# Eye Gesture Recognition on Portable Devices

**Vytautas Vaitukaitis**
University of Cambridge
vytautas@vaitukaitis.lt

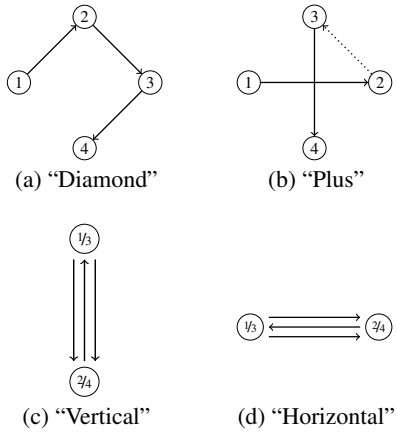
**Andreas Bulling**
University of Cambridge
Lancaster University
andreas.bulling@acm.org

## ABSTRACT
Hand-held portable devices have received only little attention as a platform in the eye tracking community so far. This is mainly due to their – until recently – limited sensing capabilities and processing power. In this work-in-progress paper we present the first prototype eye gesture recognition system for portable devices that does not require any additional equipment. The system combines techniques from image processing, computer vision and pattern recognition to detect eye gestures in the video recorded using the built-in front-facing camera. In a five-participant user study we show that our prototype can recognise four different continuous eye gestures in near real-time with an average accuracy of 60% on an Android-based smartphone (17.6% false positives) and 67.3% on a laptop (5.9% false positives). This initial result is promising and underlines the potential of eye tracking and eye-based interaction on portable devices.

### Author Keywords
Eye Gesture, Gaze Estimation, Mobile Phone, Laptop, Eye Tracking

### ACM Classification Keywords
C.3 Special-Purpose and Application-Based Systems: Real-time and embedded systems

## INTRODUCTION
Portable devices, such as mobile phones and tablets, have become an integral part of people's everyday life. Previous device generations were limited in their sensing capabilities and processing power. The advent of powerful smartphones equipped with high-resolution front-facing cameras points the way toward gaze-based interfaces that will become pervasively usable in everyday life [1]. Eye tracking on portable devices bears several challenges, in particular still inferior processing power compared to desktop computers and the lack of infrared illumination that is commonly used to increase tracking robustness to varying lighting conditions.
In this paper we present the first prototype implementation of an eye gesture recognition system that relies solely on hardware available on modern portable devices. Unlike previous approaches, our system does not require any external cameras or infrared illumination. Instead, it uses image processing and computer vision techniques to track the head and eyes in the video recorded from the front-facing camera.



(a) "Diamond" (b) "Plus" (c) "Vertical" (d) "Horizontal"

Figure 1: Eye gestures investigated in this work with circles indicating the discrete gaze directions used as gesture components. Numbers in the circles and lines between them illustrate the sequences of directions that form each gesture.

## RELATED WORK
Accurate eye tracking is challenging, particularly on the small screens of portable devices. Drewes *et al.* proposed eye gestures – sequences of several consecutive eye movements – as an alternative to pointing-based interaction. They demonstrated the feasibility of using eye gestures for interaction with a mobile phone but eye tracking and gesture recognition were still performed using a remote eye tracker and a high-performance desktop computer [3]. Nagamatsu *et al.* described a gaze-based interface for mobile devices that fused gaze with touch input but required two external video cameras with infrared diodes as well as a laptop computer for the computationally intensive video processing [6]. Miluzzo *et al.* are one of the few to investigate full on-device processing [5]. Their system did not distinguish between different gaze directions or recognise eye movement sequences but used the phone's built-in front-facing camera to detect which of the nine grid areas overlaying the camera image the user's eye was located in. The user could then perform a blink to trigger a command associated with each of these areas.


*UbiComp '12*, Sep 5-Sep 8, 2012, Pittsburgh, USA.
ACM 978-1-4503-1224-0/12/09.

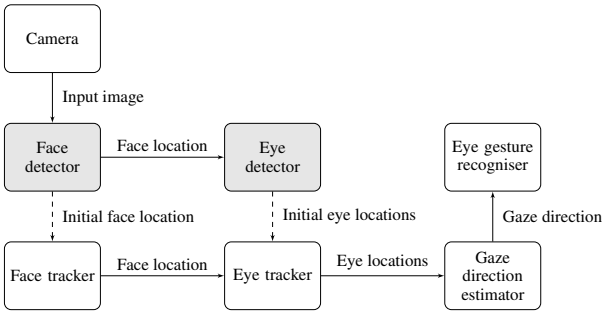



Figure 2: Architecture and information flow within the eye gesture recognition system. Grey rectangles indicate processing components that are only called during initialization.

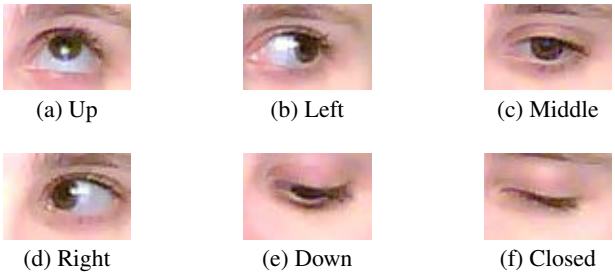



Figure 3: Image templates for different gaze directions taken during initialization. During operation, current eye image is matched against these templates to estimate gaze direction.

## EYE GESTURE RECOGNITION

Our eye gesture recognition system runs entirely on an Android-based mobile phone without the need for any additional equipment. The system was implemented using the OpenCV computer vision library. Figure 2 provides an overview of the system. Input to the system is a raw video stream from the device's front-facing camera. Because accurate point-of-gaze estimation is not required for gesture recognition, these images are scaled down to reduce processing time and thus increase system performance. The *face detector* first detects the user's face in each video frame using a Viola-Jones 20x20 frontal face detector [4]. The face location is then used by the *eye detector* to detect both eyes in the facial area using 18x12 eye detectors [2]. As soon as the face detector recognises a face, the *face tracker* is initialized. During initialization, a histogram of hue values is extracted in the face rectangle. In further frames, this histogram is back-projected onto the image to create a probability matrix values in which approximate probabilities of pixels belonging to the face. To track this distribution across the consecutive frames we use a mean shift algorithm. The *eye tracker* uses template matching with the output image of the eye detector as a template to track the eyes in consecutive frames. The matching is performed using a normalized correlation coefficient metric. Locations of both eyes are passed to the *gaze direction estimator* together with their matching scores. Gaze estimation also relies on template matching using a normalized correlation coefficient metric. In the initialization phase, the user is instructed to look at several different directions. Template images are taken for each of the left, right, up, down and middle (looking at the screen) directions as well as for "eyes closed" for both eyes (see Figure 3). During operation, all six templates are matched against the current image of the eye that had the higher matching score in the tracker. The template with the best match is returned as the estimated gaze direction. Finally, the *eye gesture recogniser* analyses the sequence of gaze directions to recognise a gesture. It first discards the "middle" direction and merges several occurrences of the same gaze direction. It then considers the last four filtered gaze directions together with their timestamps. If such four directions were detected within a window of 4 seconds, the sequence is compared to the predefined set of eye gestures. The gesture that matches the sequence exactly is returned as the system's output. Although the recogniser supports an arbitrary number of eye gestures, here, we only investigate four example gestures (see Figure 1).

## EVALUATION

We conducted a study to investigate the feasibility of running the gesture recognition system on two devices, namely an Android-based smartphone and a laptop. Five participants (four male and one female) aged between 21 and 25 years took part in the study. None of them had used eye-based interfaces before.

### Setup

The study was performed in a quiet, indoor laboratory setting with constant lighting conditions (see Figure 4). The following hardware was used for the study:

- laptop with a 2GHz dual-core CPU, 2GB RAM and an integrated camera capturing a 640x480 video at 25 frames per second (fps);
- Samsung Galaxy S smartphone running Android 2.3 with a 1GHz CPU, 512 MB RAM and an integrated front-facing camera capturing a 640x480 video at 30 fps.

### Procedure

Participants were seated about 70cm away from the first device facing its centre. While participants' movements were not constrained in front of the devices, we asked them to keep the distance to the device as stable as possible. Also, four key fobs were used to indicate the different gaze directions. The experimental procedure for each participant consisted of four parts: initial training, input of four predefined eye gestures on a laptop as well as a smartphone (see Figure 1), and interaction with a photo gallery application running on the phone using the same eye gestures. In the first three parts, we only recorded the videos from the built-in cameras of the devices. During the fourth task, the recognition was carried out in real time.

**Initial training.** Participants were asked to perform eight repetitions of each gesture in front of a laptop computer. The gestures were performed at fixed, 15-second intervals.

**Eye gesture input on a laptop.** Participants were asked to browse the Internet. At intervals uniformly distributed be-

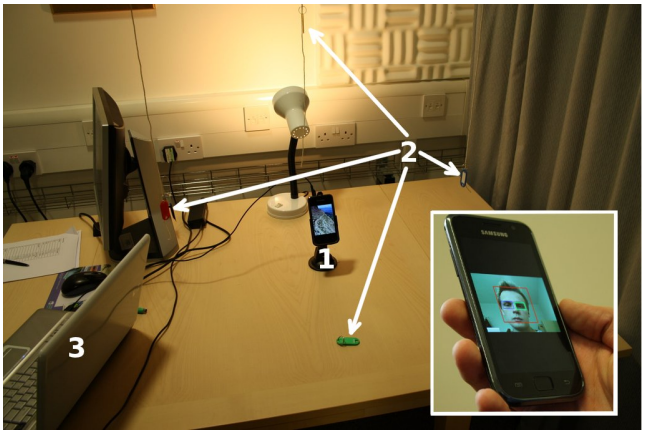



Figure 4: Experimental setup consisting of a smartphone placed in a dock (1), key fobs to mark the suggested gaze directions (2), and a laptop computer to monitor the course of the experiment (3). The picture in the lower right shows the gesture recognition system running on the phone.

tween 20 and 40 seconds participants were prompted to perform a specific eye gesture. This was repeated four times for each gesture.

**Eye gesture input on a mobile phone.** Participants were asked to interact with the phone's browser. At intervals uniformly distributed between 20 and 40 seconds participants were prompted to perform a specific eye gesture, which resulted in another 4 repetitions for each eye gesture.

**Interaction with the photo gallery.** Participants were asked to complete four different tasks using the photo gallery application (see Figure 1) for which we recorded the time and number of unsuccessful attempts until completion:

1. move to the next photo using the "Diamond" gesture;
2. return to the previous photo with the "Horizontal" gesture;
3. open the list of photo albums using the "Vertical" gesture;
4. close the photo gallery with the "Plus" gesture.

### Data analysis

We manually annotated the recorded videos to identify frames at which participants had finished a gesture. Then, an evaluation software was run to compute true positive (TP), true negative (TN), false positive (FP) and false negative (FN) counts as follows. A TP was counted if a correct gesture was detected within a tolerance period of 3 seconds before (corresponding to the typical duration of one gesture) and 1.5 seconds after each annotated frame. If none or an incorrect gesture was recognised during the period, it was counted as a single FN. If a gesture was recognised outside the tolerance period or an incorrect gesture was recognised within, the FP count was increased. Similarly, if the correct gesture was detected repeatedly during a single tolerance period the FP count was increased by the number of false detections. Finally, the TN count was calculated as the number of inter-gesture periods in the recorded video when the algorithm correctly did not recognise any gestures. For instance, if 16 gestures were performed in the recorded video and the algorithm recognised a gesture in two of the intervals between gestures, TN count would be 15. From these we computed sensitivity, false positive rate (FPR) and accuracy.

| | Laptop | Mobile phone |
|---|---|---|
| Tracking | 22.73 fps | 3.95 fps |
| | 174 ms | 574 ms |
| Recognition | 13.70 fps | 4.95 fps |
| | 188 ms | 319 ms |
| Baseline | 24.39 fps | 7.52 fps |
| | 108 ms | 298 ms |

Table 1: Mean frame rate (in fps) and maximum frame processing time (in ms) for tracking and recognition on the laptop and the mobile phone. The baseline corresponds to displaying the video on the screen without any processing.

## RESULTS AND DISCUSSION

### Real-time capabilities

We first analysed whether the system was able to process the video and recognise gestures in real-time. To this end, we measured the processing time for each frame and computed the mean frame rate.

As can be seen from Table 1, without any video processing, the mean frame rate on the laptop is very close to the maximum frame rate of the camera. The frame rate while tracking is also close to this limit. If the system performs face and eye detection the mean frame rate drops to 14 fps. However, the system spends only a small fraction of operating time in this state. By analysing the recorded videos we found that this proportion amounted to less than 0.03% of the experiment time. The maximum processing times are always below 200 ms. This implies that the system can process video at a rate greater than 5 fps even under high processing load.

On the mobile phone the frame rate without any processing is 7.5 fps. This is considerably lower than the maximum frame rate of the phone's front-facing camera. The most likely cause for this drop in performance is the video drawing on the screen that could not be switched off in the OpenCV port we used. As expected, additional processing further reduced this frame rate. Our current prototype implementation is able to detect the face and the eyes at about 5 fps and track both at around 4 fps.

### Recognition performance

A low frame rate may hamper eye gesture recognition if one or more gaze directions are missed and the system is therefore no longer able to recognise the corresponding eye gesture. We therefore analysed the system performance on both devices in more detail (chance level: 25%). Table 2 shows that, on the laptop, the recognition system achieved a mean accuracy for the initial training of 77.4% (sensitivity 61.4%,

| Experiment | Sens. [%] | FPR [%] | Acc. [%] |
|---|---|---|---|
| | *Laptop* | | |
| Initial training | 61.4 | 7.8 | 77.4 |
| Recognition | 33.8 | 5.9 | 67.3 |
| *Average* | 52.9 | 7.2 | 74.1 |
| | *Mobile phone* | | |
| Recognition | 28.3 | 17.6 | 60.0 |

Table 2: Sensitivity (Sens), false positive rate (FPR) and accuracy (Acc) of the eye gesture recognition system averaged over all five participants.

| | Sensitivity / FPR [%] | | |
|---|---|---|---|
| | **Initial training** | **Recognition on laptop** | **Recognition on smartphone** |
| P1 | 78.1 / 0.0 | 68.8 / 5.9 | 35.3 / 58.8 |
| P2 | 29.0 / 33.3 | 6.7 / 23.5 | 6.7 / 0.0 |
| P3 | 73.3 / 0.0 | 71.4 / 0.0 | 70.0 / 17.6 |
| P4 | 93.5 / 0.0 | 0.0 / 0.0 | 30.0 / 5.9 |
| P5 | 31.0 / 6.1 | 11.1 / 0.0 | 0.0 / 5.9 |

Table 3: Sensitivity and false positive rate (FPR) for each participant and all parts of the study.

FPR 7.8%), averaged over all participants. The performance in the recognition trials was slightly lower with a mean accuracy of 67.3% (sensitivity 33.8%) but at a lower FPR of 5.9%. As can also be seen from the table, the mean accuracy for gesture recognition on the smartphone was lower with 60.0% (sensitivity 28.3%, FPR 17.6%). This was expected given the lower processing frame rate on the smartphone.

Table 3 shows, however, that recognition performance varied considerably across participants. On the laptop, the system achieved a sensitivity of more than 65% for three of the five participants with performance reaching up to 78.1% for participant 1 for the initial training. The top performance was achieved by participant 4 with a sensitivity of 93.5% but with zero sensitivity in the recognition part. On the smartphone, the system achieved a sensitivity of up to 70% for participant 3, while for participant 2 the sensitivity was only 6.7%. After manual inspection of the recorded videos we found that the most likely reason for these differences were low-quality eye images and templates caused by head movements.

Three participants successfully completed all tasks with the gallery application; the other two were unable to complete all tasks and, after five unsuccessful input attempts, were asked to proceed to the next one. Overall, participants finished a task with a single attempt in 50% of the cases.

### Limitations and future work

The current work has several limitations. Our study only involved five participants and only considered a stationary indoor setting with controlled lighting conditions, fixed phone position and distance to the participant. In addition, the number of gestures as well as the interaction tasks were deliberately kept minimal. Furthermore, the prototype system does not run in true real-time (i.e. at least at 15 fps) and is sensitive to upper body and head movements.

We will address these limitations with a larger user study in a natural daily life setting using an improved second prototype of the recognition system. More specifically, we plan to investigate more efficient and potentially more robust techniques for eye tracking, such as gaze estimation using low-level image features and machine learning [8], as well as techniques for continuous head pose estimation to make the system more robust to head movements [7].

## CONCLUSION

In this work we presented the first eye gesture recognition system that runs entirely on a mobile phone and that, in contrast to earlier approaches, does not require any additional equipment. Our prototype combines several image processing, computer vision and pattern recognition techniques and achieves a near real-time gesture recognition accuracy of 60% on a smartphone and 67.3% on a laptop. These initial results not only demonstrate the feasibility but also underline the potential of eye gesture recognition and gaze-based interaction with portable devices.

## REFERENCES


1. A. Bulling and H. Gellersen. Toward Mobile Eye-Based Human-Computer Interaction. *IEEE Pervasive Computing*, 9(4):8–12, 2010.
2. M. Castrillón Santana, O. Déniz Suárez, M. Hernández Tejera, and C. Guerra Artal. Encara2: Real-time detection of multiple faces at different resolutions in video streams. *Journal of Visual Communication and Image Representation*, pages 130–140, 2007.
3. H. Drewes, A. De Luca, and A. Schmidt. Eye-gaze interaction for mobile phones. In *Proc. Mobility 2007*, pages 364–371. ACM Press, 2007.
4. R. Lienhart and J. Maydt. An Extended Set of Haar-like Features for Rapid Object Detection. In *Proc. ICIP 2002*, volume 1, pages 900–903, 2002.
5. E. Miluzzo, T. Wang, and A. T. Campbell. Eyephone: activating mobile phones with your eyes. In *Proc. MobiHeld 2010*, pages 15–20. ACM Press, 2010.
6. T. Nagamatsu, M. Yamamoto, and H. Sato. Mobigaze: development of a gaze interface for handheld mobile devices. In *Ext. Abstr. CHI 2010*, pages 3349–3354. ACM Press, 2010.
7. J. Tu, T. Huang, and H. Tao. Accurate head pose tracking in low resolution video. In *Proc. FGR 2006*, pages 573 –578, 2006.
8. Y. Zhang, A. Bulling, and H. Gellersen. Discrimination of gaze directions using low-level eye image features. In *Proc. PETMEI 2011*, pages 9–14. ACM Press, 2011.