# EyeTab: Model-based gaze estimation on unmodified tablet computers

Erroll Wood*
University of Cambridge
Cambridge, United Kingdom

Andreas Bulling†
Max Planck Institute for Informatics
Saarbrücken, Germany

## Abstract

Despite the widespread use of mobile phones and tablets, hand-held portable devices were only recently identified as a promising platform for gaze-aware applications. Estimating gaze on portable devices is challenging given the limited computational resources, the low quality of the integrated front-facing RGB camera, and the small screen to which gaze is mapped. In this paper we present *EyeTab*, a model-based approach for binocular gaze estimation that runs entirely on an unmodified tablet. EyeTab builds on set of established image processing and computer vision algorithms and adapts them for robust and near-realtime gaze estimation. A technical prototype evaluation with eight participants in a normal indoors office setting shows that EyeTab achieves an average gaze estimation accuracy of 6.88° of visual angle at 12 frames per second.

**Keywords:** gaze estimation, eye tracking, attentive user interfaces, gaze-based interfaces, portable devices

## 1 Introduction

Portable devices, such as mobile phones and tablets, have become daily life companions for billions of people. With fewer and fewer users actually making calls with their phones, gaze has become the second most important human sense at the phone/user interface, following touch. The way we visually interact with portable devices can reveal a lot about our visual and physical context [Bulling et al. 2011] but also the usability of the mobile user interface itself. Portable devices therefore have considerable potential as the first mainstream attentive user interface and promise a variety of gaze-aware applications, such as mobile user experience testing, enhancing the reading experience [Biedert et al. 2010], monitoring and understanding reading behaviour [Kunze et al. 2013], or simply selection of out-of-reach content on the screen.

Despite their potential, hand-held portable devices have so far still remained a niche platform for gaze-aware applications. This is mainly because estimating users' gaze on portable devices in mobile settings is significantly more challenging than "classical" eye tracking. Current eye trackers can rely on infrared illumination, high quality video cameras, substantial computational power, as well as a relatively stable position of the tracker relative to the users' eyes. In contrast, gaze estimation on portable devices has to deal with limited computational resources, low resolution images recorded using the integrated front-facing RGB camera and is further impacted by adverse lighting conditions, small screen sizes and UI elements to which gaze has to be mapped, short interaction spans [Oulasvirta et al. 2005], as well as head movements.

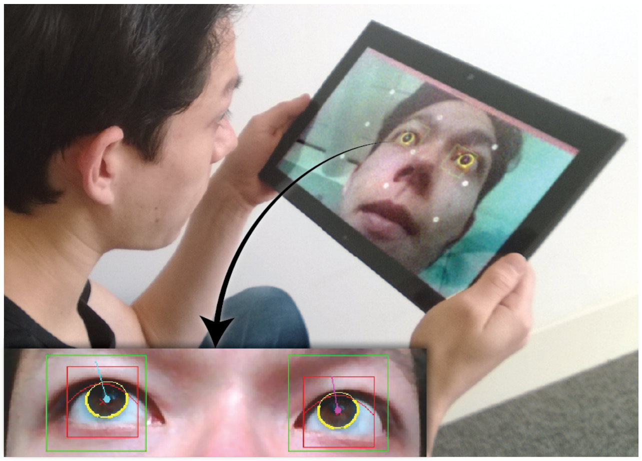


**Figure 1:** *EyeTab enables gaze-based interaction with unmodified tablet computers in near realtime using a model-based approach for binocular gaze estimation.*

In this work we present a model-based approach for binocular gaze estimation that runs entirely on an unmodified tablet. Our system, *EyeTab*, does not require any external cameras or infrared illumination but extends and adapts different image processing and computer vision techniques to track gaze using a portable device's front-facing camera (see Figure 1). In a technical prototype evaluation we show that the system can estimate gaze on a screen of 1920 × 1080 pixels with an accuracy of 6.88° of visual angle at 12 frames per second (fps). While this performance is still inferior to state-of-the-art remote and head-mounted eye trackers, we believe it represents a significant step towards the ultimate goal of developing a robust, fast and accurate gaze estimation system for portable devices. For the benefit of the research community, EyeTab's source code is available under an open source license [Wood 2013].

## 2 Related work

### 2.1 Gaze estimation from RGB images

Most commercially available eye trackers require specialised hardware and infrared light sources for gaze estimation. In contrast, appearance-based approaches work under visible light and use image processing and machine learning techniques to estimate gaze directly from video images. For example, Sesma et al. performed a study to evaluate the pupil centre eye corner (PC-EC) vector as an alternative to the well-established pupil corneal reflection method for low-cost gaze estimation [Sesma et al. 2012]. Their results show that PC-EC significantly reduced the accuracy of the gaze estimation under standard working conditions to $2 - 3°$ of visual angle. Valenti et al. describe a system for accurately tracking head-pose and eye positions using a low resolution webcam [Valenti et al. 2012]. The authors determine a point of gaze using a simple regression mapping of eye-features after an initial calibration. Our approach builds on the model-based "one-circle" algorithm presented in [Wang et al. 2003] which tracks gaze in RGB images using the

*e-mail: eww23@cam.ac.uk
†e-mail: andreas.bulling@acm.org

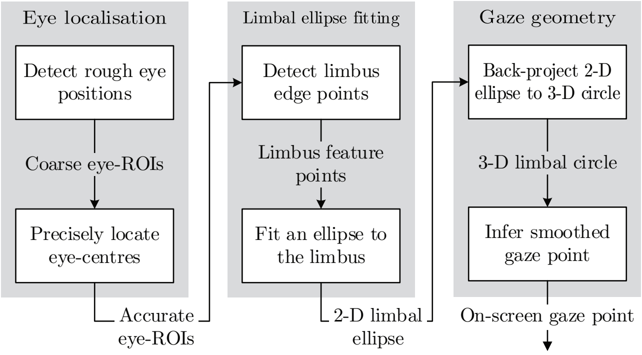

Eye localisation
Detect rough eye positions
Coarse eye-ROIs
Precisely locate eye-centres
Accurate eye-ROIs
Limbal ellipse fitting
Detect limbus edge points
Limbus feature points
Fit an ellipse to the limbus
2-D limbal ellipse
Gaze geometry
Back-project 2-D ellipse to 3-D circle
3-D limbal circle
Infer smoothed gaze point
On-screen gaze point

**Figure 2:** *Components of the EyeTab system: Eye localisation using cascade classifiers and a shape-based approach, limbus ellipse fitting to identify the 2D limbus positions, and point-of-gaze estimation using limbus back projection and gaze smoothing.*

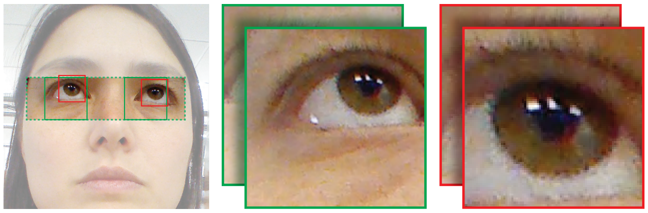
iris-contour (limbus). However, they used high quality DLSR cameras with zoom lenses, and only tested still images.

### 2.2 Gaze estimation on portable devices

Only a few previous works aim to enable gaze-based interaction with portable devices using full on-device processing. Common to all is the use of the front-facing camera integrated in most state-of-the-art smartphones, and estimation of gaze directly from recorded RGB video images. Miluzzo et al. present EyePhone, a system that allows users to interact with their phone using their eyes [Miluzzo et al. 2010]. While it doesn't require any external equipment, their system does not estimate gaze but rather uses the phone's front-facing camera to detect in which of nine grid areas overlaying the video image the user's eye is located. The user then performs a blink to trigger a command associated with each of these areas. Vaitukaitis and Bulling presented the first eye gesture recognition system that run entirely on a mobile phone [Vaitukaitis and Bulling 2012]. Their system combines established methods for face and eye detection with template matching on sequences of discrete gaze directions to detect a set of six different eye gestures. They reported an average gesture recognition accuracy of 60 % at about 4 fps. In a recent work, Holland et al. describe an eye tracking system for unmodified tablets [Holland et al. 2013]. Their system first detects the face, eyes and irises in the video image, and then uses an appearance-based approach in combination with a neural network to estimate on-screen gaze location. The authors reported an average spatial accuracy of about $4^\circ$ of visual angle but a temporal resolution of only 0.65 Hz. Kunze et al. introduced a reading application for smart phones and tablets [Kunze et al. 2013]. Their system uses the approximate head angle and eye positions detected in the video images to estimate if the user was looking at the screen.

## 3 The EyeTab system

Figure 2 provides an overview of EyeTab. We first extract rough eye positions from the image using cascade classifiers and locate eye-centres using a shape-based approach. We then detect possible limbus edge points within the resulting eye regions of interest (ROI) as maxima of the radial derivative, and determine the limbus' elliptical outline with robust model-fitting. The 2D ellipses are then back-projected to 3D to locate the limbus centres and the circle's normal vectors – the eyes' optical axes. Finally, we estimate a joint point-of-gaze (POG) by averaging and smoothing both eyes' POGs using the intersections between the 3D optical axes and the screen. In the following we describe each processing stage in more detail.



*Detected eye-pair* *Coarse eye-ROIs* *Refined eye-ROIs*

**Figure 3:** *Multi-stage precise eye localisation: from the rough position of an eye-pair to refined eye regions-of-interest.*

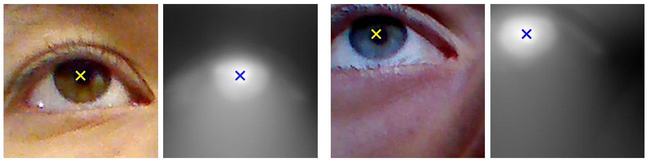


**Figure 4:** *Coarse eye ROIs and their estimated eye-centres.*

### 3.1 Eye localisation

EyeTab first determines the eyes' ROIs, reducing the search space for all subsequent processing stages, with a two-step approach (see Figure 3): rapidly approximating a rough ROI within the entire image and then refining ROI size and position to ensure they exhibit eye-features predictably. For rough ROI extraction, we use Haar-like feature based cascade classifiers to detect eye-pairs, and then segment two coarse ROIs from the eye-pair region. We opted to detect eye-pairs instead of faces because this approach is more robust in cases where part of the face may fall out of image bounds.

This approach is fast, reliable and user-independent but not always precise as coarse ROIs may be incorrectly centred and parts of the iris may lie outside ROI bounds. To precisely localise the eye centre we use a shape-based approach that exploits the predicted dark circular appearance of eye features [Timm and Barth 2011]. The centre of a circular pattern is defined as the point where the most image gradient vectors intersect. This is calculated for all possible points using an objective function to measure how well gradient vectors and eye-centre displacement vectors are aligned. As the eye-centre generally appears dark, we also weight each point by it's inverse intensity. As shown in Figure 4, the objective function can be accumulated in a centre-map, showing the eye-centre likelihood of each point. Once we know the precise positions of both a eyes in the image, we reposition the ROIs about them and resize the ROIs to a ratio of the screen-space interocular distance. This minimises them while still capturing important eye-features within their bounds.

### 3.2 Limbus ellipse fitting

Once we obtained accurate eye ROIs, we determine the limbus' elliptical outline. A popular iris boundary detector is Daugman's integro-differential operator which detects radial edges with an exhaustive search [Daugman 1993]. This approach is too computationally intensive for detecting ellipses in real-time. Instead, in EyeTab we rapidly identify a set of possible limbus edge points and robustly fit an ellipse model to them.

#### Detecting possible limbus points

We detect limbus boundary points as parts of radial edges by analysing the ROI's radial derivative. As shown in Figure 5, this is

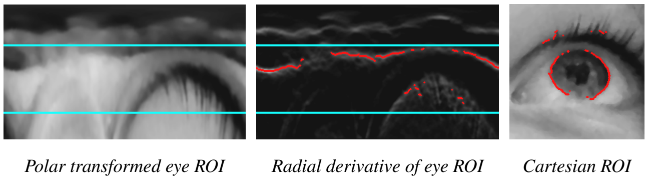

*Polar transformed eye ROI* *Radial derivative of eye ROI* *Cartesian ROI*

**Figure 5:** *Potential limbus points (red) within the search region (blue), in both polar and Cartesian domains.*

calculated using the vertical derivative of the ROI's polar transform. We take the maximum of each column of pixels in the radial derivative as a possible limbus point. We search between a minimum and maximum radius to avoid mis-classifying a pupil/iris boundary as a limbus point. The likelihood of an eyelid overlapping the iris is greatest at its top and bottom so we ignore these extreme angles.

Unlike Canny-based techniques [Wang et al. 2003], our approach does not use pre-defined thresholds. This makes it robust under a range of eye appearances, both across users and lighting conditions. It also supports out-of-focus images, as it does not require a strong edge. The surface of the cornea is reflective so nearby light sources can cause specular reflections that confound limbus detection. To suppress these, we threshold out the darkest half of the image to obtain a set of possible specularities. These appear in the thresholded image as small connected components which we then inpaint. As we only inpaint small regions this is not computationally expensive.

Unless eyes are extremely wide open, it is likely that the set of possible limbus points contains some that belong to eyelids or surrounding skin. Before fitting the ellipse, we remove erroneous points by approximating the upper eyelid with a parabola, and discarding exterior points. This parabola passes through three points: the iris/eyelid boundary above the eye-centre, and two eye corners. The iris/eyelid point is detected as a horizontal edge and we estimate eye-corners using a ratio of the on-screen interocular displacement.

### Robust ellipse fitting

The set of potential limbus points may contain outliers that adversely affect ellipses fit using a direct least-squares method. To handle these outliers we use a random sample consensus (RANSAC) method (see Figure 6) with modifications specifically designed for gaze tracking ellipse fitting [Świrski et al. 2012]. These include a modified support function which takes the image into account, preferring ellipse models with geometric gradients that match strong image gradients.

## 3.3 Gaze geometry

When a 3D circle is projected onto an image plane, it appears as an ellipse. We reverse this process to locate the limbus centre in 3D space, and the 3D circle's normal vector – the optical axis. We use pinhole camera projective equations to determine the distance between the camera and the limbus. This relies on pre-calculated camera internal parameters and prior physiological knowledge of the limbus' size – we assume its radius to be roughly constant at 6mm. We then map the 2D ellipse's centre into its 3D position using knowledge of its distance and camera focal lengths.

We determine the optical axis by finding the 3D rotation which maps a unit normal vector facing the screen into the 3D limbus' position. While a closed form linear algebra solution exists [Wu et al. 2004], we employ a trigonometric approximation for simplicity. First we rotate the normal vector about the $y$-axis to account for ellipse eccentricity, and then about the $z$-axis for ellipse orientation.

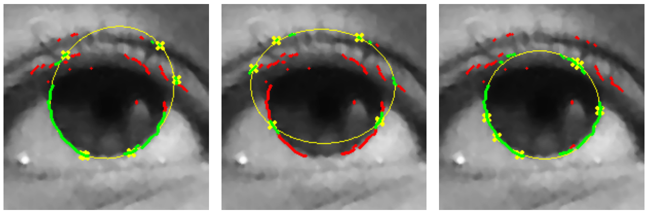


**Figure 6:** *After three RANSAC iterations we choose the best model (the rightmost). Yellow crosses: initial sample; green dots: inliers.*

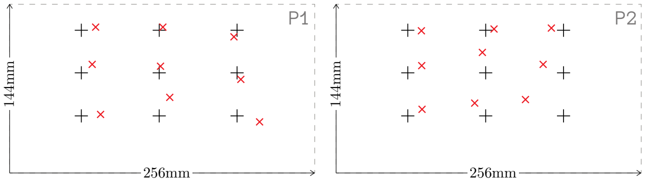


**Figure 7:** *POG (black) and average estimated POG (red) for participants P1 and P2. The on-screen grid position is shown.*

We then correct for the displacement of the 3D limbus centre by rotating the normal by the 3D limbus' $x$ and $y$ angular offset.

After calculating each eye's 3D position and orientation, we estimate its POG as the intersection between that eye's optical axis and the 3D screen. We assume the screen lies in-plane with the camera origin, and obtain a joint POG by averaging both eyes' POGs. The raw output of gaze tracking systems typically contains noise, so as a final step we smooth the joint POG using a triangle kernel over the previous five gaze points [Špakov 2012], and discard any extreme outliers beyond screen boundaries.

# 4 Technical evaluation

We evaluated EyeTab to investigate key performance parameters, in particular gaze estimation accuracy and processing speed. To this end we deployed EyeTab on an 11-inch (1920 × 1080 pixels) commodity tablet computer with a quad-core 2 GHz processor and 8 GB of RAM running Windows 8. We recruited 8 participants (1 female) aged between 20 to 27. We asked each participant to look at nine on-screen locations that were laid out in a 3 × 3 grid pattern. At each location we recorded smoothed POG data over 30 video frames. Gaze estimation accuracy was measured as the root-mean-square error (RMSE) between all true and estimated gaze points. Accuracy was measured across the full pattern, and for the top-half of the screen only (six points). Because we approximate the direction of gaze with the optical axis, no calibration procedure was performed. Participants held the device themselves in a normal indoors office environment. The tablet was held in reverse-landscape orientation, with the front-facing camera at the bottom. Though participants were allowed free head movement, they were instructed to hold the device at about 20 cm away from their eyes.

### Results

The system processed camera frames of resolution 1280×720 pixels at 12 fps and the average user distance was 21.6 cm. RMSE was 25.8 ± 6.79 mm (6.88 ± 1.80°) for the full pattern, and 19.1 ± 6.21 mm (5.08 ± 1.65°) for the top-half. Table 1 show RMSE across participants, and Figure 7 visualises estimated POGs against the true POG grid for P1 and P2. The system failed to track gaze (RMSE $> 20^\circ$) for P4 and P6, and failed to detect eyes for P8.

**Table 1:** *Root-mean-square error of gaze estimation accuracy in mm and degrees of visual angle for the full and top-half pattern. Results not given for failure cases* `P4`*,* `P6`*, and* `P8`*.*

| | *full pattern* | | *top-half pattern* | |
|---|---|---|---|---|
| **Participant** | **mm** | **degree** | **mm** | **degree** |
| `P1` | 15.7 | 3.83 | 10.9 | 2.68 |
| `P2` | 20.5 | 5.95 | 15.9 | 4.65 |
| `P3` | 32.3 | 7.52 | 27.1 | 6.32 |
| `P5` | 27.2 | 9.73 | 15.7 | 5.67 |
| `P7` | 33.3 | 7.74 | 25.5 | 5.94 |

## 5 Discussion

While EyeTab's accuracy was slightly lower than that reported in the most related previous work [Holland et al. 2013], in contrast to that work, EyeTab provides gaze vectors at 12 fps calculated with a full model-based approach that can handle head movement more robustly [Hansen and Ji 2010]. Post-hoc profiling revealed that precise eye-centre localisation was the bottleneck, accounting for more than 50 % of the processing time – this could be alleviated with a parallelized implementation.

The performance we achieved allows for basic interactions, e.g. dismissing "toast" notifications. The range of results (see Table 1) suggests that performance can likely be improved with more sophisticated algorithms. We believe an accuracy around 3° should be possible and would allow for UI-element interactions. The difference between visual and optical axes currently prevents EyeTab from achieving an accuracy comparable to fully calibrated commercial eye trackers, so in future work we aim to perform per-user calibration to estimate this offset and improve accuracy. The system's performance depends on the spatial resolution of eye-images – more limbus edge pixels would more accurate ellipse fitting. Though 20 cm may seem too close, different devices with higher resolution cameras or better lenses would allow accurate ellipse fitting with the device held further away.

For `P8` coarse eye-pair detection failed completely, and for some others it depended on how they held the tablet. Haar-like cascade classifiers are generally reliable, but not rotationally invariant. We found robustness could be improved by detecting eye-pairs at multiple angles; and in the rare situations where eye-pairs are not detected under ideal head-pose, we found backing-off to a single-eye detector helpled. Though coarse eye-localisation succeeded for `P4`, eye-centre localisation failed systematically. This is because certain lighting conditions cause extremely dark shadows around deep inset eyes, covering the limbus and confounding eye-centre localisation. We found that this can be avoided by combining the gradients method with the more illumination-invariant circular pattern isocentres technique [Valenti et al. 2012].

Across all participants, a constant source of error was outliers in the set of potential limbus points. When eyelid localisation fails, a large number of incorrect feature points are passed to RANSAC resulting in a poorly fit ellipse. This was especially problematic for `P6`. This could be alleviated with a better eyelid localisation technique, such as a parabolic Hough transform or active contours at higher computational cost. This issue is aggravated when the user looks downwards, as top and bottom eyelid/iris occlusion becomes severe, so the number of true limbus points found is low. As no edge points can be found for the top and bottom of the limbus, the ellipse's height is inaccurate. This explains the drop in accuracy towards the bottom of the pattern. This limitation could be avoided by designing gaze-interaction techniques around it, e.g. only using gaze-based interfaces in the screen's top-half.

## 6 Conclusion

We presented EyeTab, the first model-based binocular gaze estimation system for unmodified tablet computers. A technical evaluation showed that our prototype implementation estimates gaze with an accuracy of about 7° at 12 frames per second. While gaze estimation on portable devices still faces a number of significant challenges, these results are promising and open up new avenues for research on mobile gaze-based and attentive user interfaces.

## References

Biedert, R., Buscher, G., Schwarz, S., Hees, J., and Dengel, A. 2010. Text 2.0. In *Proc. CHI*, 4003–4008.

Bulling, A., Roggen, D., and Tröster, G. 2011. What's in the eyes for context-awareness? *IEEE Pervasive Computing*.

Daugman, J. G. 1993. High confidence visual recognition of persons by a test of statistical independence. *IEEE TPAMI*.

Hansen, D. W., and Ji, Q. 2010. In the eye of the beholder: A survey of models for eyes and gaze. *IEEE TPAMI*.

Holland, C., Garza, A., Kurtova, E., Cruz, J., and Komogortsev, O. 2013. Usability evaluation of eye tracking on an unmodified common tablet. In *Proc. CHI*, 295–300.

Kunze, K., Ishimaru, S., Utsumi, Y., and Kise, K. 2013. My reading life: towards utilizing eyetracking on unmodified tablets and phones. In *Proc. UbiComp*, 283–286.

Miluzzo, E., Wang, T., and Campbell, A. T. 2010. Eyephone: activating mobile phones with your eyes. In *Proc. MobiHeld*, 15–20.

Oulasvirta, A., Tamminen, S., Roto, V., and Kuorelahti, J. 2005. Interaction in 4-second bursts: the fragmented nature of attentional resources in mobile hci. In *Proc. CHI*.

Sesma, L., Villanueva, A., and Cabeza, R. 2012. Evaluation of pupil center-eye corner vector for gaze estimation using a web cam. In *Proc. ETRA*, 217–220.

Świrski, L., Bulling, A., and Dodgson, N. 2012. Robust, real-time pupil tracking in highly off-axis images. In *Proc. ETRA*, 173–176.

Timm, F., and Barth, E. 2011. Accurate eye centre localisation by means of gradients. In *Proc. VISAPP*, 125–130.

Vaitukaitis, V., and Bulling, A. 2012. Eye gesture recognition on portable devices. In *Proc. PETMEI*, 711–714.

Valenti, R., Sebe, N., and Gevers, T. 2012. Combining head pose and eye location information for gaze estimation. *IEEE Transactions on Image Processing 21*, 2, 802–815.

Špakov, O. 2012. Comparison of eye movement filters used in hci. In *Proc. ETRA*, 281–284.

Wang, J.-G., Sung, E., and Venkateswarlu, R. 2003. Eye gaze estimation from a single image of one eye. In *Proc. ICCV*.

Wood, E., 2013. EyeTab source. `http://www.cl.cam.ac.uk/research/rainbow/projects/eyetab/`.

Wu, H., Chen, Q., and Wada, T. 2004. Conic-based algorithm for visual line estimation from one image. In *Proc. AFGR*.