

**AI VIRTUAL KEYBOARD**

**A PROJECT REPORT**

**for**

**Artificial Intelligence**

**in**

**B.Tech (IT)**

**by**

**Raavi Shiva Seshi Reddy(19BIT0173)**

**Fall Semester 2021 -2022**

**Under the Guidance of**

**Dr. R.Subhashini**

**SITE**



**VIT<sup>®</sup>**  
**Vellore Institute of Technology**  
(Deemed to be University under section 3 of UGC Act, 1956)

**School of Information Technology and Engineering**

**NOV, 2021**

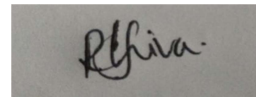
### **DECLARATION BY THE CANDIDATE**

We here by declare that the project report entitled “AI VIRTUAL KEYBOARD” submitted by us to Vellore Institute of Technology University, Vellore in partial fulfillment of the requirement for the award of the course Artificial Intelligence (ITE2010) is a record of bonafide project work carried out by us under the guidance of DR. R SUBHASHINI. We further declare that the work reported in this project has not been submitted and will not be submitted, either in part or in full, for the award of any other course.

Place : Vellore

Signature

Date : 09/12/2021

A rectangular box containing a handwritten signature in black ink. The signature appears to be 'R. Subhashini' written in a cursive style.



**VIT<sup>®</sup>**

**Vellore Institute of Technology**  
(Deemed to be University under section 3 of UGC Act, 1956)

## **School of Information Technology & Engineering [SITE]**

### **CERTIFICATE**

This is to certify that the project report entitled "TITLE" submitted by Student Name 1 (Reg. No), Student Name 2 (Reg.No) to Vellore Institute of Technology University, Vellore in partial fulfillment of the requirement for the award of the course Artificial Intelligence (ITE2010) is a record of bonafide work carried out by them under my guidance.

DR.R SUBHASHINI

GUIDE

SITE

# **AI VIRTUAL KEYBOARD**

**SHIVA**

**Department of Information Technology, VIT University, Vellore,  
Tamil Nadu, India**

## **Abstract**

*Virtual Keyboard is just another example of today's computer trend of 'smaller and faster'. It uses sensor technology and artificial intelligence to let users work on any surface as if it were a keyboard. Computer vision is one of the parts of Artificial intelligence here I am going to use openCV to develop my project. OpenCV is the most popular library for the task of computer vision, it is a cross-platform open-source library for machine learning, image processing, etc. using which real-time computer vision applications are developed. CVzone is a computer vision package, where it uses OpenCV and MediaPipe libraries as its core that makes us easy to run like hand tracking, face detection, facial landmark detection, pose estimation, etc., and also image processing and other computer vision-related applications. Check here for more information.*

## **I. Introduction:**

In this project I am going to create a virtual key board based on Artificial intelligence. Virtual keyboard uses hand tracking and detecting module and to let users work on laptop screen surface as if it were a keyboard. This project develops an application to - visualize the keyboard of computer with the concept of image processing. So basically this is giving the virtual keyboard.

This AI virtual keyboard is because to avoid the physical keyboard instead of physical keyboard we can develop this Virtual keyboard this may be a great deal apart from the

physical features. However, this may required the substantial amount of space on computer screen to display the keyboard.

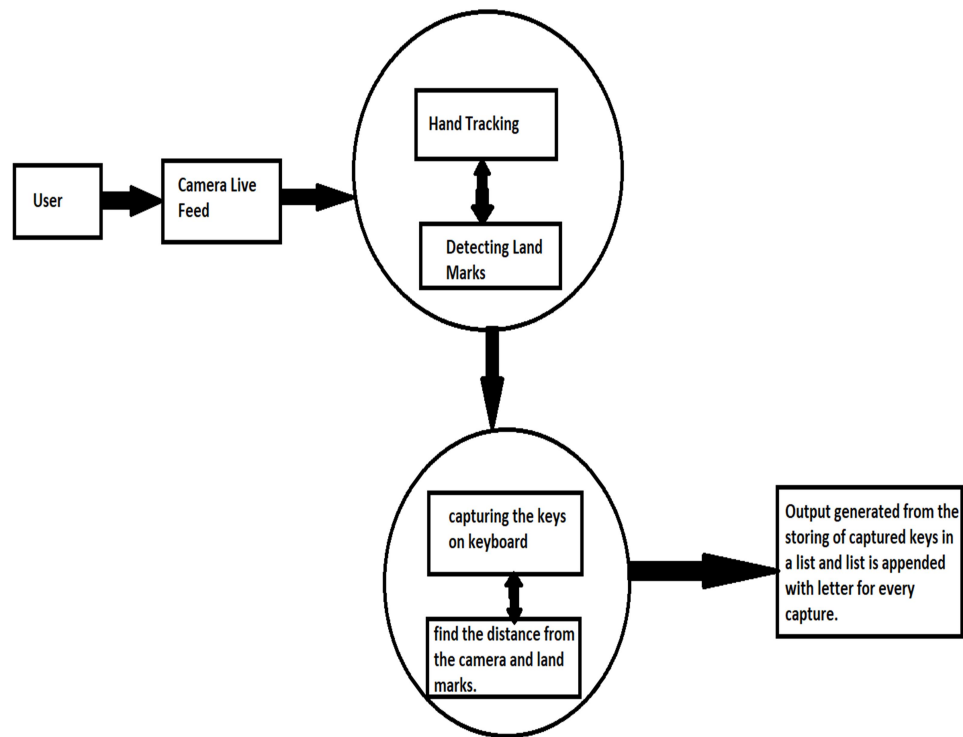
## **II. Background**

### **Modules/Libraries Required:**

1. numpy
2. Opencv
3. Cvzone
4. Pynput
5. Mediapipe.

CVzone is a computer vision package, where it uses OpenCV and MediaPipe libraries as its core that makes us easy to run like hand tracking, face detection, facial landmark detection, pose estimation, etc., and also image processing and other computer vision-related applications.

## Architecture diagram:



camera feed covers the hand finger moments and process the data to trace the finger movements and produces the output.

### **III. Literature Survey Sample :**

1. **Talveen kaur, Simran Makhijani** done the AI virtual keyboard project based on image processing it uses libraries like Dlib, OpenCV and other libraries with the help of which we would be able to detect points on face like eyes. By blink detection and eye detection able to find out the Character which a person wants to express.

2. **Chinnam Datta Sai Nikhil, Chukka Uma Someswara Rao, E.Brumancia, K.Indira, T.Anandhi, P.Ajitha** are done the virtual keyboard project based on the Finger Recognition.

3. **Dibakar saha** done the ai virtual keyboard based on the gesture recognition by processing the colour where he used to make a one colour finger strip he wear to his finger and the camera detect the colour where it blink before the screen.

4. **Zhi-hua Chen,<sup>1</sup> Jung-Tae Kim,<sup>1</sup> Jianning Liang,<sup>1</sup> Jing Zhang,<sup>1,2</sup> and Yu-Bo Yuan** done their project for hand movnet recognition by the way of,First, the hand is detected using the background subtraction method and the result of hand detection is transformed to a binary image. Then, the fingers and palm are segmented so as to facilitate the finger recognition. Moreover, the fingers are detected and recognized. Last, hand gestures are recognized using a simple rule classifier.

<b>Name of algorithm/technique</b>	<b>Advantages of techniques</b>	<b>Disadvantages of using this technique</b>
Eye Blink Detection	One of the best image processing technique to make a AI virtual keyboard instead of	Continues blinking of eye may not at proper thing for the it consumes lot time to get the output, there May be chance of wrong output.
Finger movement Recognition.	One of most developing and most used technology which make a lot of good impact on project	accuracy rate of working is slower
Gesture Recognition by colour image processing	Colour detection technique is main good advantage of the project	Time consuming. There are many disadvantages based on colour model chosen.
Hand Gesture Recognition	Enable the human computer interaction , easily understand the concepts by gestures representation for disabled people.	May be a chance of improper segmentation due to the image captured environment and processing it.

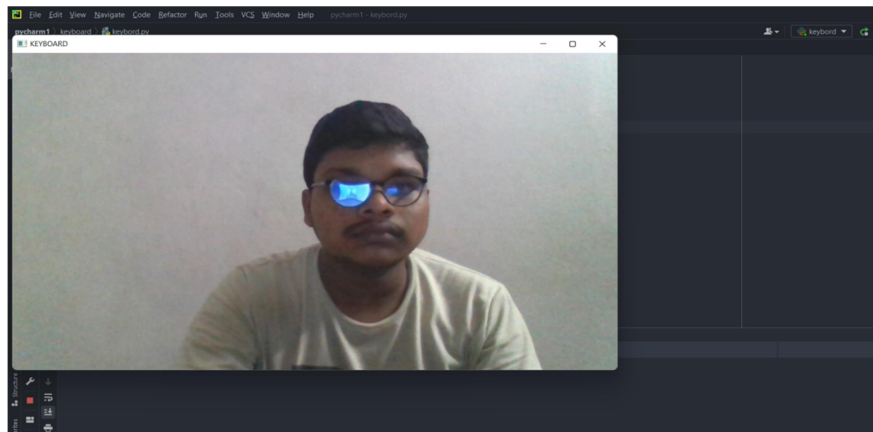


#### IV. Proposed Algorithm:

1. The execution start from the video capturing using the cv2 module by setting up set the video capturing scene dimensions.

```
import cv2
cap = cv2.VideoCapture(0)
cap.set(3, 1000)
cap.set(4, 450)

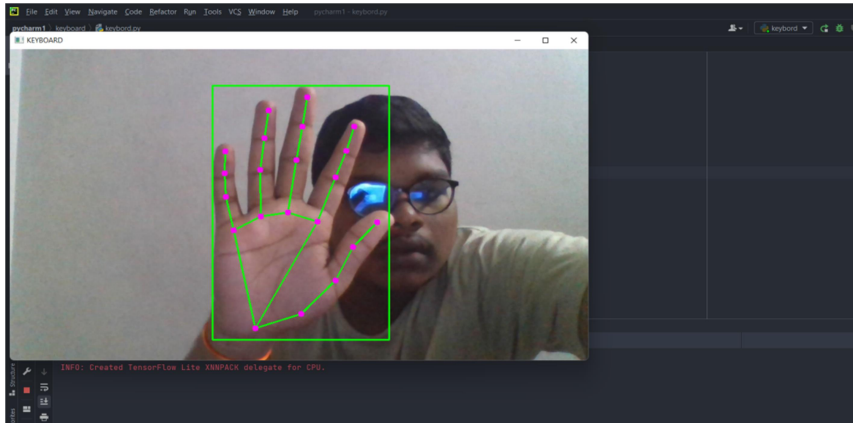
while True:
    success, img = cap.read()
    cv2.imshow("KEYBOARD", img)
    cv2.waitKey(1)
```



2. Then we start the hand detecting by importing the hand detecting module from cvzone.

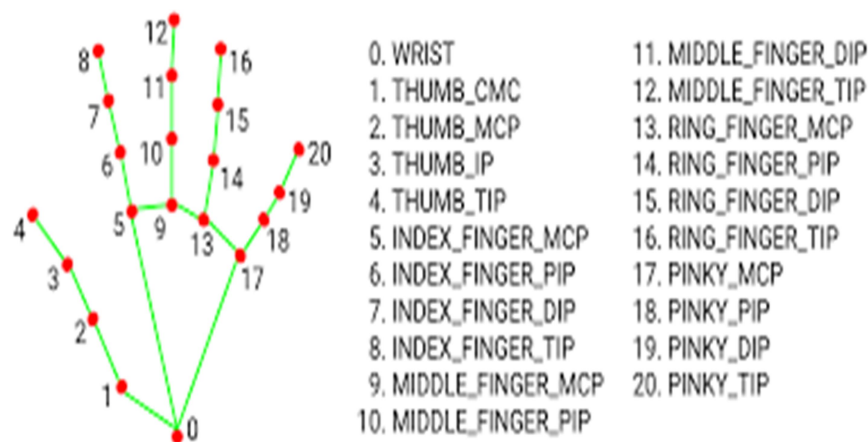
```
from cvzone.HandTrackingModule import HandDetector
detector = HandDetector(detectionCon=0.8)
```

Then, it start detecting of our hands and we can also see the land marks of our hand on video capturing.



Based on the landmarks of the hand we can capture the tap on the keyboard keys to get the text as output.

Here, the configuration of land marks on the hand, by look into this we will perform the further tasks. This is the one of the main part of the project.



3. Now, we can start the designing of our keyboard and keys as images initially, later on we will make those image keys as buttons and we will create a list which will store the letters that the keys taped by us.

We also do the designing part of the keys which feels like a clickable button.

➤ key's and final text list creation:

```

keys = [
    ["1", "2", "3", "4", "5", "6", "7", "8", "9", "0"],
    ["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", ",", ".", "/"],
    [">", "<"]]
finalText = []

```

- Designing the key images in an organized specific manner to display as a board of keys.

```

class Button():
    def __init__(self, pos, text, size=None):
        if size is None:
            size = [50, 50]
        self.pos = pos
        self.size = size
        self.text = text

buttonList = []
for i in range(len(keys)):
    for x, key in enumerate(keys[i]):
        buttonList.append(Button([75 * x + 30, 75 * i + 30], key))

```

- Now, Customize the created keyboard keys which feels like button while tapping by providing the properties like height, width, colour, font and some required effects, for these we will use cvzone and cv2.

```

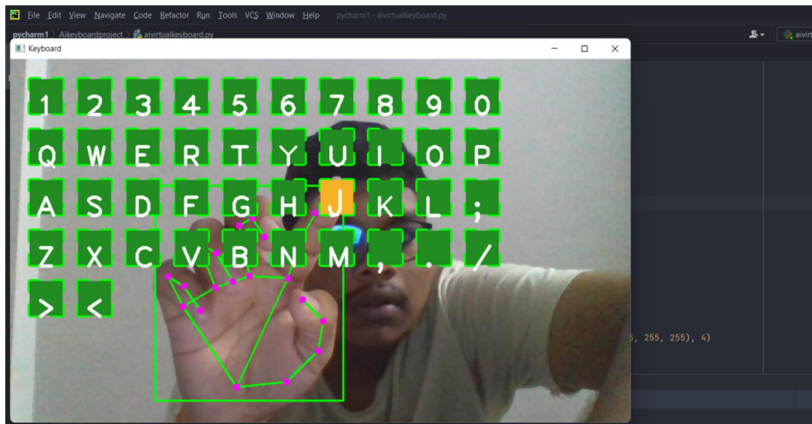
def drawAll(img, buttonList):
    for button in buttonList:
        x, y = button.pos
        w, h = button.size
        cvzone.cornerRect(img, (button.pos[0], button.pos[1],
button.size[0], button.size[1]), 15, rt=0)
        cv2.rectangle(img, button.pos, (x + w, y + h), (34, 139, 34),
cv2.FILLED)
        cv2.putText(img, button.text, (x + 10, y + 55),
cv2.FONT_HERSHEY_PLAIN, 3, (255, 255, 255), 4)
    return img

def transparent_layout(img, buttonList):
    img = np.zeros_like(img, np.uint8)
    for button in buttonList:
        x, y = button.pos
        cvzone.cornerRect(img, (button.pos[0], button.pos[1],
button.size[0], button.size[0]), 15, rt=0)
        cv2.rectangle(img, button.pos, (x + button.size[0], y +
button.size[1]), (34, 139, 34), cv2.FILLED)
        cv2.putText(img, button.text, (x + 10, y + 55),

```

```
cv2.FONT_HERSHEY_PLAIN, 4, (0, 0, 0), 4)

out = img.copy()
alpaha = 0.5
mask = img.astype(bool)
print(mask.shape)
out[mask] = cv2.addWeighted(img, alpaha, img, 1-alpaha, 0)[mask]
return out
```



4. Now we need to store the clicked key text to show the output in some other applications like notepad, Google, etc. for that we need to import Key, Controller from one of the famous module Pynput.

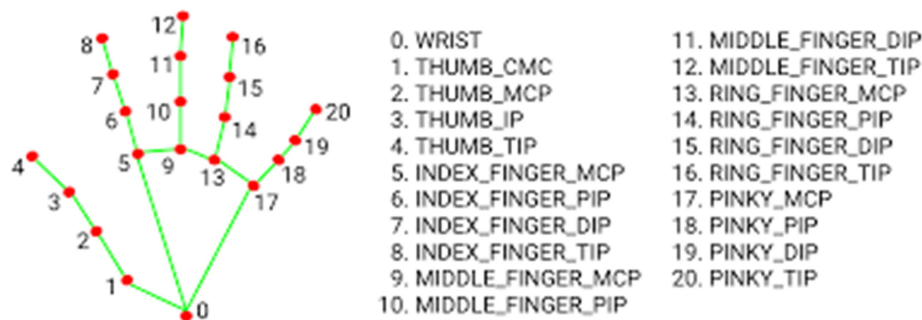
- For space bar functionality we need to code the working process to make space while on tap on the keyboard key for that we need to create an object for controller().

```
from pynput.keyboard import Key, Controller

keyboard = Controller()
keyboard.press(Key.space)
keyboard.release(Key.space)
```

- Next we need to find a way to store the text of key on keyboard, as we can do it in many ways now, I choose a way here, I can capture the text on key by enclosing between my hand detecting land marks and the Land

mark points as I shown at the starting of the algorithm and such as,



- Now, here I chosen the two land mark points 8 and 12 when, I enclosed the key in between those land marks from my hand the letter associated to the key will be appended to the list and printed as output. For, that we need do code to capture the letter when it comes in between the assigned land marks on the hand.
- We also need to code for the distance upto where the camera to detect land marks and it will captured .Here, I given the distance as less than the 60 cm.
- Then, we need to code the next part how to store the Text of the key, when all the conditions like distance and finger land marks distance are satisfied and also need to do for the working of space button in while loop where, the all the complete main process was going on like capturing video , displaying keys etc.
- Sleep() is used to give the time space to capture the enclosed key on keyboard. Here I given the 0.25 sec as time span for the sleep to capture the key.

```

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, button.pos, (x + w, y + h), (38, 178, 247),
cv2.FILLED)
            cv2.putText(img, button.text, (x + 10, y + 55),
cv2.FONT_HERSHEY_PLAIN, 4, (255, 255, 255), 4)
            l, _, _ = detector.findDistance(8, 12, img, draw=False)
            print(l)

            if l < 60:
                if "<" in button.text:
                    finalText.pop()
                    keyboard.press(Key.backspace)
                if ">" in button.text:
                    finalText.append(" ")
                    keyboard.press(Key.space)
                else:
                    keyboard.press(button.text)
                    finalText.append(button.text)
            sleep(0.25)

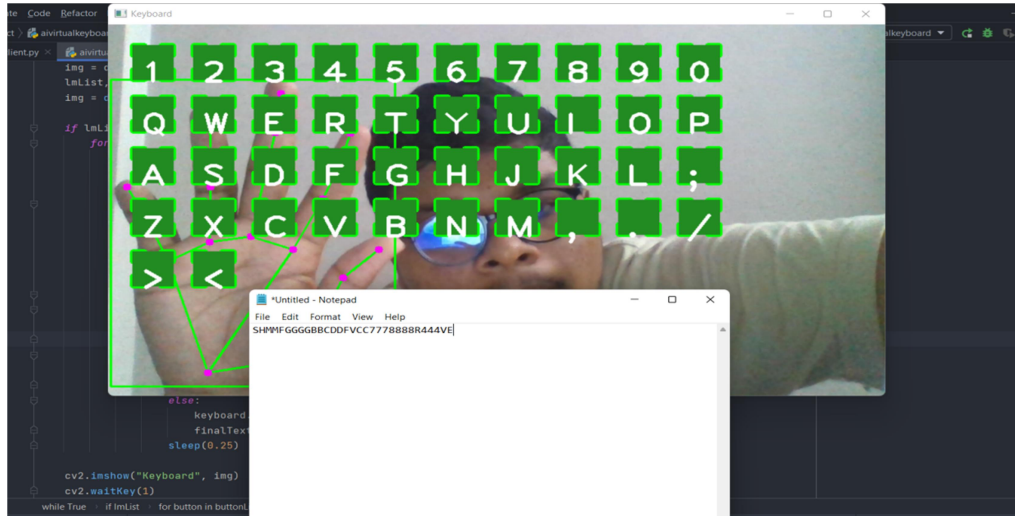
```

## V. EXPERIMENTS RESULTS

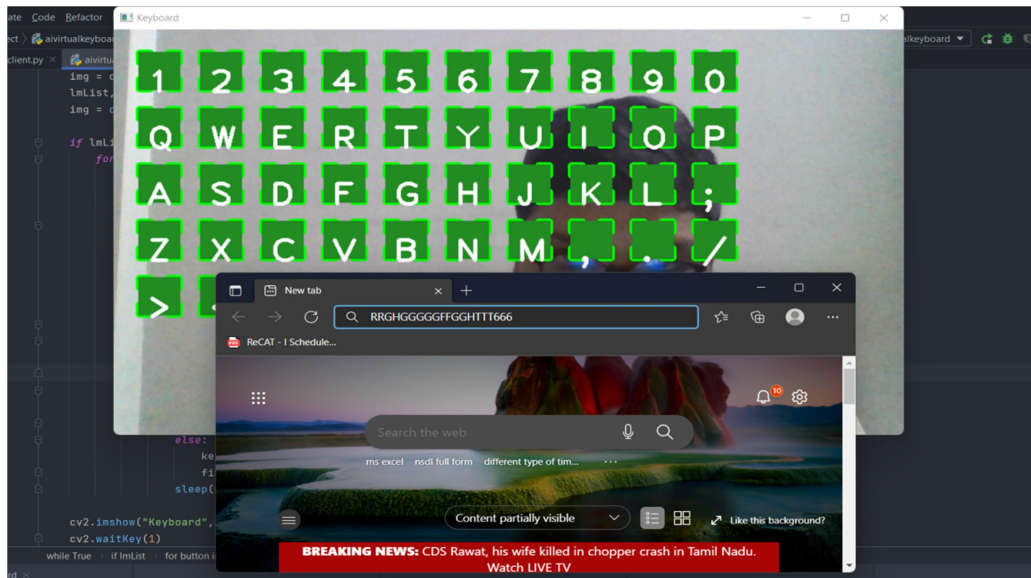
Here we can see the experimental results as per my given algorithm it shows the positive output. But, due the some user system draw backs the flexibility of video capturing and system flexibility to handle thus all the stuff was not support to give the accurate result however the system Virtual keyboard working is good as we can get the output.

## Result:

- Here, I used notepad to take the output from the keyboard.



- Then , I used Microsoft edge to take the output.



## Complete code flow for the proposed algorithm:

```
import cv2
import cvzone
import numpy as np
from cvzone.HandTrackingModule import HandDetector
from time import sleep
from pynput.keyboard import Key, Controller

cap = cv2.VideoCapture(0)
cap.set(3, 1000)
cap.set(4, 450)

detector = HandDetector(detectionCon=0.8)
keys = [
    ["1", "2", "3", "4", "5", "6", "7", "8", "9", "0"],
    ["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", ",", ".", "/"],
    [">", "<"]]
finalText = []

keyboard = Controller()
keyboard.press(Key.space)
keyboard.release(Key.space)

keyboard.press(Key.backspace)
keyboard.release(Key.backspace)

def drawAll(img, buttonlist):
    for button in buttonList:
        x, y = button.pos
        w, h = button.size
        cvzone.cornerRect(img, (button.pos[0], button.pos[1],
button.size[0], button.size[1]), 15, rt=0)
        cv2.rectangle(img, button.pos, (x + w, y + h), (34, 139, 34),
cv2.FILLED)
        cv2.putText(img, button.text, (x + 10, y + 55),
cv2.FONT_HERSHEY_PLAIN, 3, (255, 255, 255), 4)
    return img

def transparent_layout(img, buttonList):
    img = np.zeros_like(img, np.uint8)
    for button in buttonList:
        x, y = button.pos
        cvzone.cornerRect(img, (button.pos[0], button.pos[1],
button.size[0], button.size[0]), 15, rt=0)
        cv2.rectangle(img, button.pos, (x + button.size[0], y +
button.size[1]), (34, 139, 34), cv2.FILLED)
        cv2.putText(img, button.text, (x + 10, y + 55),
cv2.FONT_HERSHEY_PLAIN, 4, (0, 0, 0), 4)

    out = img.copy()
    alpha = 0.5
    mask = img.astype(bool)
    print(mask.shape)
    out[mask] = cv2.addWeighted(img, alpha, img, 1-alpha, 0)[mask]
```



```

        return out

class Button():
    def __init__(self, pos, text, size=None):
        if size is None:
            size = [50, 50]
        self.pos = pos
        self.size = size
        self.text = text

buttonList = []
for i in range(len(keys)):
    for x, key in enumerate(keys[i]):
        buttonList.append(Button([75 * x + 30, 75 * i + 30], key))

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

    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, button.pos, (x + w, y + h), (38, 178,
247), cv2.FILLED)
                cv2.putText(img, button.text, (x + 10, y + 55),
cv2.FONT_HERSHEY_PLAIN, 4, (255, 255, 255), 4)
                l, _, _ = detector.findDistance(8, 12, img, draw=False)
                print(l)

                if l < 60:
                    if "<" in button.text:
                        finalText.pop()
                        keyboard.press(Key.backspace)
                    if ">" in button.text:
                        finalText.append(" ")
                        keyboard.press(Key.space)
                    else:
                        keyboard.press(button.text)
                        finalText.append(button.text)
                sleep(0.25)

cv2.imshow("Keyboard", img)
cv2.waitKey(1)

```

## **VI. COMPARATIVE STUDY**

- While comparing with Virtual keyboard which works on the colour detection processing the hand tracking is very easy to implement. It enhance the better feasibility while comparing to colour processing like, we can't hide the colour for each letter to letter taping but, we can do more better than that by, operating between the finger land marks by changing the distance.
- By comparing the physical keyboard we can implement this type of virtual keyboard which gone be work on hand tracking, colour, eye gauze, etc. It may help a lot for disable people and personally uncomfortable people who are not comfortable to work with physical keyboards and touch screen mechanism also very different from these technique.

## **VII. CONCLUSION AND FUTURE WORK**

The world is changing a lot to work on technologies like AI, ML DL ,IOT etc. By, using these kind of super intelligence technology we can make a good virtual keyboards and virtual appliances by integrating these technologies we can live a world of virtual technology and also with ease of good cost which can able to avail for a common men . So, implementing these type of ideas may help full for many people how want to overcome their disability to work. I will look forward for further developments for further extension by integrating technologies like IOT by learning all those requirements.

## VIII. REFERENCES

1. L.Yun, Z. Lifeng, Z. Shujun, "A Hand Gesture Recognition Method Based on Multi- Feature Fusion and Template Matching", Procedia Engineering, Vol. 29, pp 1678-1684, 2012
2. [https://www.researchgate.net/publication/342852779\\_Finger\\_Recognition\\_and\\_Gesture\\_based\\_Virtual\\_Keyboard](https://www.researchgate.net/publication/342852779_Finger_Recognition_and_Gesture_based_Virtual_Keyboard)
3. <https://ai.googleblog.com/2019/08/on-device-real-time-hand-tracking-with.html>.
4. R. Zaman, K. Noor, A. Ibraheem, "Hand Gesture Recognition:A Literature Review", International Journal of Artificial Intelligence & Applications (IJAIA), Vol.3, No.4, July 2012.
5. <https://openrepository.aut.ac.nz/bitstream/handle/10292/9928/ZhangY.pdf?sequence=3&isAllowed=y>.
6. L.Yun, Z. Lifeng, Z. Shujun, "A Hand Gesture Recognition Method Based on Multi- Feature Fusion and Template Matching", Procedia Engineering, Vol. 29, pp 1678-1684, 2012
7. F.S. Chen , C.M . Fu, C.L. Huang, "Hand gesture recognition using a real- time tracking method and hidden Markov models", Image and Vision Computing , vol. 21, issue 8, pp 745-758, 2003