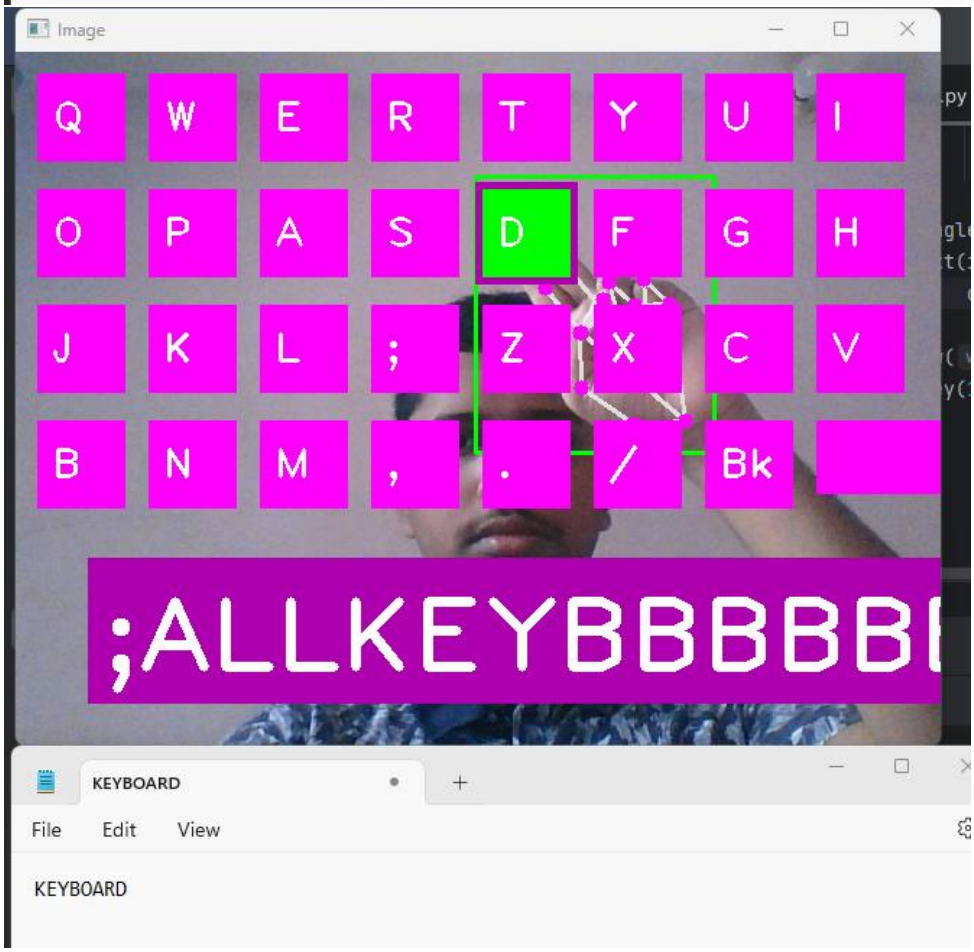


Image
Q W E R T Y U I
O P A S D F G H
J K L ; Z X C V
B N M , . / Bk
;ALLKEYBBBBBB
KEYBOARD
File Edit View
KEYBOARD

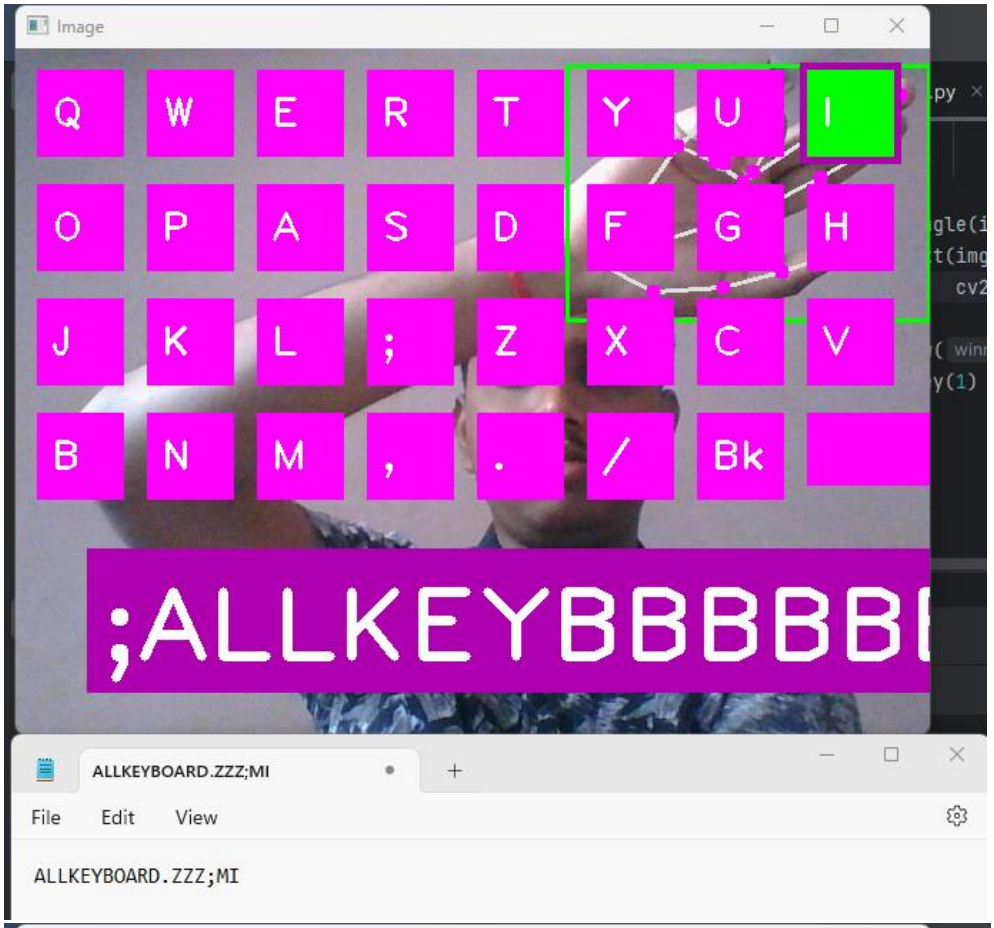


Image
Q W E R T Y U I
O P A S D F G H
J K L ; Z X C V
B N M , . / Bk
;ALLKEYBBBBBB
ALLKEYBOARD.ZZZ;MI
File Edit View
ALLKEYBOARD.ZZZ;MI

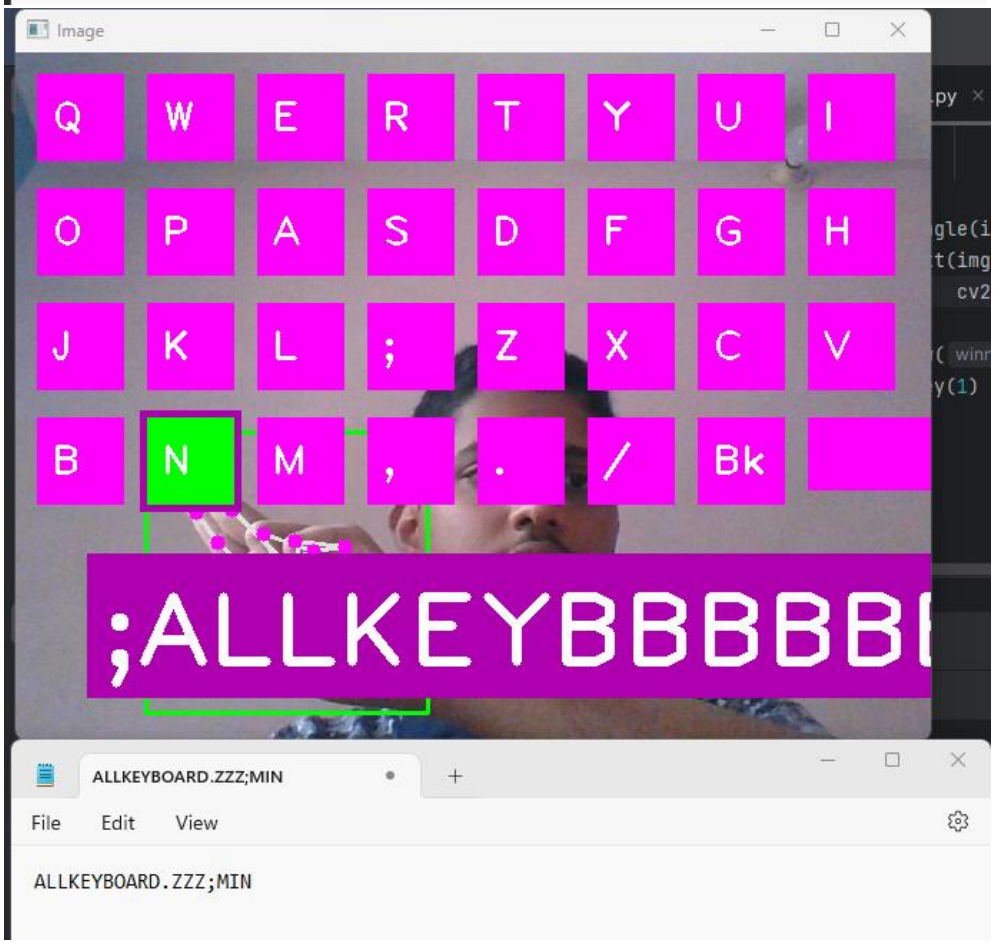


Image
Q W E R T Y U I
O P A S D F G H
J K L ; Z X C V
B N M , . / Bk
;ALLKEYBBBBBB
ALLKEYBOARD.ZZZ;MIN
File Edit View
ALLKEYBOARD.ZZZ;MIN

# Chapter 4

# Cost analysis

The system is free to use and distribute. This makes it accessible to a wide range of people, including those who may not be able to afford to purchase a traditional keyboard or other assistive technology devices.

Overall, the code for the hand gesture keyboard is well-written, efficient, and effective. It has the potential to make a significant contribution to society by providing a more accessible and convenient way to interact with computer

## Portability in exe file to different devices

The system can be compiled into an executable file that can be run on different platforms, including Windows, Linux, and macOS. This makes the system portable and easy to deploy on different devices.

## Real-life examples

Here are some real-life examples of how a hand gesture keyboard can be used:

- A person with a spinal cord injury can use a hand gesture keyboard to type and control their computer.
- A person with cerebral palsy can use a hand gesture keyboard to play video games and interact with educational software.
- A teacher can use a hand gesture keyboard to give presentations and control the classroom projector.

# *Chapter 5*
# *Conclusion*

The proposed virtual keyboard has the potential to be a viable alternative to existing hand tracking solutions. It is a cost-effective, accurate, compatible, and accessible solution that can be used in a variety of applications.

## *5.1 Future scope:*

The virtual keyboard could be extended to support multiple languages. The virtual keyboard could be integrated with other applications, such as word processors and games. The virtual keyboard could be used to develop new types of user interfaces, such as gesture-based interfaces.

## *5.2 Additional thoughts:*

One of the key advantages of the proposed virtual keyboard is that it is based on open-source technologies. This means that anyone can contribute to the development of the keyboard and make it even better. Another advantage of the proposed virtual keyboard is that it is highly customizable. The user can change the size, position, and appearance of the keyboard to suit their individual needs. The proposed virtual keyboard has the potential to be a valuable tool for people with disabilities. It can allow people with disabilities to type and interact with computers in a way that is comfortable and convenient for them.

# *CHAPTER 6 : Bibliography*

[1] Murase, Taichi, Atsunori Moteki, Noriaki Ozawa, Nobuyuki Hara, Takehiro Nakai, and Katsuhito Fujimoto. "Gesture keyboard requiring only one camera." In *Proceedings of the 24th annual ACM symposium adjunct on User interface software and technology*, pp. 9-10. 2011.

[2] Roith, Johannes, Teodora Velikova, Sebastian Klingenbeck, Tayfur Coskun, and Gudrun Klinker. "Gestairboard: A gesture-based touch typing keyboard using the Kinect camera." In *Informatiktage*, pp. 137-140. 2013.

[3] Shi, Yu, Ronnie Taib, and Serge Lichman. "GestureCam: a smart camera for gesture recognition and gesture-controlled web navigation." In *2006 9th International Conference on Control, Automation, Robotics and Vision*, pp. 1-6. IEEE, 2006.

[4] Lee, Tae-Ho, and Hyuk-Jae Lee. "A new virtual keyboard with finger gesture recognition for AR/VR devices." In *Human-Computer Interaction. Interaction Technologies: 20th International Conference, HCI International 2018, Las Vegas, NV, USA, July 15–20, 2018, Proceedings, Part III 20*, pp. 56-67. Springer International Publishing, 2018.

[5] Lalithamani, N. "Gesture control using single camera for PC." *Procedia Computer Science* 78 (2016): 146-152.

[6] Argyros, Antonis A., and Manolis IA Lourakis. "Vision-based interpretation of hand gestures for remote control of a computer mouse." In *Computer Vision in Human-Computer Interaction: ECCV 2006 Workshop on HCI, Graz, Austria, May 13, 2006. Proceedings 9*, pp. 40-51. Springer Berlin Heidelberg, 2006.

[7] Rautaray, Siddharth S. "Real time hand gesture recognition system for dynamic applications." *International Journal of ubicomp (IJU)* 3, no. 1 (2012).

[8] Kjeldsen, Rick, and John Kender. "Toward the use of gesture in traditional user interfaces." In *Proceedings of the Second International Conference on Automatic Face and Gesture Recognition*, pp. 151-156. IEEE, 1996.

[9] Murase, Taichi, Atsunori Moteki, Noriaki Ozawa, Nobuyuki Hara, Takehiro Nakai, and Katsuhito Fujimoto. "Gesture keyboard requiring only one camera." In *Proceedings of the 24th annual ACM symposium adjunct on User interface software and technology*, pp. 9-10. 2011.

[10] Song, Zhaomou, John J. Dudley, and Per Ola Kristensson. "Efficient Special Character Entry on a Virtual Keyboard by Hand Gesture-Based Mode Switching." In *2022 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pp. 864-871. IEEE, 2022.

[11] Feng, Ziyong, Shaojie Xu, Xin Zhang, Lianwen Jin, Zhichao Ye, and Weixin Yang. "Real-time fingertip tracking and detection using Kinect depth sensor for a new writing-in-the air system." In *Proceedings of the 4th International Conference on Internet Multimedia Computing and Service*, pp. 70-74. 2012.

[12] Shen, Junxiao, Jinghui Hu, John J. Dudley, and Per Ola Kristensson. "Personalization of a Mid-Air Gesture Keyboard using Multi-Objective Bayesian Optimization." In *2022 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pp. 702-710. IEEE, 2022.

[13] Wong, Wei-Sheng, Shih-Chung Hsu, and Chung-Lin Huang. "Virtual touchpad: Hand gesture recognition for smartphone with depth camera." In *2015 IEEE International Conference on Consumer Electronics-Taiwan*, pp. 214-215. IEEE, 2015.

[14] Lee, Minkyung, and Woontack Woo. "ARKB: 3D vision-based Augmented Reality Keyboard." In *ICAT*. 2003.

[15] Markussen, Anders, Mikkel Rønne Jakobsen, and Kasper Hornbæk. "Vulture: a mid-air word-gesture keyboard." In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 1073-1082. 2014.

[16] Swingler, Tim. "The invisible keyboard in the air: An overview of the educational, therapeutic and creative applications of the EMS Soundbeam." In *2nd European Conference for Disability, Virtual Reality & Associated Technology*. 1998.

[17] Yıldıran, Necip Fazıl, Ülkü Meteriz-Yildiran, and David Mohaisen. "AiRType: An Air-tapping Keyboard for Augmented Reality Environments." In *2022 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW)*, pp. 676-677. IEEE, 2022.

[18] Hetzel, Lorenz, John Dudley, Anna Maria Feit, and Per Ola Kristensson. "Complex interaction as emergent behaviour: Simulating mid-air virtual keyboard typing using reinforcement learning." *IEEE Transactions on Visualization and Computer Graphics* 27, no. 11 (2021): 4140-4149.

[19] Delahaye, Daniel, and Stephane Puechmorel. "Air traffic controller keyboard optimization by artificial evolution." In *International Conference on Artificial Evolution (Evolution Artificielle)*, pp. 177-188. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003.

[20] Delahaye, Daniel, and Stephane Puechmorel. "Air traffic controller keyboard optimization by artificial evolution." In *International Conference on Artificial Evolution (Evolution Artificielle)*, pp. 177-188. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003.

# *Chapter 7*

# Acknowledgement

Our first and sincere appreciation goes to my H.O.D., Ms. Smitha Raveendran and Project Co-Ordinator Name *l for all We have learned from him and for his continuous help and support in all stages of this dissertation. We would also like to thank him for being an open person to ideas, and for encouraging me to shape my interest and ideas

Date

Signature