PuReST: Robust Pupil Tracking for Real-Time Pervasive Eye Tracking

Thiago Santini
University of Tübingen
Tübingen, Germany
thiago.santini@uni-tuebingen.de

Wolfgang Fuhl
University of Tübingen
Tübingen, Germany
wolfgang.fuhl@uni-tuebingen.de

Enkelejda Kasneci
University of Tübingen
Tübingen, Germany
enkelejda.kasneci@uni-tuebingen.de

ABSTRACT

Pervasive eye-tracking applications such as gaze-based human computer interaction and advanced driver assistance require real-time, accurate, and robust pupil detection. However, automated pupil detection has proved to be an intricate task in real-world scenarios due to a large mixture of challenges – for instance, quickly changing illumination and occlusions. In this work, we introduce the *Pupil Reconstructor with Subsequent Tracking* (*PuReST*), a novel method for fast and robust pupil tracking. The proposed method was evaluated on over 266,000 realistic and challenging images acquired with three distinct head-mounted eye tracking devices, increasing pupil detection rate by 5.44 and 29.92 percentage points while reducing average run time by a factor of 2.74 and 1.1. w.r.t. state-of-the-art 1) pupil detectors and 2) vendor provided pupil trackers, respectively. Overall, *PuReST* outperformed other methods in 81.82% of use cases.

CCS CONCEPTS

• **Computing methodologies** → **Object detection**; *Object recognition*; *Image processing*; Shape analysis;

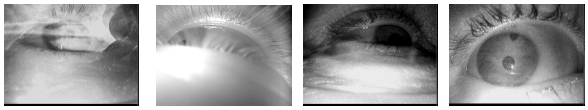
KEYWORDS

Pupil tracking, pupil detection, eye tracking, pervasive, embedded, real-time, open source

ACM Reference Format:
Thiago Santini, Wolfgang Fuhl, and Enkelejda Kasneci. 2018. PuReST: Robust Pupil Tracking for Real-Time Pervasive Eye Tracking. In *ETRA '18: ETRA '18: 2018 Symposium on Eye Tracking Research and Applications, June 14–17, 2018, Warsaw, Poland*. ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3204493.3204578

1 INTRODUCTION

Pupil detection is the fundamental layer in the eye-tracking stack as virtually all subsequent layers rely on the signal generated by this layer – e.g., for calibration [Santini et al. 2017a], gaze estimation [Morimoto and Mimica 2005], model construction [Świrski and Dodgson 2013], and automatic identification of eye movements [Santini et al. 2016]. As a result, errors in the pupil detection layer propagate to subsequent layers, systematically degrading eye-tracking performance. Unfortunately, robust real-time pupil detection in natural environments remains an elusive challenge. An elusiveness that is evidenced by several reports of difficulties and low pupil detection rates in natural environments such as driving [Chu et al. 2010; Kasneci 2013; Kübler et al. 2015; Schmidt et al. 2017; Trösterer et al. 2014; Wood et al. 2017], museum visit [Santini et al. 2018a], shopping [Kasneci et al. 2014], walking [Foulsham et al. 2011; Sugano and Bulling 2015], in an operating room [Tien et al. 2015], and during human-robot interaction [Aronson et al. 2018]. These difficulties in pupil detection stems from multiple factors; for instance, reflections (Fig. 1a), occlusions (Fig. 1b), complex illuminations (Fig. 1c), and physiological irregularities (Fig. 1d) [Fuhl et al. 2016c; Hansen and Hammoud 2007; Hansen and Pece 2005].



(a) (b) (c) (d)

Figure 1: Examples of pupil detection challenges in real-world scenarios: (a) reflections, (b) occlusions, (c) complex illuminations, and (d) physiological irregularities.

With the recent pervasive increase in the adoption of video-based head-mounted eye trackers, these challenges have become progressively more prevalent and relevant. As a result, a plethora of pupil detection methods have been proposed lately to try and alleviate low pupil detection rates – e.g., [Fuhl et al. 2015, 2017, 2016a,b; Javadi et al. 2015; Othman et al. 2017; Santini et al. 2018b; Świrski et al. 2012; Timm and Barth 2011; Vera-Olmos and Malpica 2017]. From these algorithms, *PuRe* [Santini et al. 2018b] stands out for four particular reasons: *PuRe* 1) is currently the best performing algorithmic approach, 2) detects the pupil outline (in contrast to pupil center only), 3) provides a *meaningful* confidence metric, and 4) runs in real-time even for high frame rates (higher than 120 Hz). In this work, we build on top of *PuRe*, using it as a seeding *detection* step, and propose a novel *tracking* method that exploits the *spatiality* and *shape* of the previously estimated pupil in order to detect the pupil in the subsequent frame, Throughout this work, we refer to this novel combination as *PuReST* (Pupil Reconstructor and Subsequent Tracking). The following sections describe and evaluate *PuReST* against state-of-the-art methods, showing improvements in terms of detection rate and run time. A reference open-source C++ implementation, which has also been integrated into *EyeRecToo* [Santini et al. 2017b], is available at www.ti.uni-tuebingen.de/perception.

ETRA '18, June 14–17, 2018, Warsaw, Poland

ACM ISBN 978-1-4503-5706-7/18/06…$15.00
https://doi.org/10.1145/3204493.3204578

2 PROPOSED METHOD

PuReST consists of three distinct parts orchestrated to produce a fast and robust pupil tracking algorithm. When no *reliable* pupil information from the previous frame is available, *PuReST* employs *PuRe* [Santini et al. 2018b] to find a *seed* pupil estimate. Otherwise, the search space is spatially constrained to a square region of interest (ROI) centered at the *previous pupil*'s center. This ROI's size can be tuned to cover a precise range of eye movements if the inter-frame period and eye position w.r.t. the camera are known[1] – e.g., by modeling the pupil movement range [Świrski and Dodgson 2013]. In this work, we take a more straightforward approach by using an adaptive ROI with lateral equal to twice the *previous pupil*'s major axis. Within this ROI, two methods attempt to track the *previous pupil*. The first method's (*outline tracker*) goal is to locate the pupil during *fixations*, *slow smooth pursuits / vestibulo-ocular reflexes*, and *micro saccades* by evaluating the alignment between the *previous pupil*'s outline and the edges lying in a small band around this outline. The second method's (*greedy tracker*) goal is to *greedily* combine *good* edge segments to reconstruct and detect the pupil when it has moved from the previous location – e.g., during *saccades* and *smooth pursuits*. These methods are described in detail in the sequence.

2.1 Initial Pupil Detection

As previously mentioned, we employ *PuRe* [Santini et al. 2018b] to perform an initial pupil detection. *PuRe* is an edge-based pupil detection method that approaches the task by 1) selecting curved edge segments that are likely to belong to the pupil outline, 2) conditionally combining segments pair-wise to construct further candidates, 3) fitting an ellipse to the candidates, and 4) producing a confidence measure for each candidate based on the ellipse aspect ratio, angular edge spread w.r.t. the ellipse, and ratio of outline points whose inner-to-outer contrast supports the hypothesis of the ellipse being a pupil. Naturally, the candidate with the highest confidence is taken as pupil estimate. [Santini et al. 2018b] suggest a *cut-off threshold* ($\tau_{confidence}$) of 0.66 for this confidence metric, which we have adopted. Therefore, whenever a pupil with enough confidence is detected, the subsequent frame pupil detection is performed with the tracking methods.

2.2 Shared Tracking Preamble

Since both the *outline tracker* and the *greedy tracker* operate on edge images using information from histogram analysis, *PuReST* starts with a shared preamble that generates all data common to both methods. The first step of this preamble is to downscale the ROI image if necessary. We have chosen a maximum working size of 100×100 px empirically[2]. It is worth noticing that when downscaling happens, it has a denoising effect, and the results must be upscaled to the original size, introducing an intrinsic error. Afterwards, edges are extracted using a *Canny* edge operator [Canny 1986]. The resulting edge image is then manipulated with morphological operations to thin and straighten edges as well as break up orthogonal connections following the procedure described by Fuhl et. al [Fuhl et al. 2016b]. Additionally, the histogram analysis establishes two masks: *dark* and *bright*. The *dark* mask assumes that the pupil is the darkest region in the ROI and tries to estimate this region. The threshold for this mask is found iteratively by accumulating the histogram counts – starting at the darkest value – until the accumulated pixel count is larger than the *previous pupil* area. The resulting binary image is then morphologically *closed* to lessen spurious regions that might result from small reflections or eyelashes. The *bright* mask aims at identifying glints and small reflections. A brightness threshold is first selected by accumulating histogram counts – starting at the brightest value – until the accumulated pixel count is larger than 5% of the ROI area. To cover the outskirts of corneal reflections, the resulting threshold is further decreased by a bias of five, and the resulting binary image is morphologically *dilated*. In our implementation, the aforementioned morphological operations employ an elliptical 7×7 kernel.

2.3 Outline Tracker

First, edges outside of the *dark* mask and edges inside the *bright* mask are removed to lessen the influence from edges that belong to reflections and near-pupil iris features. The resulting edge image is then intersected with a mask of the *previous pupil*'s outline (width=5 px), and an ellipse is fit[3] to the remaining pixels. Afterwards, the *alignment ratio* between the intersected edges and the *previous pupil*'s outline circumference is calculated, akin to the metric proposed by [Prasad and Leung 2012][4]. If the ratio is below a minimum alignment threshold ($\tau_{align} = 0.65$), the *outline tracker* gives up. Otherwise, the edge pixel intersection and *alignment ratio* procedures are repeated using a new ellipse fit to the initially intersected edges. The edges resulting from this second iteration are then used to fit a final pupil outline candidate, and a confidence measure is established following *PuRe*'s method.

Note that this procedure might be dangerous: In its ingenuity, it consumes any edges that lie within the pupil outline enclosing band assuming these edges to belong to the pupil outline. Hence, the reasoning for eliminating edges belonging to reflections. Nonetheless, spurious edges from the eyelids and eyelashes might still remain and cause the outline tracker to attach to these edges. To prevent such behavior, the *outline tracker* keeps track of the initial pupil seed throughout consecutive outline trackings and breaks the tracking if the major axis of the estimated pupil is larger than 1.05 times the seed pupil's major axis. Consequently, the *outline tracker* does not track continuously dilating pupils by design.

2.4 Greedy Tracker

The *greedy tracker* starts by clustering the edge pixels into connected segments using the topological structural analysis proposed by Suzuki et. al [Suzuki et al. 1985]. To remove significantly plain shapes, the segments are approximated with polylines using the

[1] Note that one can use the ROI size to trade-off covered range and run time.

[2] This scale was chosen based on run time measurements from three mobile ultra low-power CPUs (Intel® Core™ i7-4510U, i7-4600U, and i5-6300U). Such processors power devices that might be effortlessly carried around but are powerful enough to run a fully fledged eye tracking framework at high frame rates ($\approx$ 120 Hz) – e.g., *EyeRecToo* [Santini et al. 2017b] and Pupil Labs Capture [Kassner et al. 2014]), making them excellent candidates for pervasive real-time eye tracking platforms.

[3] Throughout this work, we employed the least-squares ellipse fitting method proposed by Fitzgibbon and Fisher [Fitzgibbon and Fisher 1995]

[4] The alignment ratio measures the ratio between edges and ellipse circumference. We saturate this metric at value one since it might result larger than that due to discretization and the nature of the edge operator.

Douglas-Pecker algorithm [Douglas and Peucker 1973] ($\epsilon = 1.5$), and approximations with three or less points are discarded. Afterwards, *good* segments are identified. In this context, we define *good* candidates as those segments whose majority of pixels fall within the *dark* mask. The following step gives the *greedy tracker* its name: this tracker *greedily* generates all possible segment combinations *without repetition*. Therefore, for N initial segments, the number of candidates evaluated by the *greedy tracker* is $\sum_{i=1}^{N} \binom{N}{i}$. In practice, however, the number of combinations quickly becomes unfeasible to process in a timely manner. For this reason, we sort the candidates according to their diameter, which is evaluated as the largest distance between two of the segment's points, and use only the largest five segments as seeds for the combination process. Subsequently, the convex hull [Sklansky 1982] of each candidate is calculated so that straight lines within curved segments are simplified, and an ellipse is fit to the hull points. A confidence for the candidate is calculated using *PuRe*'s method. In this process, there are four candidate discarding mechanisms: 1) if the ellipse fit is not possible, 2) if the ellipse major axis is smaller than *PuRe*'s minimum pupil size (pd_{min}), 3) if the ellipse aspect ratio is smaller than *PuRe*'s ratio threshold (R_{th}), and 4) if the pupil confidence is smaller than the confidence cut-off threshold ($\tau_{confidence}$). If the *greedy tracker* finds a pupil, the new estimate is considered a new pupil seed. Otherwise, *PuReST* falls back to detecting using *PuRe*.

3 EXPERIMENTAL EVALUATION

Before we commence, it is worth clarifying the distinction we make between a pupil *detector* and a *tracker*. Although both aim at locating the pupil in an image, *detectors* use the information of a single frame, whereas *trackers* use information from the current and previous frames. *PuReST* is a pupil tracker, and, thus, we focus on evaluating it against approaches from the same class. Nevertheless, *PuRe* was also included in this evaluation as a representative from the *detector* class. *PuRe* was chosen since it is the base for *PuReST* and also the current best performing real-time detector. For a description of *PuRe*, please consult Section 2.1. For the *tracker* class, we initially considered Starburst [Li et al. 2005], but its detector performance was found to be particularly low (< 15%, see [Fuhl et al. 2015, 2016b,c; Kassner et al. 2014]). Thus, instead we settled for two state-of-the-art methods with default parameters provided as part of the Pupil (v1.1) software platform [Kassner et al. 2014], which is officially supported by Pupil Labs [Pupil Labs 2017a], to compare against. These are briefly described in the sequence.

The *Pupil Labs 2D* tracker uses the *Pupil Labs detector* to first locate the pupil outline. This detector uses a center-surround to estimate a coarse location for the pupil, which is used as ROI for the remaining of the algorithm. Afterwards, the lowest and highest spikes in this ROI's intensity histogram are located. Dark areas are defined using an offset from this lowest spike, and areas with intensity above the highest threshold are considered spectral reflections. Edges are detected within the ROI, and those outside of the dark areas or inside spectral reflections are discarded, resulting in *selected edges*. These selected edges are extracted into contours and split using a curvature continuity criteria. Candidate pupil ellipses are found using ellipse fitting, and the final ellipse fit is found through an augmented combinational search [Kassner et al. 2014]. The tracker stage is integrated after the *selected edges* step and tracks the pupil outline by considering edges that support the pupil outline based on their distance to the outline ellipse[5].

The *Pupil Labs 3D* tracker builds on top of the *Pupil Labs 2D*, augmenting it with 3D eye model information derived similarly to the approach proposed by Swirski et. al [Świrski and Dodgson 2013]. Whenever the evidence from the *Pupil Labs 2D* tracker is considered weak, constraints from competing eye models are used to robustly fit the pupil [Pupil Labs 2017b].

For this evaluation, we employed five data sets acquired with three distinct head-mounted eye tracking devices, namely, the *Świrski* [Świrski et al. 2012], *ExCuSe* [Fuhl et al. 2015], *ElSe* [Fuhl et al. 2016b], *LPW* [Tonsen et al. 2016], and *PupilNet* [Fuhl et al. 2016a] data sets. In total, these data sets contain 266,786 realistic and challenging images, encompassing 99 distinct *use cases* – i.e., 99 individual eye videos.

3.1 Pupil Detection Rate

A pupil is considered detected if the algorithm's pupil center estimate lies within a radius of n pixels from the ground-truth pupil center. Similar to previous work, we use $n = 5$ to account for small deviations in the ground-truth annotation process [Fuhl et al. 2015, 2016b; Santini et al. 2018b; Tonsen et al. 2016; Vera-Olmos and Malpica 2017]. As can be seen on the left side of Fig. 2, *PuReST* surpassed other algorithms for all pixel errors, improving the detection rate at the 5 px mark by 5.44 and 29.92 percentage points w.r.t. *PuRe* and the Pupil Labs algorithms, respectively. The right side of this figure indicates the generality of *PuReST*, showing that the proposed method reaches detection rates above 87.62% for the majority of the individual uses cases. Furthermore, Fig. 3 contrasts *PuReST* with the *rival* (i.e., the best performer from the other algorithms) in a use case granularity level. At this level, *PuReST* outperformed other approaches in 81.82% of use cases, where as *PuRe*, *Pupil Labs 3D*, and *Pupil Labs 2D* were the best performers in 13.13%, 3.03%, and 2.02%, respectively.

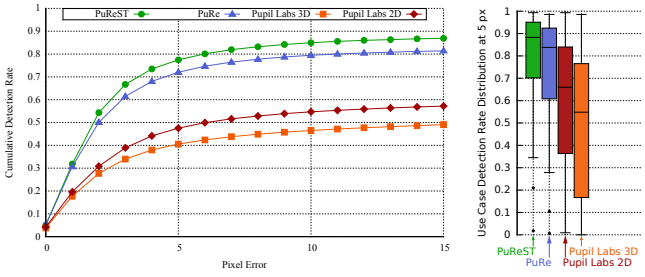


Figure 2: On the left, the cumulative detection rate for the aggregated 266,786 images from all data sets. On the right, the distribution of the detection rate per *use case* as a *Tukey boxplot* [Frigge et al. 1989].

[5]We could not find reviewed references to this tracking stage; please consult the *strong prior* part in https://github.com/pupil-labs/pupil/blob/v1.1/pupil_src/shared_modules/pupil_detectors/detect_2d.hpp#L189 for further details.

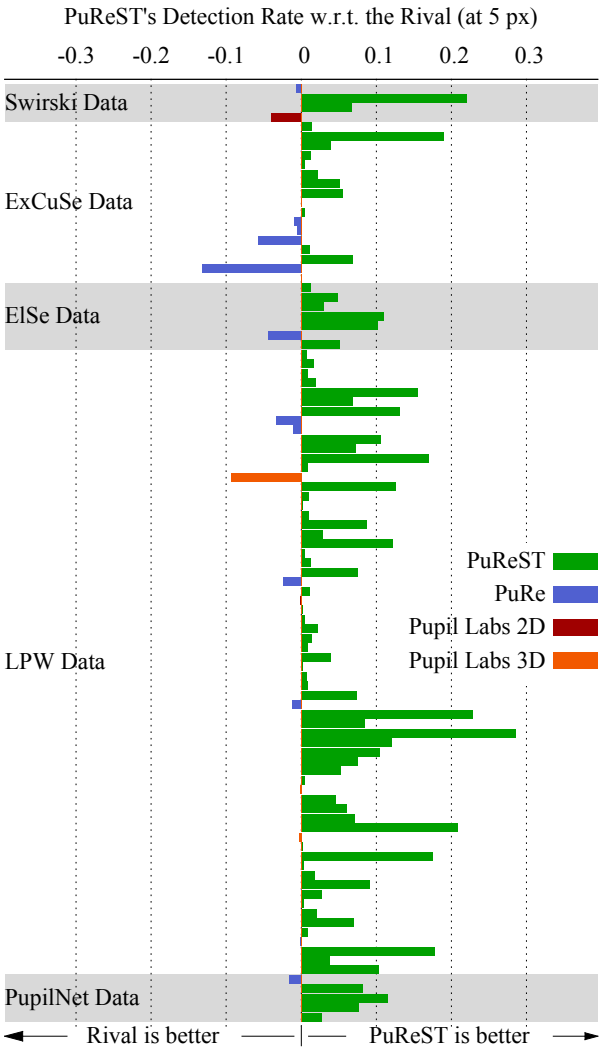


Figure 3: *PuReST* w.r.t. to the rival; each line within a data set represents a distinct *use case*. *PuReST* is the best algorithm in 81.82% of cases, *PuRe* in 13.13%, *Pupil Labs 3D* in 3.03%, and *Pupil Labs 2D* in 2.02%.

3.2 Run Time

A key requirement for real-time gaze-based applications, such as human-computer interaction, is the algorithm run time. In particular, as faster cameras become more accessible, the challenge of meeting the imposed processing deadline grows. For instance, Pupil Labs recently released a new camera capable of 200 fps, which corresponds to a slack of 5 ms. This challenge is further increased by the fact that system resources must be shared to process (and sometimes record) video streams from multiple cameras – e.g., two eye cameras and a field camera. We evaluated the algorithms using a Intel® Core™ i5-4590 CPU @ 3.30GHz with 16GB RAM under Windows 8.1, which is similar to systems employed by eye tracker vendors. All algorithms are coded in C++. Fig. 4 shows the resulting run time distribution for the evaluated algorithms. *PuReST* exhibited the fastest average run time (1.89 ms), reducing the average run time by a factor of 2.74 w.r.t. *PuRe* (5.17 ms). The Pupil Labs trackers have a very competitive average run time (*Pupil Labs 2D*≈2.08 ms and *Pupil Labs 3D*≈2.47 ms). Relative to these algorithms, *PuReST*'s run time reduction factor is limited to only ≈ 1.1. It is worth noticing that *PuRe* already struggles and does not meet the required slack, but *PuReST* is the fastest algorithm regardless of using *PuRe*, indicating that the trackers are responsible for the majority of the detections. In fact, the *outline tracker*, *greedy tracker*, and *PuRe* were responsible for 72.55%, 22.47%, and 4.98% of the correctly detected pupils, respectively. In a first glance, this distribution misleadingly indicates that the *outline tracker* is the main *PuReST* driver. However, this distribution is skewed due to the *LPW* data set, which was collected using a slowly moving object, resulting in an overwhelming majority of slow smooth pursuits. When excluding this data set, the distribution is more evenly spread with the *outline tracker*, *greedy tracker*, and *PuRe* contributing 49.32%, 40.81%, and 9.87%, respectively. It is worth noticing the upper bound for run time results for cases in which both trackers fail and *PuRe* must be run; if no pupil is detect to be tracked, run time results similar to *PuRe*.

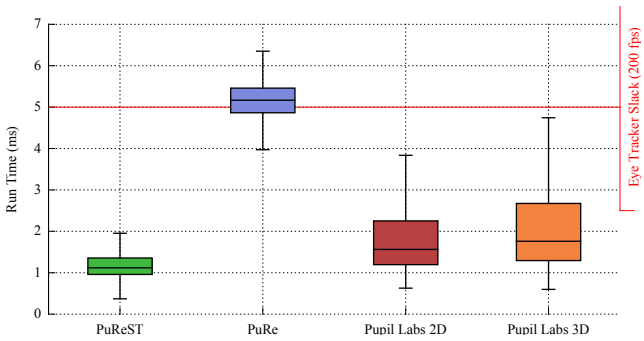


Figure 4: Run time distribution across all evaluation images using a Tukey schematic boxplot; outliers are not shown for the sake of visualization. *PuReST* has an average run time of $\mu = 1.88$ ms and standard deviation $\sigma = 2.19$ ms, whereas the others resulted: *PuRe* ($\mu = 5.17$ ms,$\sigma = 0.51$ ms), *Pupil Labs 2D* ($\mu = 2.08$ ms,$\sigma = 1.65$ ms), and *Pupil Labs 3D* ($\mu = 2.47$ ms,$\sigma = 5.14$ ms).

4 CONCLUSION

In this work, we have proposed and evaluated *PuReST*, a novel algorithm for fast and robust pupil tracking. The proposed method was evaluated on over 266,000 realistic and challenging images acquired with three distinct head-mounted eye tracking devices, increasing pupil detection rate by 5.44 percentage points while reducing average run time by a factor of 2.74 when compared to state-of-the-art pupil detectors. Relative to state-of-the-art pupil trackers provide by a vendor, *PuReST* increases detection rate by 29.92 percentage points while reducing average run time by a factor of 1.1. Overall, *PuReST* outperformed other methods in 81.82% of use cases. While this is our first iteration of *PuReST*, which uses prior knowledge

only from a single past frame, this initial evaluation resulted in such a significant improvement in term of both detection rate and run time that we found it worth to share our findings and a reference open-source implementation with the community already[6]. However, the *tracking* design space offers a wide range of possibilities, and a significant amount of work remains for future work. In particular, we find the approach of the *Pupil Labs 3D* tracker to point in an interesting direction by attempting to derive and employ a 3D eye model to improve pupil detection. Furthermore, the applicability and required modifications to more generic trackers should also be investigated such as Kernelized Correlation Filters [Henriques et al. 2015; Li and Zhu 2014] and Tracking-Learning-Detection [Kalal et al. 2012]. Finally, we hypothesize that the tracking can generate a *more stable* gaze estimation signal given that prior information is taken into account, which we plan to evaluate in the near future.

REFERENCES

Reuben Aronson et al. 2018. Eye-Hand Behavior in Human-Robot Shared Manipulation. In *Proceedings of the 13th Annual ACM/IEEE International Conference on Human Robot Interaction (To appear)*.

John Canny. 1986. A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence* 6 (1986), 679–698.

Byoung Sun Chu et al. 2010. The effect of presbyopic vision corrections on nighttime driving performance. *Investigative ophthalmology & visual science* 51, 9 (2010), 4861–4866.

David H Douglas and Thomas K Peucker. 1973. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica: The International Journal for Geographic Information and Geovisualization* 10, 2 (1973), 112–122.

Andrew W. Fitzgibbon and Robert B. Fisher. 1995. A Buyer's Guide to Conic Fitting. In *Proceedings of the 6th British Conference on Machine Vision (Vol. 2) (BMVC '95)*. BMVA Press, Surrey, UK, UK, 513–522. http://dl.acm.org/citation.cfm?id=243124.243148

Tom Foulsham, Esther Walker, and Alan Kingstone. 2011. The where, what and when of gaze allocation in the lab and the natural environment. *Vision research* 51, 17 (2011), 1920–1931.

Michael Frigge, David C Hoaglin, and Boris Iglewicz. 1989. Some implementations of the boxplot. *The American Statistician* 43, 1 (1989), 50–54.

Wolfgang Fuhl et al. 2015. Excuse: Robust pupil detection in real-world scenarios. In *International Conference on Computer Analysis of Images and Patterns*. Springer, 39–51.

Wolfgang Fuhl et al. 2017. PupilNet v2.0: Convolutional Neural Networks for CPU based real time Robust Pupil Detection. (2017).

Wolfgang Fuhl, Thiago Santini, Gjergji Kasneci, and Enkelejda Kasneci. 2016a. PupilNet: convolutional neural networks for robust pupil detection. *arXiv preprint arXiv:1601.04902* (2016).

Wolfgang Fuhl, Thiago C Santini, Thomas Kübler, and Enkelejda Kasneci. 2016b. Else: Ellipse selection for robust pupil detection in real-world environments. In *Proceedings of the Ninth Biennial ACM Symposium on Eye Tracking Research & Applications*. ACM, 123–130.

Wolfgang Fuhl, Marc Tonsen, Andreas Bulling, and Enkelejda Kasneci. 2016c. Pupil detection for head-mounted eye tracking in the wild: an evaluation of the state of the art. *Machine Vision and Applications* 27, 8 (2016), 1275–1288.

Dan Witzner Hansen and Riad I Hammoud. 2007. An improved likelihood model for eye tracking. *Computer Vision and Image Understanding* 106, 2 (2007), 220–230.

Dan Witzner Hansen and Arthur EC Pece. 2005. Eye tracking in the wild. *Computer Vision and Image Understanding* 98, 1 (2005), 155–181.

João F Henriques, Rui Caseiro, Pedro Martins, and Jorge Batista. 2015. High-speed tracking with kernelized correlation filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 37, 3 (2015), 583–596.

Amir-Homayoun Javadi, Zahra Hakimi, Morteza Barati, Vincent Walsh, and Lili Tcheang. 2015. SET: a pupil detection method using sinusoidal approximation. *Frontiers in neuroengineering* 8 (2015).

Zdenek Kalal et al. 2012. Tracking-learning-detection. *IEEE transactions on pattern analysis and machine intelligence* 34, 7 (2012), 1409–1422.

Enkelejda Kasneci. 2013. *Towards the automated recognition of assistance need for drivers with impaired visual field.* Ph.D. Dissertation. Universität Tübingen, Germany.

Enkelejda Kasneci et al. 2014. Homonymous Visual Field Loss and Its Impact on Visual Exploration: A Supermarket Study. *Translational vision science & technology* 3, 6 (2014), 2–2.

Moritz Kassner, William Patera, and Andreas Bulling. 2014. Pupil: an open source platform for pervasive eye tracking and mobile gaze-based interaction. In *Proceedings of the 2014 ACM international joint conference on pervasive and ubiquitous computing: Adjunct publication*. ACM, 1151–1160.

Thomas C Kübler et al. 2015. Driving with glaucoma: task performance and gaze movements. *Optometry & Vision Science* 92, 11 (2015), 1037–1046.

Dongheng Li, David Winfield, and Derrick J Parkhurst. 2005. Starburst: A hybrid algorithm for video-based eye tracking combining feature-based and model-based approaches. In *Computer Vision and Pattern Recognition-Workshops, 2005. CVPR Workshops. IEEE Computer Society Conference on*. IEEE, 79–79.

Yang Li and Jianke Zhu. 2014. A Scale Adaptive Kernel Correlation Filter Tracker with Feature Integration.. In *ECCV Workshops (2)*. 254–265.

Carlos H Morimoto and Marcio RM Mimica. 2005. Eye gaze tracking techniques for interactive applications. *Computer vision and image understanding* 98, 1 (2005), 4–24.

Mohammad Othman et al. 2017. CrowdEyes: Crowdsourcing for Robust Real-World Mobile Eye Tracking. (2017).

Dilip K Prasad and Maylor KH Leung. 2012. Methods for ellipse detection from edge maps of real images. In *Machine Vision-Applications and Systems*. InTech.

Pupil Labs. 2017a. Accessed in 2017-12-26. (2017). https://pupil-labs.com/

Pupil Labs. 2017b. Accessed in 2017-12-26. (2017). https://pupil-labs.com/blog/2016-03/pupil-v0-7-release-notes/

Thiago Santini, Hanna Brinkmann, Luise Reitstätter, Helmut Leder, Raphael Rosenberg, Wolfgang Rosenstiel, and Enkelejda Kasneci. 2018a. The Art of Pervasive Eye Tracking. In *Proceedings of the 2018 ACM Symposium on Eye Tracking Research & Applications (ETRA) – Adjunct: PETMEI*. ACM.

Thiago Santini, Wolfgang Fuhl, David Geisler, and Enkelejda Kasneci. 2017b. EyeRecToo: Open-Source Software for Real-Time Pervasive Head-Mounted Eye-Tracking. In *Proceedings of the 12th Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*.

Thiago Santini, Wolfgang Fuhl, and Enkelejda Kasneci. 2017a. CalibMe: Fast and Unsupervised Eye Tracker Calibration for Gaze-Based Pervasive Human-Computer Interaction. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. ACM, 2594–2605.

Thiago Santini, Wolfgang Fuhl, and Enkelejda Kasneci. 2018b. PuRe: Robust pupil detection for real-time pervasive eye tracking. *Computer Vision and Image Understanding* (Feb 2018). https://doi.org/10.1016/j.cviu.2018.02.002

Thiago Santini, Wolfgang Fuhl, Thomas Kübler, and Enkelejda Kasneci. 2016. Bayesian identification of fixations, saccades, and smooth pursuits. In *Proceedings of the Ninth Biennial ACM Symposium on Eye Tracking Research & Applications*. ACM, 163–170.

Jürgen Schmidt et al. 2017. Eye blink detection for different driver states in conditionally automated driving and manual driving using EOG and a driver camera. *Behavior Research Methods* (2017), 1–14.

Jack Sklansky. 1982. Finding the convex hull of a simple polygon. *Pattern Recognition Letters* 1, 2 (1982), 79–83.

Yusuke Sugano and Andreas Bulling. 2015. Self-calibrating head-mounted eye trackers using egocentric visual saliency. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*. ACM, 363–372.

Satoshi Suzuki et al. 1985. Topological structural analysis of digitized binary images by border following. *Computer vision, graphics, and image processing* 30, 1 (1985), 32–46.

Lech Świrski, Andreas Bulling, and Neil Dodgson. 2012. Robust real-time pupil tracking in highly off-axis images. In *Proceedings of the Symposium on Eye Tracking Research and Applications*. ACM, 173–176.

Lech Świrski and Neil A. Dodgson. 2013. A fully-automatic, temporal approach to single camera, glint-free 3D eye model fitting [Abstract]. In *Proceedings of ECEM 2013*.

Tony Tien et al. 2015. Differences in gaze behaviour of expert and junior surgeons performing open inguinal hernia repair. *Surgical endoscopy* 29, 2 (2015), 405–413.

Fabian Timm and Erhardt Barth. 2011. Accurate Eye Centre Localisation by Means of Gradients. *Visapp* 11 (2011), 125–130.

Marc Tonsen, Xucong Zhang, Yusuke Sugano, and Andreas Bulling. 2016. Labelled pupils in the wild: a dataset for studying pupil detection in unconstrained environments. In *Proceedings of the Ninth Biennial ACM Symposium on Eye Tracking Research & Applications*. ACM, 139–142.

Sandra Trösterer, Alexander Meschtscherjakov, David Wilfinger, and Manfred Tscheligi. 2014. Eye Tracking in the Car: Challenges in a Dual-Task Scenario on a Test Track. In *Proceedings of the 6th AutomotiveUI*. ACM.

FJ Vera-Olmos and N Malpica. 2017. Deconvolutional Neural Network for Pupil Detection in Real-World Environments. In *International Work-Conference on the Interplay Between Natural and Artificial Computation*. Springer, 223–231.

Joanne M Wood, Richard A Tyrrell, Philippe Lacherez, and Alex A Black. 2017. Nighttime pedestrian conspicuity: effects of clothing on drivers' eye movements. *Ophthalmic and physiological optics* 37, 2 (2017), 184–190.

[6]We provide a C++ reference implementation at www.ti.uni-tuebingen.de/perception