COGAIN Symposium
Wuppertal, August 21st 2017.

Robust, real-time eye movement classification for gaze interaction using finite state machines

Antonio Diaz-Tula
University of São Paulo, Brazil

Carlos H. Morimoto
University of São Paulo, Brazil

Fixations and saccades are commonly used in gaze-based interfaces. State-of-the-art algorithms for eye movement segmentation work well for high speed and accurate eye trackers, which are still too expensive and bulky for most gaze interaction applications. For low-end eye trackers running at 30 to 60 Hz and with accuracy of about 1 degree, such algorithms do not perform as well. We propose a robust, real-time method to classify eye movement data into four categories: fixations, saccades, drifts, and none. The classifier is based on a finite-state machine (FSM) and is robust to missing data and blinks. The approach first filters raw gaze data to recover missing samples and smoothes the data. The current filtered sample is then classified by computing spatial dispersion and absolute eye velocity using a small number of recent gaze samples and the current state of the machine. Qualitative evaluation have shown evidence that FSM reduces latency after blinks, reduces the number of refocusing events and improves user experience during the interaction compared with a simple fixation detector based on a running average window. The source code is publicly available at `https://bitbucket.org/diaztula/gaze_movements_fsm/`.

Keywords: gaze interaction, fixations, saccades, finite state machine

Introduction

Dwell-time gaze based interfaces use fixations for pointing and selection ("clicking") of elements of the interface (Majaranta & Räihä, 2002). Though simple, this approach suffers from involuntary selections (known as the Midas touch problem (Jacob, 1990)), particularly when the dwell-time is very short. One alternative to avoid the Midas touch problem is the use of smooth pursuits (Esteves, Velloso, Bulling, & Gellersen, 2015) and gaze gestures (Wobbrock, Rubinstein, Sawyer, & Duchowski, 2008; Kurauchi, Feng, Joshi, Morimoto, & Betke, 2016). The classification of gaze gestures can be simplified by a robust method for the segmentation of fixations and saccades from the raw eye data stream (Salvucci & Goldberg, 2000; Belkacem, Shin, Kambara, Yoshimura, & Koike, 2015).

The accuracy of video-based eye trackers (that estimate point-of-gaze) is currently between 0.5-2 degrees of visual angle. Accuracy in gaze estimation is affected by several factors, such as variations in illumination conditions and by user's head movements. Besides intrinsic errors of the eye tracker, the segmentation of a fixation on an interface element (such as a key on a virtual keyboard) commonly requires smoothing of the raw eye data to remove high frequency components. Besides noise, gaze data is also corrupted by missing data e.g. during eye blinks. Dealing with blinks is an important issue for gaze interaction. For example, in a dwell-time based interface, a blink can interrupt the selection process, forcing the selection timer to start over again and slowing down the interaction.

There are freely available tools to detect fixations in raw gaze data, such as the ETU driver (Spăkov, 2017). Nonetheless, detecting fixations does not solve the problem of losing focus due to blinks or transient lost tracking of the eyes. In this paper we propose a rubost, real-time eye movement classification algorithm to improve the performance of gaze interfaces using a finite state machine (FSM).

The 2017 COGAIN Symposium: Communication by Gaze Interaction
Wuppertal, Germany. August 19th and 21st, 2017

Filtering and classification

A gaze data sample is basically composed of the coordinates of the estimated point-of-gaze and its time stamp. Some eye trackers might report also the pupil(s) diameter(s) and other features. Not all samples coming from a video-based eye tracker are valid samples. Invalid or undefined samples may be due to noise, lost tracking of the eyes, but most are reported during eye blinking.

We assume as input for our algorithm an eye data stream of samples that correspond to the nominal frequency of the eye tracker, where defined and undefined samples can be eas-

ily determined. The incoming samples are inserted in a queue Q of size N, where the element $Q[0]$ is the most recent, and $Q[N-1]$ is the oldest in the queue. The algorithm classifies the sample $Q[N/2]$. If this sample is undefined, its coordinates are estimated by averaging the coordinates of its left and right neighbors, provided that they are defined. If the current sample is defined, it is filtered using a Savitzky-Golay filter (Savitzky & Golay, 1964) and then classified as a fixation, saccade, or a drift, based on the spatial dispersion and absolute eye velocity computed from the samples in Q.

The parameters for detecting fixations are the maximum dispersion and minimum fixation duration. Dispersion is computed by averaging the variance of the x and y-coordinates. The maximum variance and minimum duration to classify the current sample as part of a fixation were empirically set to 2 degrees and 50 ms, respectively.

To decide if the current sample belongs to a saccade, we compute the absolute gaze velocity in degrees per milliseconds using a differential filter. If the absolute velocity is above a threshold then the current sample belongs to a saccade. This velocity threshold is computed dynamically from previous samples that belong to a fixation or a drift. Those samples are fitted to a normal distribution. After several tests, we defined the velocity threshold as the mean plus 1.7 times the standard deviation. A defined sample that belongs neither to a fixation nor to a saccade is classified as a drift.

Improving robustness using a finite state machine

To improve the robustness of the classification algorithm we defined the FSM shown in Figure 1. There are four main states: NONE, DRIFT, FIXATION, and SACCADE. Solid lines show transitions between these four main states. Different from existing approaches (e.g. (Salvucci & Goldberg, 2000)), our FSM includes additional (shadowed) states that model the behavior in the case of missing samples, blinks, and other artifacts. For example, if a undefined sample arrives during a fixation, then the state changes to NOISE_FIX. If shortly after entering NOISE_FIX a defined sample is received, then the state goes back to FIXATION. On the other hand, if undefined samples keep arriving for too long (more than *minBlink* milliseconds) then the state changes to BLINK_FIX, indicating that the user is likely to be blinking. A defined sample while in BLINK_FIX changes the state back to fixation. If the state remains in BLINK_FIX for longer that *maxBlink* milliseconds, then the state changes to NONE.

For the DRIFT state the behavior is similar to FIXATION. For the SACCADE state, the behavior is similar to FIXATION and DRIFT, except for the fact that after a blink during a saccade (BLINK_SACC) the state changes to DRIFT, since there are not enough defined samples to compute the absolute velocity reliably.

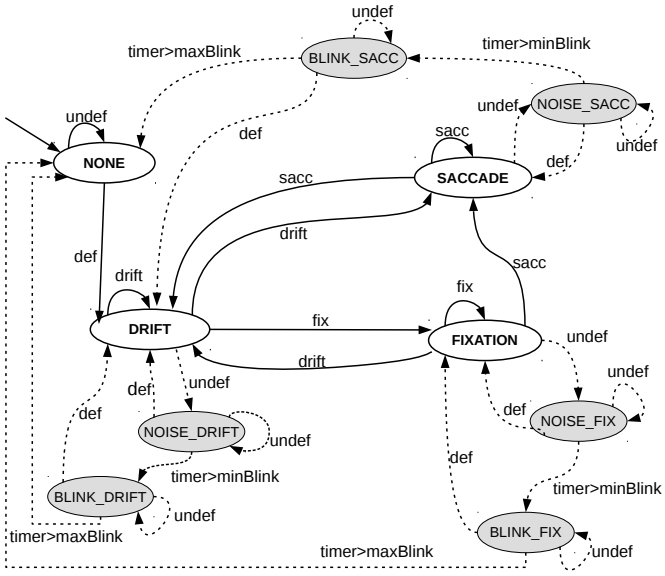

Figure 1. Finite state machine to classify gaze samples into fixation, saccade, drift, and none.

We have tested the proposed FSM in a dwell-time based virtual keyboard with frequencies of 500, 120, 60, and 30Hz. Even at 30Hz FSM has shown to be robust to blinks and small correctional saccades made while fixating a key. We have compared FSM with a simple fixation detector based on a running average window in the same front-end virtual keyboard. Though results are only qualitative, there is evidence that FSM reduces latency after blinks, reduces the number of re-focusing events and improves user experience during the interaction.

Conclusions

Our classification algorithm uses a finite state machine to classify the raw gaze data into 4 different states: fixation, saccade, drift, and none. We have tested the algorithm in a typing experiment using a dwell-time virtual keyboard (Diaz-Tula & Morimoto, 2016). Preliminary results show that the algorithm outperforms a simple running average window that just computes fixations, resulting in better performance and improved user experience. The algorithms are implemented in Python and run in real time. The source code is freely available for download at `https://bitbucket.org/diaztula/gaze_movements_fsm/`.

Acknowledgements

This work is supported by the São Paulo Research Foundation (FAPESP), grants 2016/10148-3 and 2012/04426-0.

References

Belkacem, A. N., Shin, D., Kambara, H., Yoshimura, N., & Koike, Y. (2015). Online classification algorithm for eye-movement-based communication systems using two temporal {EEG} sensors. *Biomedical Signal Processing and Control*, *16*, 40 - 47. Retrieved from http://www.sciencedirect.com/science/article/pii/S1746809414001554

Diaz-Tula, A., & Morimoto, C. H. (2016). Augkey: Increasing foveal throughput in eye typing with augmented keys. In *Proceedings of the 2016 chi conference on human factors in computing systems* (pp. 3533–3544). New York, NY, USA: ACM. Retrieved from http://doi.acm.org/10.1145/2858036.2858517

Esteves, A., Velloso, E., Bulling, A., & Gellersen, H. (2015). Orbits: Gaze interaction for smart watches using smooth pursuit eye movements. In *Proceedings of the 28th annual acm symposium on user interface software & technology* (pp. 457–466). New York, NY, USA: ACM. Retrieved from http://doi.acm.org/10.1145/2807442.2807499

Jacob, R. J. K. (1990). 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* (pp. 11–18). New York, NY, USA: ACM. Retrieved from http://doi.acm.org/10.1145/97243.97246

Kurauchi, A., Feng, W., Joshi, A., Morimoto, C., & Betke, M. (2016). Eyeswipe: Dwell-free text entry using gaze paths. In *Proceedings of the 2016 chi conference on human factors in computing systems* (pp. 1952–1956). New York, NY, USA: ACM. Retrieved from http://doi.acm.org/10.1145/2858036.2858335

Majaranta, P., & Räihä, K.-J. (2002). Twenty years of eye typing: systems and design issues. In *Proceedings of the 2002 symposium on eye tracking research & applications* (pp. 15–22). New York, NY, USA: ACM. Retrieved from http://doi.acm.org/10.1145/507072.507076

Salvucci, D. D., & Goldberg, J. H. (2000). Identifying fixations and saccades in eye-tracking protocols. In *Proceedings of the 2000 symposium on eye tracking research & applications* (pp. 71–78). New York, NY, USA: ACM. Retrieved from http://doi.acm.org/10.1145/355017.355028

Savitzky, A., & Golay, M. J. E. (1964). Smoothing and differentiation of data by simplified least squares procedures. *Analytical Chemistry*, *36*(8), 1627-1639.

Spăkov, O. (2017). *Etu driver.* http://www.sis.uta.fi/~csolsp/projects.php. (Accessed: 2017-05-20)

Wobbrock, J. O., Rubinstein, J., Sawyer, M. W., & Duchowski, A. T. (2008). Longitudinal evaluation of discrete consecutive gaze gestures for text entry. In *Proceedings of the 2008 symposium on eye tracking research & applications* (pp. 11–18). New York, NY, USA: ACM. Retrieved from http://doi.acm.org/10.1145/1344471.1344475