Hands-free text-based user Interface for Android

HIMA SAGAR POTNURU, Indian Institute of Technology Gandhinagar, INDIA

ACM Reference Format:
Hima Sagar Potnuru. 2024. Hands-free text-based user Interface for Android. 1, 1 (August 2024), 6 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 Abstract

The main goal is to create an interface for Android that can be controlled without the user's hands. This UI will have multiple applications in the future, giving the user an excellent option to use his Android completely without using his hands. This will mainly help people who have disabilities.

So, to achieve the complete hands-free treatment for Android, I would like to start with designing a virtual keyboard for Android that can be controlled entirely with the user's eye/s.

2 Introduction

About 15% of people worldwide are disabled, and 80% of them reside in low-resource environments where they frequently experience extreme social isolation. People with disabilities living in low-resource settings cannot fully have their needs met by technology since it is either inaccessible or poorly constructed[1].

While eye-tracking devices and brain-computer interfaces based on EEG technology show promise in helping these people, they are very expensive and uncomfortable, which limits accessibility for people with disabilities living in low-resource environments, so we took the initiative to achieve the complete hands-free treatment for android starting with the implementation of virtual keyboard.

Also, to completely achieve the hands-free treatment, using the device's tilts and gestures to improvise our activities on devices is helpful and will make the user very comfortable to use. So, in this work, we will also see how the built-in sensors, such as accelerometer, proximity, etc., in our Android, can be used to achieve our goal. So this report consists of two parts: one is implementing the virtual keyboard, and the other is detecting tilts and gestures.

3 Methodologies

The open source library or framework I have used to implement the virtual keyboard is Mediapipe and Java as the main language of coding. I used Android Studio to build the XML layout files and, finally, bazel to build and install the application.

To detect tilts and gestures, I have used the Sensor Manager to access the hardware in my device and used Android Studio for complete implementation and testing.

Author's Contact Information: Hima Sagar Potnuru, himapotnuru@iitgn.ac.in, Indian Institute of Technology Gandhinagar, Gandhinagar, Gujarat, INDIA.

ACM XXXX-XXXX/2024/8-ART
https://doi.org/10.1145/nnnnnnn.nnnnnnn

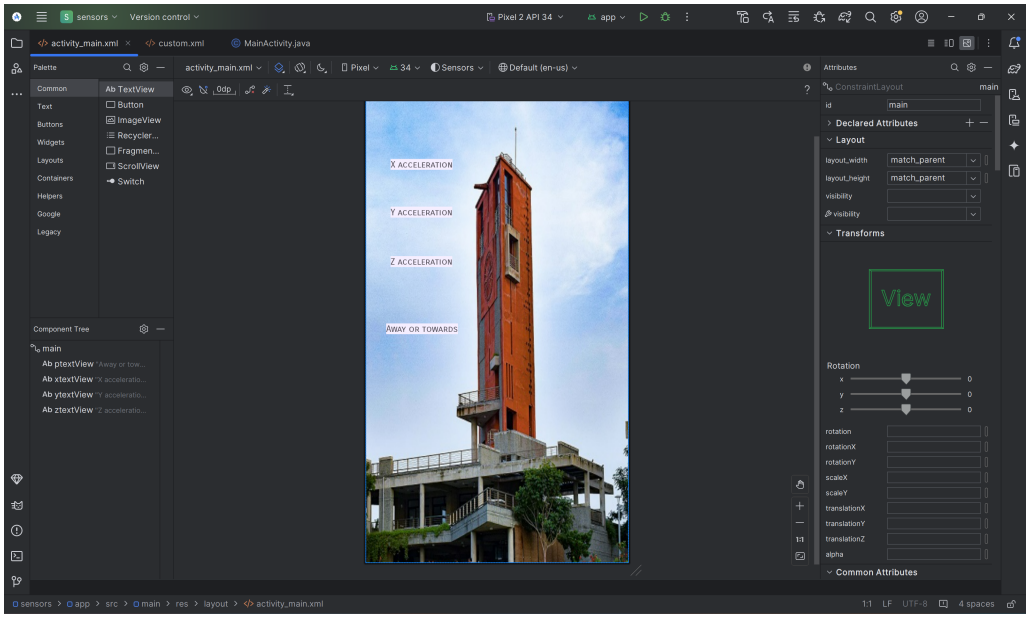

Fig. 1. Layout of Tilts and Gestures

3.1 Usage of Sensors

Tilts and gestures make the user's job easier if they have some significance, so to detect tilts such as left tilt or right tilt or some gestures such as moving the device towards the user or away from the user, I have used the Accelerometer sensor which is inbuilt in the android and can be accessible with the Sensor Manager.

An accelerometer sensor logs the device's acceleration in each of the three sensor axes. Both gravity and physical acceleration (change in velocity) are included in the measured acceleration. The sensors_event_t record contains the measurement in its x, y, and z fields[2]. You can find more information on the accelerometer sensor here.

So, with the help of Sensor Manager, accessing the acceleration in the x,y, and z fields and observing how the acceleration values change when there is a tilt either left or right or when there's a gesture either moving the device towards you or away from you and based on the changes in the values of acceleration in the fields the tilts and gestures are detected.

Here's a demo video of detecting the tilts and gestures.

Here's the Java code for the above application and Here's the layout file for the Tilts and Gestures application.

You may take a look at the layout of Tilts and Gestures looks like Fig.1.,

So, after successfully detecting tilts and the device's gestures, the main goal is to give some significance to these. So some of the ideas are as such(Future Work):

- If the user is busy reading some article or playing a game, then if he receives some unwanted notification, then performing a left tilt will clear the notification even from the notification bar.
- In the above case, if the notification is a bit important but not at the time the user received it, performing a right tilt will clear the notification from the display and will keep it in the notification bar and the user can even set a timer to view that notification later on.

- The left and right tilts can also be used to navigate between the windows or applications in the device instead of opening up the applications window and selecting which one you want to go to; these tilts will improve the user experience and also reduce time.
- If the user receives an important notification gesture like moving the device towards you should open up the notification or the application from which it is received.
- If the user performed the above task(4) then moving the device away from the user will take the user back to the previous application he was using.

NOTE: The user must click on the activate button, which activates all the above implementations, before using them; otherwise, unwanted tilts or gestures may irritate the user, and then at last, the user has to deactivate them if the user no longer wants them.

3.2 Virtual Keyboard

Our main goal is to replace the usage of our hands or fingers with something else to control our device, which makes our job easier, comfortable, and time-reducing. So, our eyes are one of the main body parts that can be easily used for a long time without much strain. Anyways, if the user is using the device, he/she uses his eyes to look at the display; introducing some more actions, such as eye blink and eye gaze, will get our job done. Let's see how.

3.2.1 Eye blink detection. Blink of an eye is a very common thing that we do every day but using our eye blink to control our device is completely fascinating.

So here are some ideas which can be implemented with the blink of an eye:

- Let's say that the user is unable to use his hands for the tilts or gestures but still wants to have the facilities mentioned in 3.1, then the user can replace the tilts with eye blink detection to have the same facilities which is if the user is busy reading some article or playing a game, then if he receives some important notification, performing an eye blink will open up the notification or the application from which it was received.
- Let's say that we managed to get the coordinates of our eyes(Achievable) where we are looking on the display, then looking at some particular application and blinking at the right time could open up the application and also without using the fingers of our hand for selection or for clicking on the application eye blinking can be used as a replacement.
- Let's say the user is on social media pages like Instagram; eye gaze can be used for scrolling and eye blink can be used for either liking it or commenting on the post or some similar type of action can be performed.

So, looking at the vast usage of eye blink, let's see how it can be detected using the media pipe framework.

Using the facemesh mediapipe framework the landmarks of the entire face can be achieved by eyeslandmarks.py.

Look at Figure.2 for all the face landmarks.

So after getting the landmarks of the entire face by close observation, the landmarks of both our eyes can be found, and using the distance between the top-most point and bottom-most point of an eye, the blink can be detected.

Here's the code for the eye blink detector.

Here's a small demo of the eye blink detection.

3.2.2 Design of virtual keyboard. The entire design of the keyboard was made using Android Studio. So the idea is to divide some part of the screen into six parts for the keys, especially the alphabets; each one consists of six different alphabets, so when the user's eyes are in the region of one of the six regions and if the user blinks then another layout will be opened which consists of six different alphabets and again if the user places his eyes in one of the six regions and blinks that letter will be selected and will be added to the Text scroll view at the top.

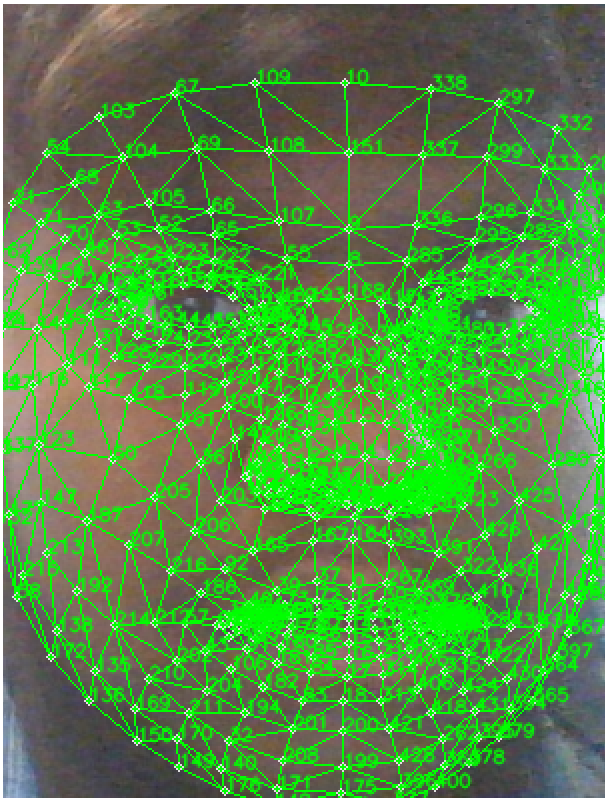

Fig. 2. Face Landmarks

The other keys and their uses are:

- `Cross Symbol`: The cross or "X" symbol can be used to go back to the parent layout file.
- `Down Arrow Symbol`: The down arrow symbol can be used to get into the new line of the text.
- `Caps Lock`: The caps lock symbol can be used whenever the user needs the letters to be in the capital.
- `Space Bar`: The space bar symbol can be used to introduce space between the alphabets.
- `Back Space`: The backspace symbol can be used to clear the alphabet individually.
- `Copy button`: The copy button can copy the entire text in the text scroll view section.

Here's the code for my layout files

Also, please look at the XML file's design in Android Studio in figure XML Design. Here's the final demo after introducing the eye blink detection into the virtual keyboard application, also find the final compiled code for the above application (NOTE: Bazel is still in developing mode, so tools like Intent, which can be used to navigate between layouts, are not working correctly. In the demo, I was unable to show the other layouts. In the future, I will ensure there won't be this kind of compromise.)

4 Future Work

- **`Navigation`**: As mentioned, due to errors in the usage of Intent, the navigation between layouts was not made, so I will figure out how to make the navigation successful.

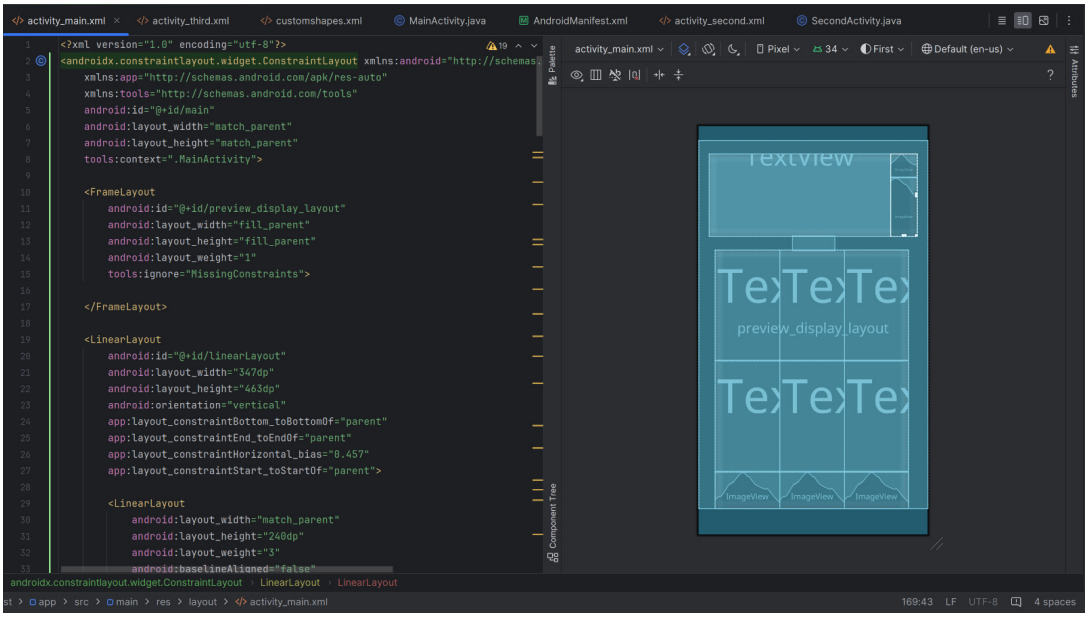

Fig. 3. XML Design

- **`Calibration`**: Currently, in the application I made, I am unable to get the exact location of where the user's eyes are focussing, so the idea is to introduce a machine learning model and train it at the beginning of activating our application and then use it to predict the location on display where the user is focussing. I tried to build a media pipe android archive model(AAR) to merge it with Android Studio and also tried to merge a machine learning model into Android Studio to work on it. But both of these attempts failed, so I will figure out a way to do all of this.
- **`Implementing tilts`**: The detection of tilts and gestures is successfully done, but no significance or not implementation is done if there's a tilt or gesture, so implementing the points made in sections 3.1 and 3.2 is one of the main features to include in future.
- **`Copy Button`**: I haven't implemented the functioning of the copy button yet, but I will do it in the future.
- **`Design`**: Changes to the layout files will be made to make the application look better.
- **`Efficiency`**: Sometimes, the device detects left and right tilts even though they shouldn't be, which leads to a decrement in accuracy. I will try to figure out those corner cases and will fix them.
- **`For IOS`**: All the implementations I specified were made for Android. The same will be implemented for IOS also.
- **`Complete Usage`**: After implementing the above points successfully, we should test whether we reached our main goal, controlling the device entirely with eye gaze and eye blink, and all the points made in this report

5 Conclusions

After detecting the tilts and eye blink successfully and implementing the virtual keyboard, I completely understood the potential of this eye gaze in the future; this will help people who are unable to use their device comfortably

with their hands, and this will be the best and cheapest solution for them, and I also believe that this eye gaze and these techniques have a lot to do in the future.

6 Acknowledgements

I sincerely thank Prof.Yogesh Kumar Meena for mentoring me throughout my internship period.

7 References

References

[1] G. Barbareschi, D. Zuleima Morgado-Ramirez, C. Holloway, S. Manohar Swaminathan, A. Vashistha, and E. Cutrell, "Disability design and innovation in low resource settings: Addressing inequality through HCI," in *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems*, 2021.

[2] "Sensor types," Android Open Source Project. [Online]. Available: https://source.android.com/docs/core/interaction/sensors/sensor-types. [Accessed: 09-Jul-2024].

[3] "Face landmark detection guide," Google for Developers. [Online]. Available: https://ai.google.dev/edge/mediapipe/solutions/vision/facelandmarker. [Accessed: 10-Jul-2024].

[4] Y. K. Meena, H. Cecotti, K. Wong-Lin, et al., "Design and evaluation of a time adaptive multimodal virtual keyboard," *Journal of Multimodal User Interfaces*, vol. 13, pp. 343–361, 2019. Available: https://doi.org/10.1007/s12193-019-00293-z.