Non - Touch Virtual Keyboard using OpenCV

Overview of the project

We have developed a virtual keyboard using computer vision and hand tracking techniques. Inorder to interact with the virtual keyboard, the user has to move their hand over the keys and make specific gestures to type a key in the textbox.
The application uses a webcam to take video feed as input and uses computer vision to detect the hand and fingers of the user. The output is the characters typed by the user in the textbox displayed on the screen.

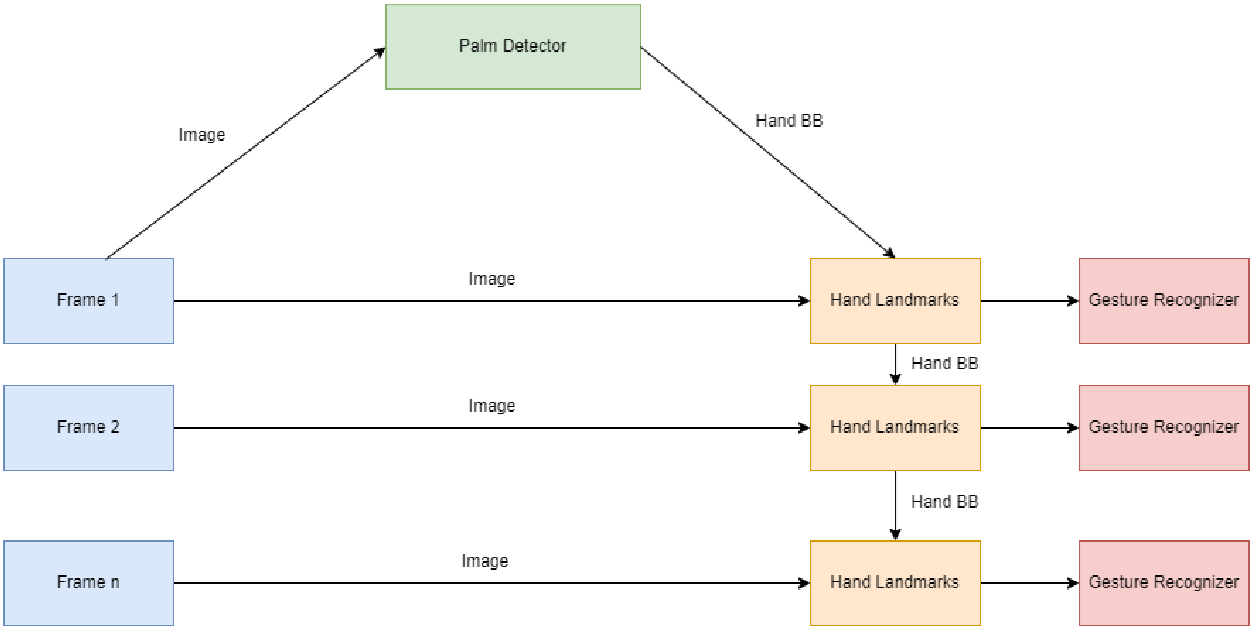
In computer vision, hand tracking and gesture recognition are well-established areas that have seen extensive use in a range of applications, including gaming, robotics, and human-computer interaction. To increase the precision and reliability of these procedures, research is still being done.

Our contribution in this project is to implement the virtual keyboard using opencv and cvzone libraries while integrating hand tracking and gesture recognition to detect the input provided by the user.

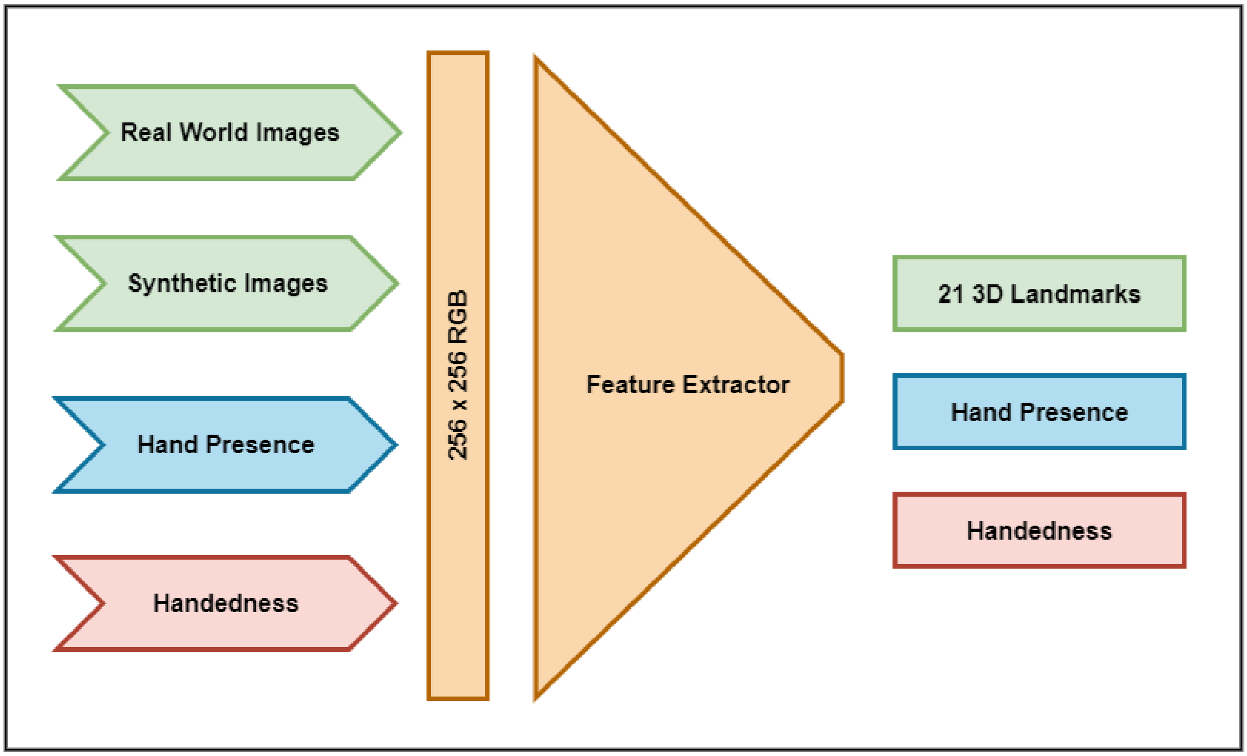
• Approach

We have written a code that is implementing a virtual keyboard using hand tracking techniques. The video feed is captured from the camera by using OpenCV and overlay the virtual keyboard on the captured image. The HandTrackingModule is used to detect hands in the captured image and to track the fingertips. The code maps the positions of the fingertips on the virtual keyboard to simulate key presses.

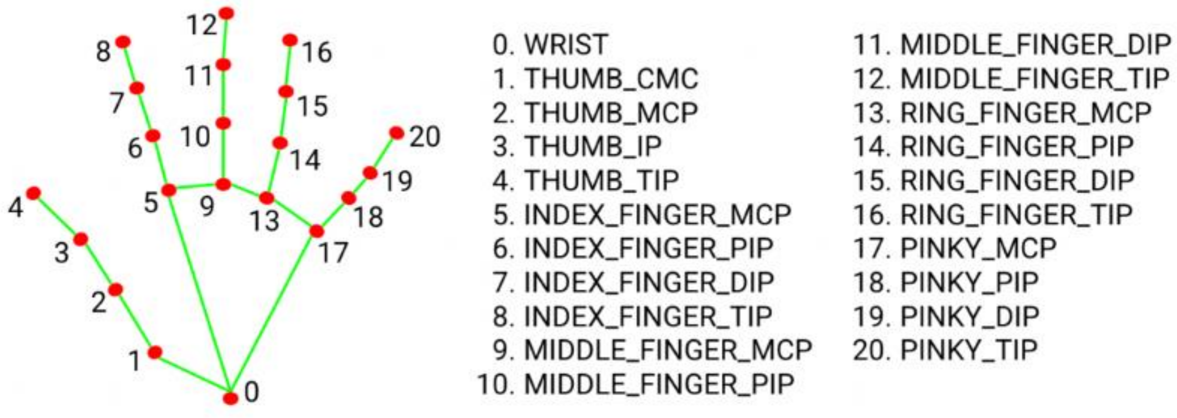
The main algorithm used in this code is the hand tracking using the HandTrackingModule. It is implemented using a pre-trained deep learning model i.e. RetinaNet for hand detection and tracking. This algorithm has already been trained on a large dataset of hand images and thus it detects hands using a high degree of accuracy.



Palm Detector Architecture



Hand Landmark Detection



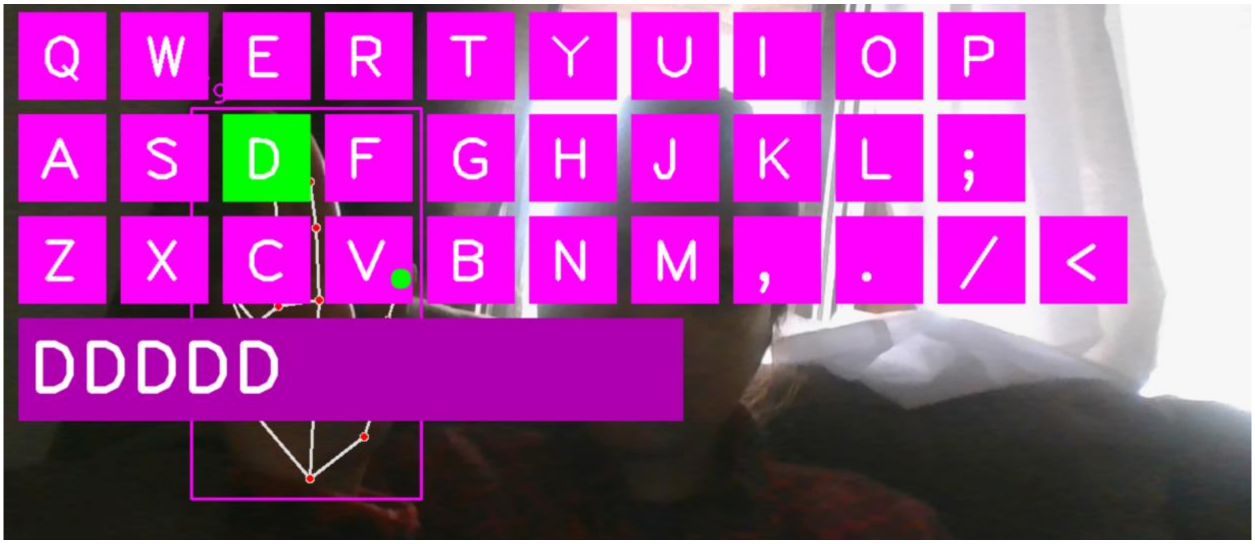
0. WRIST
1. THUMB_CMC
2. THUMB_MCP
3. THUMB_IP
4. THUMB_TIP
5. INDEX_FINGER_MCP
6. INDEX_FINGER_PIP
7. INDEX_FINGER_DIP
8. INDEX_FINGER_TIP
9. MIDDLE_FINGER_MCP
10. MIDDLE_FINGER_PIP
11. MIDDLE_FINGER_DIP
12. MIDDLE_FINGER_TIP
13. RING_FINGER_MCP
14. RING_FINGER_PIP
15. RING_FINGER_DIP
16. RING_FINGER_TIP
17. PINKY_MCP
18. PINKY_PIP
19. PINKY_DIP
20. PINKY_TIP

Image Source - https://developers.google.com/mediapipe/solutions/vision/hand_landmarker

The advantages of using this algorithm is that it is highly accurate and it can detect hands in a variety of lighting conditions and hand poses. Also, it allows real-time tracking of hand movements which is essential for the virtual keyboard application.
The disadvantages of the algorithm is that it requires a very powerful computer to run in real-time and it might not work properly in a very noisy environment.

The code also uses image processing techniques to draw the virtual keyboard on the captured image and detect when a key is pressed. The algorithm sets the position of keys on the keyboard and determines if the fingertips are within the bounds of the particular key. If a fingertip is within the bounds of a key and the distance between the fingertip and a certain point on the hand is below a certain threshold, the key is considered pressed.

The advantages of image processing techniques are that they are simple to implement. They are fast and do not require a lot of computational power.
The disadvantages is that they are less accurate than many other techniques and may not work in complex environments.



DDDDD

– What aspects of the algorithms have you coded on your own?

The following aspects of the algorithm are coded by us:

- ● The creation of Buttons class that defines the attributes and methods of each button on the virtual keyboard.
- ● The generation of a list of Buttons objects using the virtual keyboard's key definitions found in the dictionary qwerty_layout.
- ● The implementation of the display_buttons() method, which when a hand gesture is detected, draws the buttons on the screen and modifies the text of chosen buttons.
- ● The implementation of the hand gesture detection and recognition using the handTracking module and the findHands() method of the HandDetector class from the cvzone.
- ● Implemented the detection of selected button based on the position of the user's fingertips and button's position on the screen.
- ● The updation of the textbox_string variable with the selected characters to create a virtual text input field.

Our code contributes to the project by creating a functional virtual keyboard that can be controlled using hand gestures. Thus it provides an interactive and accessible user experience. The code also demonstrate the use of computer vision techniques such as hand detection recognition.

• Experimental Protocol

In our code, we did not use any pre-existing datasets as we are using the webcam feed as our input. To evaluate the success of the project, we have used qualitative measures. We inspect the output of the program visually to ensure that the virtual keyboard was correctly mapped to the

user's hand movements. We also checked that the keystrokes were accurately detected and displayed on the screen.
For computing resources, we required a computer with a webcam and the necessary libraries installed, including OpenCV, Numpy and the cvzone hand tracking module. No cloud computing resources were required as the code was run locally.

• Results

Qualitative result -

Comparison between OpenCV-based virtual keyboard and AR/VR-based virtual keyboard:
OpenCV-based virtual keyboard:

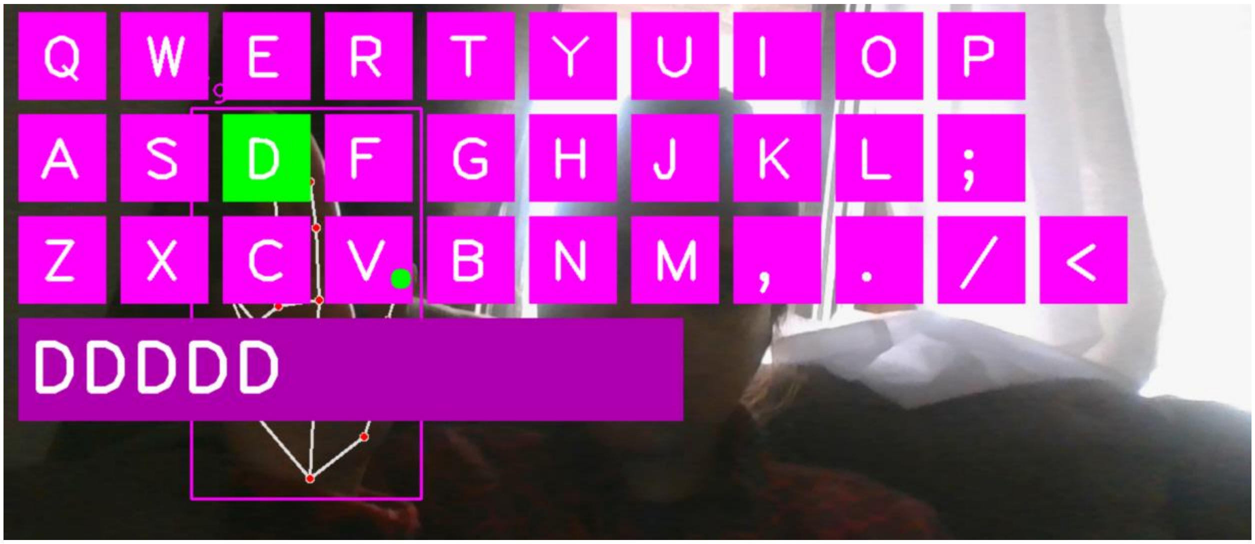
- It tracks hand motions and recognizes finger gestures on a virtual keyboard that is presented on the screen using computer vision and image processing techniques.
- It is simple to set up and use since it only needs a camera and no additional hardware or sensors.

Any device having a camera and a screen, such as smartphones, tablets, laptops, and desktop PCs, can use it.

- It enables text input without a physical keyboard by being integrated into other programs and applications.

- It is constrained by the camera's accuracy and dependability as well as the computer vision algorithms, which are susceptible to changes in lighting, background clutter, and occlusions.

AR/VR-based virtual keyboard:



- It displays a virtual keyboard in a real-world or simulated environment using augmented reality or virtual reality technologies.
- It necessitates specialized gear and sensors, including cameras, motion controllers, and AR/VR headsets, which can be pricey and call for expert software development abilities.

- Because users can view and interact with the virtual keyboard in 3D space, it can offer a more immersive and interactive experience than OpenCV-based virtual keyboards.
- Beyond text input, it can be used for a wide range of purposes, including gaming, education, and training.

- It is constrained by the accessibility and cost of AR/VR technology as well as the device's processing speed and graphical capabilities.

Interpreting and presenting results -
The screen accurately identifies the hand and position of the hand on the screen. It accurately identifies the gesture and identifies and presses the desired button. The text gets typed in the provided textbox.

In conclusion, OpenCV-based virtual keyboards are easier to use and more widely accessible, whereas AR/VR-based virtual keyboards provide more sophisticated functionality and interactivity but necessitate more specialized gear and expertise.

• **Analysis**

The algorithm uses hand tracking and computer vision techniques to track the movements of the user’s hand and detect when a button on a virtual keyboard is pressed.

Advantages:

- ● It can detect when a finger comes in touch with a button on the virtual keyboard and precisely track a user's hand movements.

- ● The virtual keyboard is adaptable for a variety of applications since it is simple to change

  to any desired layout or button size.

- ● The user receives crystal-clear visual feedback thanks to the implementation of a

  color-coded feedback system for button pushes.

Limitations:

● The technology being used, notably the camera quality and frame rate, has limitations on

the algorithm. The precision of the hand tracking may suffer from a poor camera or a slow frame rate.

- ● The environment's lighting limitations also have an impact on the algorithm. The program can struggle to precisely track the hand movements in extreme lighting conditions.
- ● In situations where numerous hands are in the frame or if the user is using both hands at once, the algorithm, which presumes that only one hand is being used at a time, may not perform as effectively.
- ● Beyond button pushes, the system is unable to distinguish specific hand gestures or motions.
- ● **Summary**

This project taught us how to create a hand-gesture-controlled virtual keyboard using OpenCV and the cvzone HandTrackingModule. It also showed us how to use conditional statements to handle events like button clicks and how to create objects in Python using classes.
In the future, we intend to construct more sophisticated computer vision apps using this project as a base. We learned how to use hand tracking and object detection, which may be used for a variety of tasks, including hand gesture control of robots and the understanding of sign language. We also learned the value of user interface design and how to make a straightforward yet user-friendly interface.

• **Bibliography**

- ● Cvzone Hand Tracking Module - https://github.com/cvzone/handTrackingModule
- ● Qwerty keyboard - https://en.wikipedia.org/wiki/QWERTY
- ● OpenCV - https://opencv.org/
- ● Python math - https://docs.python.org/3/library/math.html
- ● Numpy - https://numpy.org/
