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:

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 Pecc 2005].

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.
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 — e.g., by modeling the pupil movement range [Swirski 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 the 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 $100 \times 100$ px empirically. It is worth noticing that when down-scaling happens, it has a denosing 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 $7 \times 7$ 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 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]. 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 reason 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

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

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., EyeRecTro [Santini et al. 2017b] and Pupil Labs Capture [Kassner et al. 2014], making them excellent candidates for pervasive real-time eye tracking platforms.

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

$^1$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.
The PuRe-ST algorithm [Douglas and Peucker 1973] (r = 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 steps give 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 (pdmin), 3) if the ellipse aspect ratio is smaller than PuRe’s ratio threshold (Rth), and 4) if the pupil confidence is smaller than the confidence cut-off threshold (\( \tau_{\text{confidence}} \)). 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.

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 Siwirski 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 Siwirski [Świrski et al. 2012], ExCuSe [Fuhl et al. 2015], ELSec [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.

---

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_2dhpp4/pupil-detector.md for more details.

---

**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].
PuReST’s Detection Rate w.r.t. the Rival (at 5 px)

-0.3 -0.2 -0.1 0 0.1 0.2 0.3

Swirski Data
ExCuSe Data
ElSe Data
LPW Data
PupilNet Data

Rival is better — PuReST is better

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).](image)

### 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 already6. 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


---

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