COMP4801 Final Year Project [2023-24]

Interim Report

Virtual Keyboard App Using Computer Vision

Supervisor: Dr Pan Jia (jpan@cs.hku.hk)

Student: Mak Tsz Shing (3035685914) (u3568591@connect.hku.hk)

Data of Submission: 21 January 2024

Abstract

Texting on mobile devices is usually achieved by tapping both thumbs on the built-in on-screen virtual keyboard. However, the small form factor of mobile devices has made typing difficult. Therefore, there is need for intuitive, flexible, and efficient alternative text input method for mobile devices. In this project, a virtual keyboard Android app using computer vision technique will be developed to allow users to type like typing on a physical keyboard. The front camera of the mobile device will be utilized to track users' finger gestures of typing on a flat surface in front of the mobile device. The app will be developed in Android Studio using Kotlin as the programming language. The Hand landmark detection solution from MediaPipe will be used as the machine learning solution for hand landmark detection. Several challenges and limitations are identified, such as accuracy of tap detection and camera orientation. This project is currently on schedule. A demo application with the hand detection and tap detection using basic algorithm is developed. The next phrase will be developing the actual keyboard application. It is expected that this project will be beneficial to virtualization of other controllers using computer vision techniques.

Acknowledgement

I would also like to express gratitude to Dr. Pan Jia for supervising this final year project and providing advice.

Table of Contents

List of Figures

Abbreviations

CV	Computer Vision
ML	Machine Learning
UI	User Interface

1. Introduction

1.1. Background and motivation

The emergence of mobile devices has revolutionized the way we communicate, work and access information. As mobile networking and microprocessor technologies progress, mobile devices such as smartphones are becoming powerful enough to replace desktop computers [1]. In 2019, mobile users accounted for 53% of all web traffic [2]. People are more likely to use a smartphone instead of a desktop computer to perform tasks like online shopping, video streaming and browsing social media sites. Moreover, there is an growing trend of people using their smartphones for work-related tasks, such as emailing, calling and appointment scheduling [3].

Despite the convenience offered by mobile devices, their compact form factor makes text typing difficult. For instance, texting on smartphones usually involves tapping on the virtual on-screen keyboard with both thumbs. In contrast to a standard physical keyboard that enables users to type with all ten fingers, a virtual on-screen keyboard has considerably smaller keys that makes it hard for users to use multiple fingers for typing. The small keystroke size on an on-screen keyboard would also reduce the typing speed and increase the chance of making a typo. To improve the typing experience on mobile devices, new features like autocorrect, SwiftKey [4], and speech-to-text input are currently added to mobile virtual keyboard apps. These features allow users to type on mobile devices without tapping on individual keys on a virtual on-screen keyboard and reduces typing errors.

Inspired by the new features on mobile virtual keyboard apps, it is believed that alternative text input methods are needed on mobile devices. The new text input method should overcome the limitation of the small form factor and provide good user experience. With the increase in computation power and resolution on mobile devices, we are motivated to explore an innovative solution by leveraging computer vision and machine learning techniques. We seek to provide users with a more natural and intuitive way to interact with their mobile devices, ultimately enhancing their typing experience and overall satisfaction.

1.2. Objectives and deliverables

This paper introduces an Android virtual keyboard app that provides user with similar typing experiences on a standard keyboard. Computer Vision techniques are leveraged to track hand movements and gestures. Instead of tapping on the screen of mobile devices, users can type by tapping on any flat surfaces as tapping on a real keyboard in front of the device camera. The virtual keyboard app utilizes the device camera to track users' finger movements and gestures to determine the text input.

1.3. Report content

The remainder of this paper proceeds as follows. Section 2 explains the methodologies for developing a vision-based virtual keyboard app. Section 3 reports current progress and the proposed schedule. Section 4 discusses the challenges and limitations in developing the app. Section 5 summarizes the report.

2. Methodologies

This section introduces the methodologies of the project. Section 2.1 introduces the IDEs and programming languages used. Section 2.2 introduces the library used to capture user finger motions using the front camera. Section 2.3 – 2.5 introduces the 3 main stages to transforming typing actions into text input, including hand detection (section 2.3), tap detection (section 2.4), and key input determination (section 2.5).

2.1. IDEs and programming languages

The app is developed using Android Studio. It is the official IDE developed by Google particularly for development of Android applications [7]. Comparing to other IDEs, Android Studio offers Graphical UI for drag-and-drop components on the app interface.

Kotlin is used as the programming language for the app. It is a modern programming language used by over 60% of professional Android developers [8]. In addition to its compatibility with Java, Kotlin has a more concise syntax than Java. With the great community support, developing the app using Kotlin would be quicker and easier.

2.2. Image Capture

The typing actions of user fingers are captured using the front camera of users' mobile devices. CameraX, a jetpack library developed by Google, is used to manage the camera usage of the app, including frame capturing, frame previewing and frame processing in background. CameraX also maintains the consistency of camera behavior across various devices with different Android versions (Android 5.0+).

2.3. Hand detection

Hand detection is done using the Hand landmark detection solution provided by MediaPipe. MediaPipe is an open-sourced framework developed by Google Research for building custom on-device ML solutions [5]. It abstracts the complexity of running on-device ML solutions, such as CPU/GPU acceleration and deploying the ML solutions on different platforms. It also supports well-known programming languages such as JavaScript, Python and C++, and running on mobile environment such as iOS and Android [5].

Mediapipe provides 14 pretrained solutions of common ML use cases, such as hand gesture recognition and face detection. The hand landmark detection solution will be used for hand detection in this project. The solution detects the existence of hand in static or continuous stream of image data. If one or more hands exists, the model will return the handedness (left or right hand) and the world coordinates of the 21 hand knuckles (landmarks) on the hand in real time (see Figure 1 for details of the landmark). Figure 2 shows a example of using the land landmark detection solution in actual.

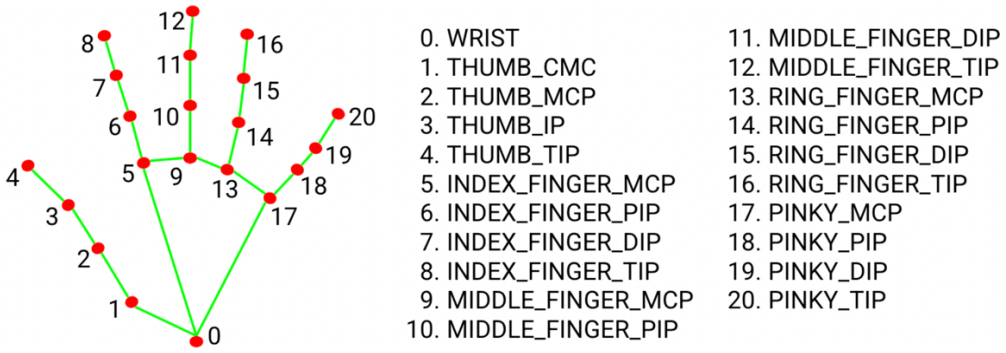

Figure 1. 21 hand landmarks detected by MediaPipe hand landmark detection solution [6]. Coordinates of the landmarks of user's hand on the captured image frame can be retrieved.

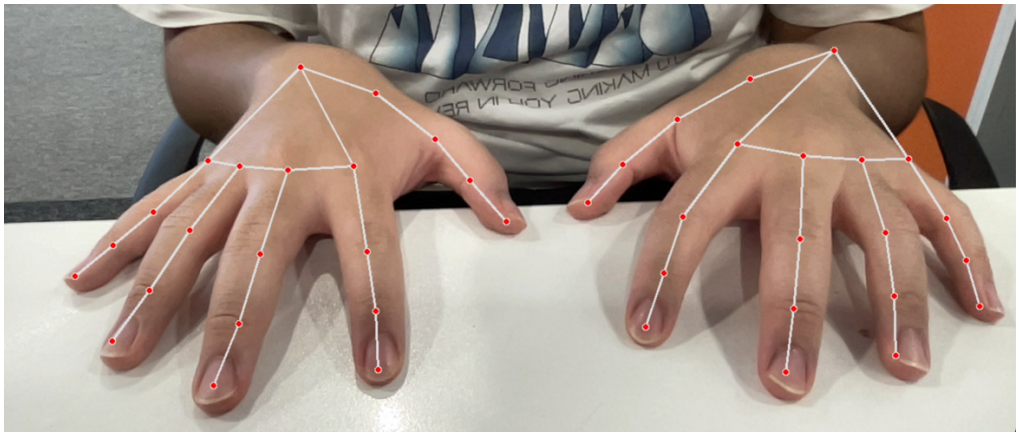

Figure 2. Actual example of using MediaPipe Hand Landmarker. The key points on both hands are shown in red dots.

There are several advantages for utilizing the hand landmark detection solution by MediaPipe as the hand detection method in this project. First, the model is trained on approximately 30K real-world images [6]. The effort of developing and training a custom ML model for hand recognition can be eliminated. Second, the latency of the model is acceptable for typing. According to the benchmark test done by Google on Pixel 6, the latency of the ML model is 17.12ms and 12.27ms on CPU and GPU acceleration respectively [6].

2.4. Tap detection

Tap detection is to detect whether the user has pressed a key on the virtual keyboard or not. Since there are no sensors installed on the tapping surface, the tap detection can only be determined based on the finger joint movements when a user press a key.

The current algorithm of tap detection is tracking the vertical motion of fingertips. A finger is considered as tapping when its fingertip keeps moving down towards the tapping surface and then moves up. The motion tracking is done by recording the coordinates of fingertips returned by the hand landmark detection model over a period of time. To eliminate errors, the downward displacement of fingertips must exceed a certain threshold.

2.5. Key input determination

Key input determination is done with reference to the coordinates of hand landmarks. Before typing with the app, users need to place their hand in a reference position. For instance, users need to place their left index finger on the “F” key and their right index finger on the “J” key. The coordinates of these position are used as reference for other keys on the virtual keyboard. A set of coordinates are rendered to map to other keys on the virtual keyboard.

After a tap is detected, the coordinates of fingertips (landmark number 4, 8, 12, 16 and 20 on Figure 1) are retrieved by the hand Landmarker. The key input is determined by comparing the fingertips coordinates with the reference coordinates rendered in previous step.

To generate a key input as a keyboard, the Android TextService API will be utilized. The input method service will be used to generate key input on the input field, and the spell checker framework will be used to enhance typing experience by providing predictions of word input.

3. Current Progress and Proposed Schedule

This section reports the current progress and the proposed schedule of the project.

3.1. Current Progress

The project has proceeded on target as proposed on the project plan. Studies on image processing techniques using OpenCV, Android app development using Android Studio and MediaPipe libraries implementation in the Android platform are conducted in the first semester.

A demo application with part of the features is developed after the studies. The demo application introduces the hand landmark detection solution from MediaPipe and perform basic tap detection using the current algorithm.

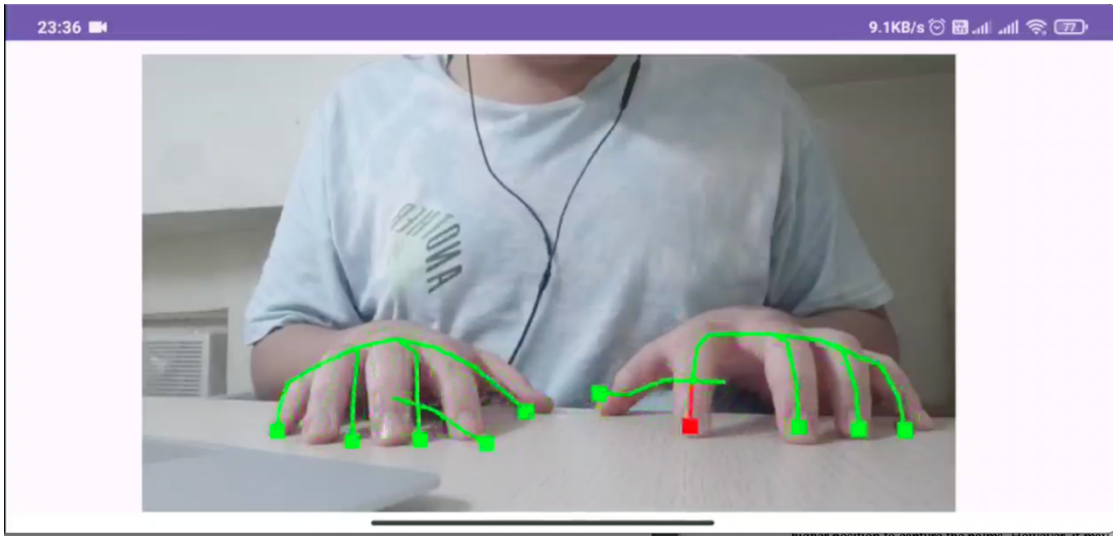

Figure 3. Demo Application Screenshot. A red line and dot are drawn on the right index finger when it is tapping

3.2. Proposed Schedule

The immediate next step is to finish the demo application by adding more features described in Section 2. It is expected that the actual virtual keyboard application will be developed from January 2024 to the end of March 2024 with reference to the demo application. Optimization and testing of the app will be carried in early April 2024. Phase 3 is expected to be delivered before 23 April 2024 with the finalized virtual keyboard app and the final report.

4. Challenges and Limitations

This section introduces some challenges and limitations encountered during the development of the app.

4.1. Tap Detection Algorithm

As described in Section 2.4, the current tap detection algorithm only involves the vertical motion of fingertips. The algorithm does not determine whether the finger has tapped on the tapping surface. False tapping may be detected when user moves the hands in the air.

The current solution is to study the relations between fingertips and the tapping surface when tapping. Possible methods including edge detection of fingers and the tapping surface using image processing techniques. Tests will be conducted in finding the intersection point between the fingertips and the tapping surface.

4.2. Performance of the Hand landmark detection model

As described in Section 2.3, the hand landmark detection model is already well-trained with more than 30K real world images. However, the performance of the model fluctuates when testing with the demo application. One possible reason is that the palms are usually not visible to the camera when a user is typing. Only the fingers are visible to the camera. Therefore, the model may have lower confidence of the landmark coordinates compared to open palm.

One solution is to adjust the camera orientation. The camera angle can be set to a higher position to capture the palms. However, it may affect the accuracy of the tap detection algorithm. Also, the adjusted camera orientation may affect users' sight to the screen of the device. If it is unfeasible, alternatives will be stabilizing the coordinates of hand landmarks by filtering out the frames with low confidence of hand detection.

5. Conclusion

This paper has presented a method to solve the difficulty of texting on small mobile devices by developing a virtual keyboard app using CV on the Android platform. The app generates text input by analyzing users' finger movements captured by the front camera on their mobile devices. It provides an intuitive and convenient text input method on mobile devices.

An important avenue for future work is to develop a customizable CV-based virtual controller app. Limited by the visible range of the front camera of mobile devices, the virtual keyboard app developed in this project has limited number of virtual keys. If a camera with wider visible area is used, there will be more room for users to customize their own virtual keyboard. For instance, users can add self-defined macro keys or change the key layout of the virtual keyboard. This will enhance user experience and their efficiency in typing on the virtual keyboard.

Another possibility for future work will be adding more recognizable hand gestures to the virtual keyboard app. Besides tapping on keys, users may also choose to control their mobile device with other hand gestures like swiping their hands in air or swiping their hands on the tapping surface. This also helps to increase the flexibility of the virtual keyboard app and provide better user experience.

6. References

[1] *"A Computer In Your Pocket: The Rise of Smartphones"*. Science Museum UK. Accessed: Jan 21, 2024 [Online]. Available: https://www.sciencemuseum.org.uk/objects-and-stories/computer-your-pocket-rise-smartphones

[2] I. Bouchrika. *"Mobile vs Desktop Usage Statistics for 2023"*. Research.com. Accessed: Jan 21, 2024 [Online]. Available: https://research.com/software/mobile-vs-desktop-usage

[3] R. Lay and A. Stanford. "*More and more people are using their smartphones for work".* Deloitte. Accessed: Jan 21, 2024 [Online]. Available: https://www2.deloitte.com/ch/en/pages/technology-media-and-telecommunications/articles/immer-mehr-menschen-arbeiten-auf-dem-smartphone.html

[4] *"Microsoft SwiftKey Keyboard"*. Microsoft. Accessed: Jan 21, 2024 [Online]. Available: https://www.microsoft.com/en-us/swiftkey?activetab=pivot_1:primaryr2

[5] *"MediaPipe | Google For Developers"*. Google. Accessed: Jan 21, 2024 [Online]. Available: https://developers.google.com/mediapipe

[6] *"Hand landmarks detection guide"*. Google. Accessed: Jan 21, 2024 [Online]. Available: https://developers.google.com/mediapipe/solutions/vision/hand_landmarker

[7] *"Download Android Studio & App Tools – Android Developers"*. Google. Accessed: Jan 21, 2024 [Online]. Available: https://developer.android.com/studio

[8] *"Kotlin and Android | Android Developers".* Google. Accessed: Jan 21, 2024 [Online]. Available: https://developer.android.com/kotlin