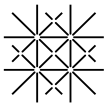

University of Basel

Word-Gesture Typing in Virtual Realty

Bachelor thesis

Databases and Information Systems
Department of Mathematics and Computer Science
Databases and Information Systems
https://dbis.dmi.unibas.ch/

Examiner: Prof. Dr. Heiko Schuldt
Supervisor: Florian Spiess

Philipp Weber
phil.weber@stud.unibas.ch
0000-000-000

20.06.2022

Acknowledgments

So Long, and Thanks for All the Fish. And the template.

Abstract

Text-entry is one of the most common forms of computer-human interaction and indispensable for many tasks such as word processing and some approaches to multimedia retrieval. The normal keyboards everybody knows have long established as the main text input method for desktop and laptop computers and even for touchscreen based devices they are very useful. But when it comes to virtual reality (VR) and augmented reality (AR), with todays technology, it lacks of tactile feedback and accurate finger tracking. As a result, text input for VR and AR is still an area of active research.
In recent years, word-gesture typing/ slide-to-type keyboards have been introduced in most major smartphone operating systems. Could this also be an efficient text input method for VR/AR?

Table of Contents

1 Introduction

1.1 Motivation

If we want to work with vitrivr-VR we have to put text into the query input text field. Right now, this is only possible with a normal keyboard, hence we have to tap on every single letter we want to write. Even though these "texts" will only contain a single word or some few words, it still might be exhausting for our arms. The keyboard in vitrivr-VR has a bigger scale than a normal keyboard in reality. We have to move quite a distance with our arms and always move up and down to not accidentally hit a key. With a word-gesture keyboard, the text input could become more comfortable. We would not need to move our arms up and down, we could just move on a flat plane from one key to another. Such a keyboard could also be smaller, because the precision is not as important as it is for the normal keyboard. For example, if we would tap in the middle of two keys, we cannot really tell which one to take for the input. But with a word-gesture keyboard, where we work with distances and graphs (more on that later), it has not that much of an impact. Therefore, a smaller keyboard is possible, and we do not have to move our arms that much.
Other applications often do also only use a normal keyboard as text input method. The word-gesture keyboard developed in this thesis will be open-source and available for everybody. Therefore, developers of other VR applications may also be using our word-gesture keyboard if they are interested.

1.2 Goals

For this thesis, we have two main goals. The first one is to develop a word-gesture keyboard. This is a keyboard, that more or less might look like a normal one. But instead of tapping on the different keys, words are written with gestures. It has to work with vitrivr-VR and has to be available as open-source. It should also be available as a Unity package, so other developers can use it in their Unity projects as well. The second goal is to evaluate said keyboard. The evaluation will be conducted according to current research standards, and we use the MacKenzie phrase set.

2

Related Work/ Background

In this chapter we introduce the environment the word-gesture keyboard is mainly developed for, some things about the normal, conventional keyboard and word-gesture keyboard in general, with an excursion about SHARK2.

2.1 vitrivr-VR and UnityVR

vitrivr[1] is an open source full stack content-based multimedia retrieval system. It supports image, audio and 3D collections and features a very broad set of query paradigms that are supported. vitrivr was developed by the Database and Information Systems group[2] (dbis) of the university of Basel. For our thesis, we use the VR part of vitrivr, namely vitrivr-VR. This is being developed in Unity[3]. Unity is a tool for developers, where one can create projects in 2D, 3D and VR. To a certain degree, Unity is free to use, but one can provide assets that are not for free. In Unity, a developer can also provide packages. These can be imported and used by other developers in their Unity projects.

2.2 Conventional Keyboard

A conventional keyboard is the most used keyboard type. On desktop and laptop computers we normally use such a keyboard. One thing that might be different in some countries is the layout. The layout does not change the functionality but can influence the appearance. This also applies to the most used keyboard for phones, tablets and other touchscreen-based devices. The only difference is that we do not press physical keys, but tap on the screen, where a certain key is. These keyboards work really well for text input with the previously mentioned devices. But when it comes to virtual reality (VR) or augmented reality (AR), this may not be the best possible text input method. Right now, it lacks of tactile feedback and accurate finger tracking. While this could be improved during the next years, yet it is not really there. Another reason is the size of such keyboards in VR. Due to the lack of

[1] https://vitrivr.org/
[2] https://dbis.dmi.unibas.ch
[3] https://unity.com

accurate finger tracking, we have to tap on the keys with our controllers. If the keys are too close together, it might cause a problem in recognizing which key was pressed. Therefore, there needs to be either bigger keys or bigger spaces between two adjacent keys. This results in a bigger keyboard, which results in more needed movement with the arms. If we have to move our arms a lot to input some text, this can quickly become exhausting.

2.3 Word-Gesture Keyboard

A word-gesture keyboard may look pretty much the same as a conventional keyboard described in the last section, but works completely different. Independent of the details of the implementation, every word-gesture keyboard (also called slide-to-type keyboard) works with gestures. That means, instead of tapping on single keys, we have to draw one line or a shape on the keyboard. This will then be evaluated by an algorithm, that determines the closest word from a lexicon. Here closeness is determined by shape comparison between the user input and a word from the lexicon.

2.3.1 SHARK2

SHARK2 is a "large vocabulary shorthand writing system for pen-based computers" [1] invented by Shumin Zhai and Per-Ola Kristensson. It can compare the user inputted graph with a perfect graph of any word. A perfect graph is the graph, that is produced, if we start from the center of the word's first letter on the keyboard. Then we draw a straight line to the center of the next letter of the word and so on, until we reach the last letter.
The SHARK2 system needs a lexicon with words and all their perfect graphs stored. To get the most probable word from the lexicon the user intended to write, it compares the shapes of the graphs in the shape channel. For this, an amount of N sampling points has to be calculated for every graph. These N points need to be equidistant. Then they have to be normalized in scale and location. This means, that the graphs are all normalized by scaling the largest side of the graph's bounding box to a predetermined length L:

$$s = L/max(W, H) \tag{2.1}$$

W and H are the width and height of the graph's bounding box. All points' positions have to be divided by s to get the normalized points' positions. After that, the middle point of every graph have to be set to the point (0,0). Now the distance between the user inputted graph and every word's graph, that is in the lexicon, has to be calculated. To do so, we use the following formula:

$$x_s = \frac{1}{N} \sum_{i=1}^{N} \|u_i - t_i\|_2 \tag{2.2}$$

where u_i is the ith point of the user input graph and t_i the ith point of a word's graph. This is the so-called proportional shape mathcing distance [1]. The authors stated, that words can have a similar or even same shape as other words. They wrote, that for example on an ATOMIK layout with a lexicon of 20'000 words, there were 1117 pairs of words that have an identical graph.

To avoid these so-called confusion pairs, the authors were using a second channel for the location. TODO: LOCATION CHANNEL WITH OR WITHOUT FORMULAS?

With these two distances, the most probable word, the user intended to write, can be calculated. To do so, the authors used the following three formulas: TODO: INPUT 3 FORMULAS.

The final result c(w) for every word w that met the requirements tells the probability that w is the word, the user intended to write with his/her gesture.
The authors say, that the user can draw a graph either on visual guidance from the keyboard (looking for the next letter of a word on the keyboard) or recall from memory. A graph drawn by visual guidance results in a higher location distance score than a graph drawn from memory recall. If a user draws a graph by memory recall the location distance score will be poor and the focus lays more on the shape. Therefore, they suggest a dynamic weighting of the two channels, that is to adjust the weighting of the channels according to the time needed to draw the graph. In general, graphs drawn by memory recall are faster than visual guided ones. Hence, the gesture completion time should tell, how heavy the location channel should be weighted in the final selection. The time to complete a graph for a word obviously depends on its length and complexity. The authors then use Fitts' law (TODO: EITHER FORMULA OR REFERENCE TO IT) to calculate the normative writing time for a graph of a word. They use this result together with the actual graph production time to modify δ in (TODO: REFERENCE TO FORMULA THEY SUB DELTA).
The authors achieved quite satisfactory performance with the two channels, but there still might be conflicting words. To prevent these, the authors suggest to also use language information. For SHARK2 smoothed bigrams are used as the language model. It is then used to rearrange the N-best list of words received before.

3
Implementation

In this section, we will introduce how we implemented a word-gesture keyboard using Unity and a python script and how we used SHARK2 for our algorithm.

Word Graph Generator

As mentioned in the related-work SHARK2 part TODO: REFERENCE TO MENTIONED SECTION , to make SHARK2 work, the perfect graphs for all the words in a lexicon are needed. Therefore, we needed a script, that produces or overwrites a file for every available keyboard layout and writes the sampled points of every graph in it. Such a file contains per line a word, then a certain number of points from the word's perfect graph followed by the same points, but normalized (normalized as mentioned in the related-work SHARK2 section) TODO: REFERENCE TO MENTIONED SECTION.
To run the script, the user has to provide the name of the layout that he/she wants to create the perfect graphs for. Additionally, he/she has also to write the name of the text file containing all the words (lexicon). The script then either creates a new file named "sokgraph_*layout*.txt" or if already a file with this name exists, it deletes its content to write new in it. Then it fills the file line by line as mentioned above.
The file can only be executed for one layout a time. Hence, if there are more available layouts for our word-gesture keyboard, the user has to run the script several times.

3.1 Used Algorithm

3.2 Functions

"draw" words with gestures create own layout change size choose between best 5 words keyboard is movable chose between available layouts all the time possible add new words

4

Evaluation

This is the body of the thesis.

5 Conclusion

This is the body of the thesis.

5.1 Results Discussion

5.2 Future Work

Bibliography

[1] Per Ola Kristensson and Shumin Zhai. Shark2: a large vocabulary shorthand writing system for pen-based computers. In *UIST '04*, pages 43—52, 2004.

A

Appendix

University of Basel

Faculty of Science

Declaration on Scientific Integrity

(including a Declaration on Plagiarism and Fraud)
Translation from German original

Title of Thesis:

Name Assesor: _____

Name Student: _____

Matriculation No.: _____

With my signature I declare that this submission is my own work and that I have fully acknowledged the assistance received in completing this work and that it contains no material that has not been formally acknowledged. I have mentioned all source materials used and have cited these in accordance with recognised scientific rules.

Place, Date: _____ Student: _____

Will this work be published?

☐ No

☐ Yes. With my signature I confirm that I agree to a publication of the work (print/digital) in the library, on the research database of the University of Basel and/or on the document server of the department. Likewise, I agree to the bibliographic reference in the catalog SLSP (Swiss Library Service Platform). (cross out as applicable)

Publication as of: _____

Place, Date: _____ Student: _____

Place, Date: _____ Assessor: _____

Please enclose a completed and signed copy of this declaration in your Bachelor's or Master's thesis .