Eype - Using Eye-Traces for Eye-Typing

Sabrina Hoppe
DFKI GmbH
Saarbrücken, Germany
sabrina.hoppe@dfki.de

Florian Daiber
DFKI GmbH.
Saarbrücken, Germany
florian.daiber@dfki.de

Markus Löchtefeld
DFKI GmbH
Saarbrücken, Germany
markus.loechtefeld@dfki.de

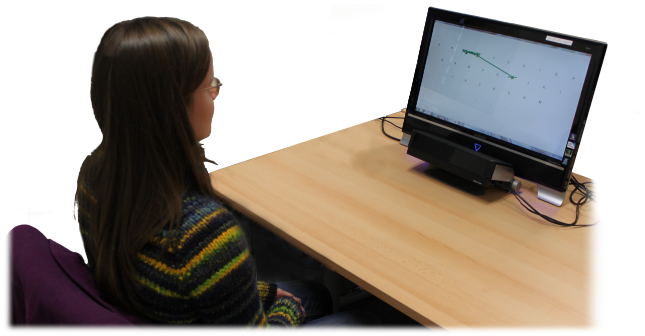

Figure 1: Eye-typing using eye-traces.

Abstract

Current eye-typing systems are suffering from the needed dwell timeout which limits the possible entry rate. In this position paper we discuss how the usage of eye-traces on on-screen keyboards could be used for almost dwell timeout free gaze based communication. This could significantly increase the entry rate of eye-typing systems.

Author Keywords

eye typing, eye traces, text entry, swype

ACM Classification Keywords

H.5.2 [Information interfaces and presentation]: User Interfaces. Input devices and strategies

Introduction and Motivation

Users with motor-disabilities such as the motor neuron disease (MND) or other motoric impairments that are not able to move any body extremity are often reliant on eye-tracking for communication with their environment. Especially text-to-speech engines controlled by the eyes are of increased importance for such users. When eye tracking is used for interaction the gaze position is most commonly used as a simple cursor, as done by Ware [7], who proposed dwell time based approaches as well as manual selection of attended targets. Unfortunately, dwell time based approaches are prone to the Midas Touch effect [1]. Besides other [6], eye-typing – gazing on letters of an on-screen keyboard – is the most wide spread approach for gaze-based communication. Unfortunately current eye-typing systems often suffer from the used dwell-timeouts where the user has to fixate the letter he wants to write for a certain time. This leads to entry rates of up to 20 wpm [4]. Even though approaches based on either a zoomable interface [6] or eye-gestures [8] have been explored they still reached on average a maximum of 26 wpm [6]. With a simulated dwell-free eye-typing system Kristensson and Vertanen achieved an entry rate of 46 wpm [2]. This indicates that dwell-time free systems will increase the performance drastically but would never be able to keep up with physical keyboards. To our knowledge no working implementation has yet been presented that would allow such dwell-free input. In this paper we describe our investigations towards an eye-typing technique called *Eype* that aims to minimize dwell-times. By applying the concept of [3, 9] that is also utilized for example in Swype [1] we use eye-traces for text entry. This allows minimizing the dwell-time and could possibly even pave the way for dwell free eye-typing.

Concept and Challenges

Although we apply concepts that are also used in touch systems (c.f. [3, 9]), eye-traces fundamentally differ from gestures on physical devices. The main differences and possible improvements are briefly discussed in the following.

Visual search
For physical systems, the eyes are more or less independent of the finger, i.e. the user can perform a visual search with his eyes, while the finger stays at a button and waits for the target identification. For Eype, this is not possible, because every eye movement the user does while searching will be used as an input trace. Though this problem will become less important for well-trained users, one could consider keyboard layouts based on digraph frequencies to reduce search time or highlight characters that are likely to follow the last input. A new keyboard layout might be refused by people who just temporarily want to use Eype, while it is an option for impaired people who do not use a qwerty keyboard in their every day life anyways.

Initial letter
A related problem concerns the matching of input traces to words from a dictionary. Swype can strongly rely on the initial letter, because the user performs a pointing task to indicate this letter. In Eype however, the first letter is also chosen via the eye tracker and is therefore not more reliable than any other letter in a word. This could be solved by either using a dwell time for the initial letter or adapting the matching algorithm.

Improvements on the matching from input to words
To cope with the uncertainty caused by the challenges above, other strategies to match input traces to words might be useful: for example, linguistic aspects like word frequencies could be considered.

In order to design a system that takes all these aspects into account, one could perform a user study. The influence of visual search by novices could be one objective as well as ways to detect search behavior, e.g. by analyzing gaze speed. Moreover, different keyboard layouts and the influence of a dwell time for the first letter could be explored.

[1]`http://www.swype.com`

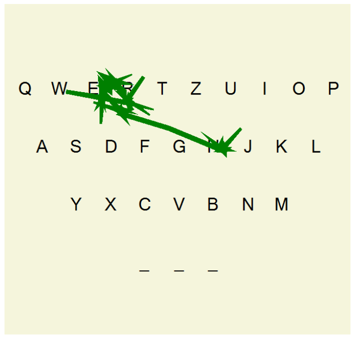

Figure 2: original input for 'here'

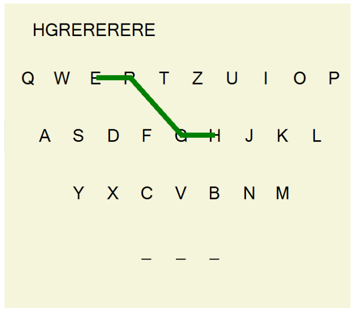

Figure 3: button representation

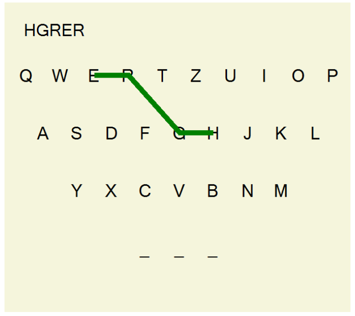

Figure 4: final input trace

Implementation

As hardware we used a Tobii X 60, a static binocular eye tracker that produces data with a rate of 60Hz, which was put in front of a 23.6" Display. The application was implemented in the .NET framework using C#. The configuration and calibration was done using the Tobii SDK 3.0 RC 13.
The process of finding the word the user wanted to type can be divided into 2 main steps: First, the input data is processed to a trace of characters. This trace is then compared to expected traces for a set of words from a dictionary.

From gaze data to eye-traces
As discussed above, eye-traces suffer from a certain uncertainty. Figure 2 illustrates what a typical input for the word 'here' looks like. While the user is still typing, we get the coordinates of his points of gaze (POG). These are matched to the buttons that are closest to the POG, so that a word is finally represented as a sequence of buttons. See Figure 3 for the button trace gained for the example input.

To avoid sequences with a high number of repetitions like 'xyxy', Eype eliminates this kind of traces if they are longer then 3 characters. This reduces the difference in length of an actual input and the optimal input, because most repetitions are caused by inaccuracy of the tracking system. However, this method could clearly be improved. For example, it could be checked, if there is a word in the dictionary that actually contains these repetitions. For most letter combinations like "qwqw" this is definitely not the case. See Figure 4 for the final input trace. Note, that this trace is what the program internally uses for processing, not what will be displayed as feedback for the user.

From character traces to a word
The system uses a word list saved in a data structure, such that we can map each character to the list of words starting with this character and at the same time, each word is linked to its optimal character trace: this is the trace that would have been the result, if a user had connected all the characters using the shortest way and there were no technical inaccuracies. To find the correct word, we thus need to compare the input to an appropriate set of words and evaluate their similarity.

Finding an appropriate set of words
A simple approach to find a set of similar words is to use all words starting with the same character. As discussed above, eye typing without dwell times cannot rely on the initial character. Therefore a neighborhood of initial letters could be used instead. However, this would dramatically increase search space. Moreover, words that have too long or too short optimal traces for an input are eliminated.

Similarity evaluation
Once we have found a subset of words of possible candidates we need to compare their optimal traces to the actual input. This is done for each possible word by the Needleman-Wunsch-algorithm [5]. Even though originally it was designed to find similarities between amino acid sequences it can easily applied to matching two character traces based on single comparisons between elements of the traces. The similarity of single characters is evaluated by a classification into three groups: equal, different and neighbors. A numerical value is assigned to every pair and finally, the best path is the one with the largest result from summing up all the values on it. To find the largest value, it is not necessary to compute all possible paths. Instead, dynamic programming is exploited to efficiently

derive the optimal solution.

Moreover, the sum of values also provides a comparable similarity measure that can be used to derive the closest word or a list of close words. This also raises the question, if it is feasible to offer the user several close words so that he can still correct the system in case his eye-trace was misinterpreted.

Conclusion

In this position paper we described our initial approach to overcome the need dwell time for eye-typing on on-screen keyboards. We propose to use the eye traces of users and compare their character traces against a dictionary using the Needleman-Wunsch-algorithm. Even though our approach still suffers from some not yet explored drawbacks such as the problem of defining the first letter, we think that it can be a valuable contribution to the workshop. Furthermore we hope that the workshop helps us defining further directions of this project.

Acknowledgments

This research project is partially supported by the Nuance Foundation.

References

[1] Jacob, R. J. K. What you look at is what you get: eye movement-based interaction techniques. In *Proceedings of the SIGCHI conference on Human factors in computing systems: Empowering people* (1990).

[2] Kristensson, P. O., and Vertanen, K. The potential of dwell-free eye-typing for fast assistive gaze communication. In *Proceedings of the Symposium on Eye Tracking Research and Applications*, ETRA '12, ACM (New York, NY, USA, 2012), 241–244.

[3] Kristensson, P.-O., and Zhai, S. Shark2: a large vocabulary shorthand writing system for pen-based computers. In *Proceedings of the 17th annual ACM symposium on User interface software and technology*, UIST '04, ACM (New York, NY, USA, 2004), 43–52.

[4] Majaranta, P., Ahola, U.-K., and Špakov, O. Fast gaze typing with an adjustable dwell time. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '09, ACM (New York, NY, USA, 2009), 357–360.

[5] Needleman, S. B., and Wunsch, C. D. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology 48*, 3 (1970), 443 – 453.

[6] Tuisku, Outi and Majaranta, Päivi and Isokoski, Poika and Räihä, Kari-Jouko. Now Dasher! Dash away!: longitudinal study of fast text entry by Eye Gaze. In *Proceedings of the 2008 symposium on Eye tracking research &#38; applications*, ETRA '08, ACM (New York, NY, USA, 2008), 19–26.

[7] Ware, C., and Mikaelian, H. H. An evaluation of an eye tracker as a device for computer input. *SIGCHI Bull. 17* (1986), 183–188.

[8] Wobbrock, J. O., Rubinstein, J., Sawyer, M. W., and Duchowski, A. T. Longitudinal evaluation of discrete consecutive gaze gestures for text entry. In *Proceedings of the 2008 symposium on Eye tracking research &#38; applications*, ETRA '08, ACM (New York, NY, USA, 2008), 11–18.

[9] Zhai, S., and Kristensson, P.-O. Shorthand writing on stylus keyboard. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '03, ACM (New York, NY, USA, 2003), 97–104.