


Facilitating Object Detection and Recognition through Eye Gaze

Conference Paper**Author(s):**

Báce, Mihai; Schlattner, Philippe; [Becker, Vincent](#) ; Sörös, Gábor

Publication date:

2017-09-04

Permanent link:

<https://doi.org/10.3929/ethz-b-000221545>

Rights / license:

[In Copyright - Non-Commercial Use Permitted](#)



Conference Paper

Facilitating Object Detection and Recognition through Eye Gaze

Author(s):

Bâce, Mihai; Schlattner, Philippe; Becker, Vincent; Sörös, Gábor

Publication Date:

2017-09-04

Rights / License:

[In Copyright - Non-Commercial Use Permitted](#) →

This page was generated automatically upon download from the [ETH Zurich Research Collection](#). For more information please consult the [Terms of use](#).

Facilitating Object Detection and Recognition through Eye Gaze

Mihai Bâce, Philippe Schlattner, Vincent Becker, Gábor Sörös

Department of Computer Science

ETH Zurich, Switzerland

{mbace | pschlatt | vbecker | soeroesg}@inf.ethz.ch

ABSTRACT

When compared to image recognition, object detection is a much more challenging task because it requires the accurate real-time localization of an object in the target image. In interaction scenarios, this pipeline can be simplified by incorporating the users' point of regard. Wearable eye trackers can estimate the gaze direction, but lack own processing capabilities. We enable mobile gaze-aware applications by developing an open-source platform which supports mobile eye tracking based on the Pupil headset and a smartphone running Android OS. Through our platform, we offer researchers and developers a rapid prototyping environment for gaze-enabled applications. We describe the concept, our current progress, and research implications.

ACM Classification Keywords

H.5.2. Information Interfaces and Presentation (e.g. HCI): User Interfaces

Author Keywords

Eye Tracking; Eye Gaze; Wearable Computing; HCI.

INTRODUCTION

We live in a world surrounded by many complex computer systems. Having so many different devices around us implies that before the interaction step one has to identify the object of interest. With camera technology becoming inexpensive and widely available, there is a significant body of research in the areas of object detection and object recognition. Object detection has the additional complexity of having to accurately locate the object of interest in a particular image. Moreover, given an increasing number of smart objects, the user has to select the desired object in case of ambiguities. Often, users direct their visual attention towards the object of interest and this can be captured by the eye gaze. Hence, eye tracking has the potential to simplify and facilitate the object detection and recognition pipeline by incorporating the users' point of interest.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

MobileHCI '17 Workshops, September 04–09, 2017, Vienna, Austria

© 2017 Copyright held by the owner/author(s).

Recent technological advancements have made wearable eye trackers (e.g., Pupil [6]) more affordable and widely accessible. One limitation common to many such platforms is that they do not have any own computing capabilities. They need to be connected to a powerful computer for data processing. Recently, there have been attempts to increase the portability of the Pupil platform. A companion Android application¹ (no iOS support yet) allows recording eye tracking sessions and processing the data offline or streaming video data over the network. While this may be sufficient for statistical analysis of the data, this mode of operation does not support online interactive gaze-aware systems.

To overcome this issue, we are developing an open-source platform to enable real-time wearable eye tracking. Users can connect the Pupil eye tracker to their own Android device and rapidly develop their own prototypes. All the routines required for pervasive eye tracking like pupil detection and calibration are running locally on their own mobile device.

RELATED WORK

Traditionally, visual recognition tasks have been mostly based on SIFT [7] or HOG [3]. More recently, methods that required some feature engineering have been outperformed by methods based on convolutional neural networks (CNNs). R-CNN [5] or Faster R-CNN [9] are examples of architectures that can achieve detection rates of around 70-80%.

To further improve object detection, interactive systems can incorporate the gaze direction as an indication of the user's point of regard. Eye gaze is a well-established area in HCI [2]. The gaze direction has been used to build systems which assist users with typing text, playing games, or controlling user interfaces. A more comprehensive list of applications enabled by eye tracking is presented by Duchowski [4].

In our work, we want to facilitate the object recognition pipeline by reducing the region of interest in an image around the user's point of gaze.

PLATFORM OVERVIEW

The mobile eye tracking platform (Figure 1A) consists of two components: The Pupil wearable eye-tracker [6] and a standard off-the-shelf smartphone with Android OS (an LG Nexus 5X in our case). The eye tracker is connected to the smartphone via a USB-C interface. The Pupil eye tracker is

¹<https://play.google.com/store/apps/details?id=com.pupillabs.pupilmobile>

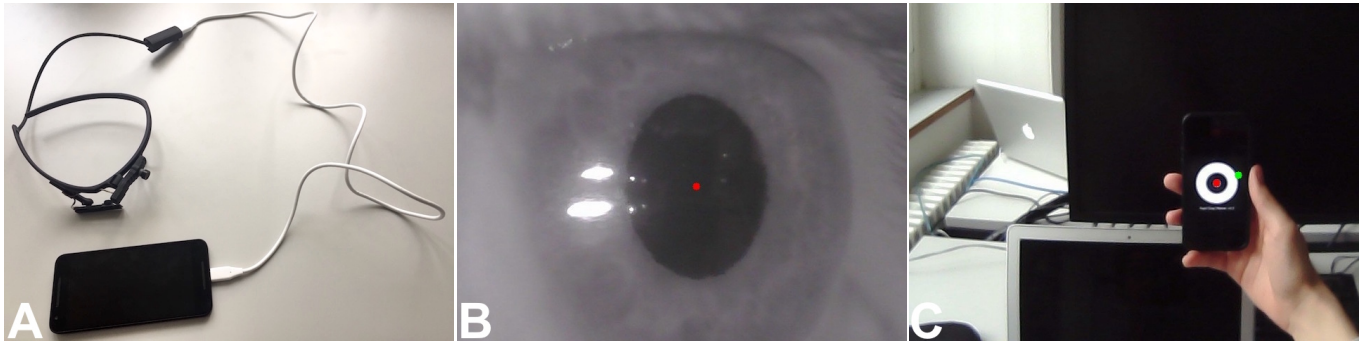


Figure 1. A) The mobile eye tracking platform: A Pupil eye tracker connected to an LG Nexus 5X. The platform allows users to move freely in the environment. B) Dark pupil detection on the image captured by the infrared spectrum camera. The red dot represents the center of the pupil. C) Calibration result: The red dot represents the ground truth detected marker and the green dot represents the predicted gaze location.

equipped with two USB cameras compliant with the UVC standard: A scene camera and an infrared eye camera for dark pupil detection. Besides the hardware, the platform provides developers the necessary software to connect the two devices and the routines for pupil detection and calibration with respect to the scene camera. The Android device is able to capture images from the Pupil's two cameras using the UVCCamera library².

Pupil detection

The user's pupil is detected with an algorithm provided in the open-source Pupil repository³. A Canny edge detector finds contours in the image. The edges which are not part of the pupil area (dark area) are filtered out. Candidate pupil ellipses, found by fitting ellipses to the contours, are ranked according to their confidence value. A confidence value for a fit represents the ratio between the ellipse fit and the length of the supporting edge and ellipse circumference. The top ranking fit is considered valid if its confidence value is above a predefined threshold.

Calibration marker detection

A calibration marker (Figure 1C) helps collect ground truth data for the calibration procedure. The marker is displayed on the user's device and represents a set of three concentric circles. To detect this marker, we use OpenCV to find three nested ellipses. The center of the inner-most ellipse is taken as the marker center.

Calibration

The calibration procedure aims to find a mapping between a point in the eye space to a point in the scene space. A point in the eye space represents the center of the pupil (Figure 1B - marked in red), while a point in the scene space represents the center of the calibration marker displayed on the device's screen (Figure 1C - marked also in red). To calibrate the eye tracker, users have to manually collect multiple such calibration points. After multiple pairs of points have been collected, the gaze mapping function has to be estimated. Our implementation is based on the Levenberg-Marquardt least-squares minimization and curve fitting and integrates the lmfit library⁴

²<https://github.com/saki4510t/UVCCamera>

³<https://github.com/pupil-labs/pupil>

⁴<http://apps.jcns.fz-juelich.de/doku/sc/lmfit>

for C++. After the calibration procedure, the mapping function can be used to predict the point of gaze in the scene image (Figure 1C - marked in green) given the center of the pupil in the eye image (Figure 1B).

CHALLENGES AND RESEARCH IMPLICATIONS

One challenge when building gaze-aware systems is the calibration procedure. This is a critical component of the system because it influences the reliability of the eye tracker. Most eye trackers offer an on-screen calibration method where users have to fixate at a sequence of markers. This method is not appropriate for our platform because the display of the smartphone is not large enough to cover a large range of visual angles.

In our current experiments, the calibration marker has been displayed on the smartphone's screen. Users have to fixate on this target and then click a button to collect a sample calibration point. In practice, we have noticed that a minimum of six points spread over the target space are required to achieve good results (e.g., Figure 1C - the green dot represents the predicted gaze point). According to Pfeuffer et al. [8], users find this task difficult and tedious.

Pursuit calibration [8] is a possible solution for the mobile eye tracking platform. However, the method relies on displaying a moving target on a large display where the speed and trajectory can be easily controlled. Our challenge is to find a similar approach where the moving target is the device itself.

CONCLUSION

We presented our initial efforts in building an open-source mobile platform for pervasive eye tracking and gaze-aware applications. By incorporating the user's point of regard, we simplify the object recognition pipeline. Our platform already has the necessary components which are required for any eye tracking system. In the future, we will develop a more user-friendly and reliable calibration procedure. Our vision is to provide developers and researchers an open platform which makes interactive gaze-enabled applications (e.g., ubiGaze [1]) easier to prototype and test.

REFERENCES

1. Mihai Băce, Teemu Leppänen, David Gil de Gomez, and Argenis Ramirez Gomez. ubiGaze: Ubiquitous Augmented Reality Messaging Using Gaze Gestures. In *Proc. SIGGRAPH Asia'16 MGIA*. 11:1–11:5.
2. Andreas Bulling and Hans Gellersen. Toward Mobile Eye-Based Human-Computer Interaction. In *IEEE Pervasive Computing '10*. 8–12.
3. Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Proc. CVPR'05*. 886–893 vol. 1.
4. Andrew T. Duchowski. A breadth-first survey of eye-tracking applications. In *Behavior Research Methods, Instruments, & Computers*. 455–470.
5. Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. In *Proc. CVPR'14*. 580–587.
6. Moritz Kassner, William Patera, and Andreas Bulling. Pupil: An Open Source Platform for Pervasive Eye Tracking and Mobile Gaze-based Interaction. In *Proc. UbiComp'14 Adjunct*. 1151–1160.
7. David G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. In *Int. Journal of Computer Vision '04*. 91–110.
8. Ken Pfeuffer, Melodie Vidal, Jayson Turner, Andreas Bulling, and Hans Gellersen. Pursuit Calibration: Making Gaze Calibration Less Tedious and More Flexible. In *Proc. UIST'13*. 261–270.
9. Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards Real-time Object Detection with Region Proposal Networks. In *Proc. NIPS'15*. 91–99.