Poster: A Calibration-free Gaze-based Mobile Gesture Control System

Xipeng Ma, Chengkun Jiang, Yao Luo, Qilong Zhao, Meng Jin and Yuan He
School of Software and BNRist, Tsinghua University
maxp17@mails.tsinghua.edu.cn, jck15@mails.tsinghua.edu.cn,
luo-y17@mails.tsinghua.edu.cn, zhaoql17@mails.tsinghua.edu.cn,
mengj@mail.tsinghua.edu.cn, heyuan@tsinghua.edu.cn

Abstract

Gaze tracking is a promising human-computer interaction (HCI) technique that has multiple usage scenarios. However, calibration step is needed in most gaze-based interaction systems, which greatly affect user experience. In this proposal, we propose FreeGaze, a light-weight calibration-free gaze based gesture control system on commercial mobile devices. FreeGaze uses no delicated hardware, requires less computation overhead, does not rely on specific user behavior and most importantly, no calibration step required.

1 Introduction

Gaze tracking is a newly developed human-computer interaction technique that has various usage scenarios such as typing [11], device controlling [3] and identity authentication [4]. However, most gaze-based interaction systems requires calibration steps, which is time consuming.

In this proposal, we present FreeGaze, a light-weight calibration-free gaze based gesture control system that works on commercial mobile devices. FreeGaze has a two-layer structure: the gaze tracking layer and the gesture recognition layer. The gaze tracking layer derives the eye movement of the user by locating the corneal reflection, and the gesture recognition layer uses the eye movement to derive the input gestures.

2 Key Insight

Our idea comes from the corneal reflection-based gaze tracking methods [8]. As is shown in Figure 1, the corneal reflects a beam of infrared light, which is then captured by delicated camera. The gaze direction is then calculated using the optical path.

While infrared light source and delicated camera are unavailable in mobile devices, corneal reflection also works on visible light, e.g. the screen light of the device. Therefore, we can use the built-in front camera to capture users' facial image, locate the screen reflection, and then derive the eye movement.

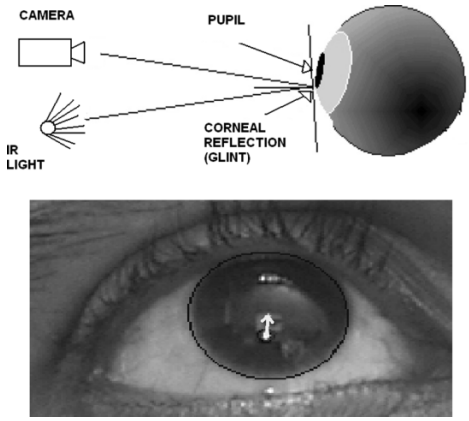

Figure 1. An pupil–corneal reflection gaze tracking technique [8]

3 System Architecture

Figure 2 shows the architecture of FreeGaze. The system awakes the front camera in certain schedule and capture the face image of the user. On recieving the data frames, the gaze tracking layer first extracts facial landmark data and thus locate the corneal reflection, then, the raw position data is pre-processed to reduce noise and then used for direction recognition. The direction data is then passed to the gesture recognition layer, which integrate the directions and then recovers the original input gestures.

3.1 Gaze Tracking Layer

To acuire gaze position data, we need to locate the reflection of screen's light in users' eyes from the input image data frames.

3.1.1 Face Detection and Landmark Alignment

For face detection, we use the built-in Haar cascade classifier [10] in openCV. Then, we use fast face alignment [2] to extract 68 landmark points on the user's face.

3.1.2 Face Alignment

FreeGaze takes the posture information of the first data frame as a template. On recieving each image data frame, FreeGaze extracts the face landmark of it, calculates the

International Conference on Embedded Wireless Systems and Networks (EWSN) 2019
25–27 February, Beijing, China

ISBN: 978-0-9949886-3-8

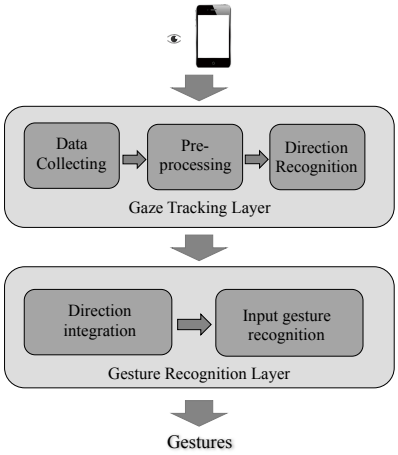

Figure 2. Architecture of the system

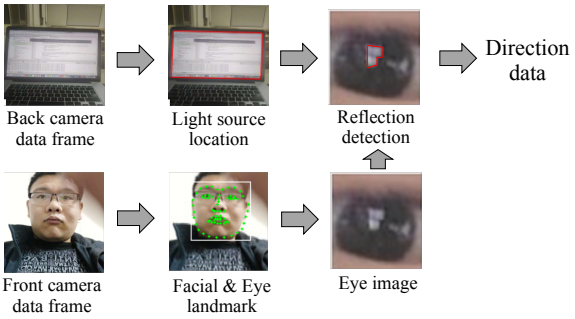

Figure 3. The process of direction recognition

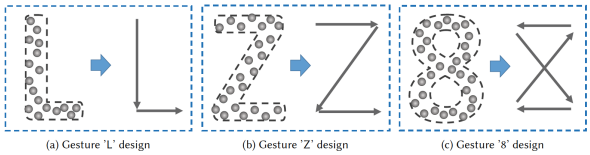

Figure 4. Gesture design examples [3]

transfer matrix M by matching landmark points, and then transforms the image to align the face position.

3.1.3 Reflection Detection

Based on the result of facial landmark detection, we cut out the images of two eyes from the font camera image data frame. To analyze the light condition of the environment, we leaverage the image from back camera and locate the light sources using OTSU [7]. Then, we match the shape of light sources with the eye images to locate the reflection. Finally, we merge the position changes in a sliding window to derive the moving directions. In SIFT algorithm [5], each feature vector of keypoint descriptors are aligned with 8 directions (”↑”, ”↓”, ”←”, ”→”, ”↖”, ”↗”, ”↙” and ”↘”,). In FreeGaze, we also use these eight directions to represent the relative position change. We use robust fitting [1] to calculate the overall direction change in each sliding window w_1. Figure 3 shows the process of direction recognition.

3.2 Gesture Recognition Layer

3.2.1 Gesture Design

Unlike traditional mobile gesture design methods that are based on absolute position [6, 9], the gesture design of FreeGaze is pretty like that of Gazeture [3], as is shown in Figure 4. The gestures are composed of the eight types of moving direction. Such design reduces user overhead of performing a gaze based gesture and can be easily extended.

3.2.2 Gesture Recognition

The first step is to merge the raw direction data of two eyes. For the data in each sliding window, if the direction data of the two eyes are of the same, the output merged data will remain the same; if there should be a contrast, the algorithm will choose the data that causes fewer direction change.

After merging the raw direction data, we use a second sliding window w_2 to scan and merge the directions into the final gesture data. The size of the sliding window is initially set to 1 and then adjusted to the length of the shortest consecutive same direction dynamically. Every consecutive same directions will be merged into one in the final output. Finally, we match the merged directions with the pre-defined gestures and find out the best fit, thus finish the gesture recognition.

4 References

[1] W. S. Cleveland. Robust locally weighted regression and smoothing scatterplots. *Journal of the American statistical association*, 74(368):829–836, 1979.

[2] V. Kazemi and J. Sullivan. One millisecond face alignment with an ensemble of regression trees. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1867–1874, 2014.

[3] Y. Li, Z. Cao, and J. Wang. Gazture: Design and implementation of a gaze based gesture control system on tablets. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 1(3):74, 2017.

[4] Z. Li, M. Li, P. Mohapatra, J. Han, and S. Chen. itype: Using eye gaze to enhance typing privacy. In *Proceedings of the IEEE International Conference on Computer Communications*, Atlanta, GA, USA, 2017.

[5] D. G. Lowe. Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, pages 1150–1157. Ieee, 1999.

[6] E. Miluzzo, T. Wang, and A. T. Campbell. Eyephone: activating mobile phones with your eyes. In *Proceedings of the second ACM SIGCOMM workshop on Networking, systems, and applications on mobile handhelds*, pages 15–20. ACM, 2010.

[7] N. Otsu. A threshold selection method from gray-level histograms. *IEEE transactions on systems, man, and cybernetics*, 9(1):62–66, 1979.

[8] J. Sigut and S.-A. Sidha. Iris center corneal reflection method for gaze tracking using visible light. *IEEE Transactions on Biomedical Engineering*, 58(2):411–419, 2011.

[9] V. Vaitukaitis and A. Bulling. Eye gesture recognition on portable devices. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, pages 711–714. ACM, 2012.

[10] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–I. IEEE, 2001.

[11] X. Zhang, H. Kulkarni, and M. R. Morris. Smartphone-based gaze gesture communication for people with motor disabilities. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, pages 2878–2889. ACM, 2017.