ExCuSe: Robust Pupil Detection in Real-World Scenarios

Wolfgang Fuhl(✉), Thomas Kübler, Katrin Sippel, Wolfgang Rosenstiel, and Enkelejda Kasneci

Eberhard Karls Universität Tübingen, 72076 Tübingen, Germany
wolfgang.fuhl@uni-tuebingen.de
http://www.ti.uni-tuebingen.de/

Abstract. The reliable estimation of the pupil position is one the most important prerequisites in gaze-based HMI applications. Despite the rich landscape of image-based methods for pupil extraction, tracking the pupil in real-world images is highly challenging due to variations in the environment (e.g. changing illumination conditions, reflection, etc.), in the eye physiology or due to variations related to further sources of noise (e.g., contact lenses or mascara). We present a novel algorithm for robust pupil detection in real-world scenarios, which is based on edge filtering and oriented histograms calculated via the Angular Integral Projection Function. The evaluation on over 38,000 new, hand-labeled eye images from real-world tasks and 600 images from related work showed an outstanding robustness of our algorithm in comparison to the state-of-the-art. Download link (algorithm and data): https://www.ti.uni-tuebingen.de/Pupil-detection.1827.0.html?\&L=1.

1 Introduction

Eye-tracking technology has helped us getting a deeper understanding of human cognition, answering questions from psychology, medicine, marketing research, and many other disciplines. With the development of mobile, head-mounted eye trackers the number of studies conducted in real-world scenarios, such as in sports, while driving a car, or shopping are increasing. Such eye trackers consist of two or more cameras, recording the subject's eyes from close-up and the scenery from the ego-perspective. The most essential step of the analysis of data recorded by such devices is the accurate identification of the center of the pupil in the camera image. In 2000, Schnipke and Todd [13] reported several difficulties arising in eye-tracking applications, e.g., changing illumination conditions, intersection of eyelashes with the image of the pupil, glasses, etc. Frequent illumination changes are often caused by the ego motion and rotation, especially when moving fast, e.g. while driving. Reflections are caused by a variety of light sources on the subject's eye itself or on glasses or contact lenses. Another factor that may negatively affect the pupil detection rate is the position of the camera that records the subject's eye. For best pupil detection the camera should be positioned directly in front of the subject's eye. Since this would influence the

G. Azzopardi and N. Petkov (Eds.): CAIP 2015, Part I, LNCS 9256, pp. 39–51, 2015.
DOI: 10.1007/978-3-319-23192-1_4

natural viewing behavior, the camera is usually positioned at the borders of the visual field and, consequently, the images recorded are highly off-axial. While in the meantime several of the above problems have been solved for pupil detection under laboratory conditions [3,6–8,10,12,15,19,21], studies employing eye tracking in real-world scenarios regularly report low pupil detection rates [5,9]. Thus, the data collected from such studies has to be processed post-experimentally and the pupil has to be manually labeled in the recorded images. A popular and robust algorithm is *Starburst* by Li et al. [7], which is based on the calculation of edges along rays from an initial guess of the pupil position in the image. Areas of high intensity change along these rays are then used as possible pupil border features. Finally, an ellipse is fitted to these features using RANSAC. In 2012, Swirski et al. [15] proposed a robust pupil tracking algorithm for highly off-axis images. Their approach is based on an initial approximation of the pupil position using Haar Wavelets and a refinement step with RANSAC-based ellipse fitting. Another approach introduced by Goni et al. [3] is based on the bright-pupil technique. In [6] and [8] the algorithm searches for the corneal reflection on IR images. The pupil is expected close to the corneal reflection and extracted using histograms and thresholds-based techniques [6]. In Long et al. [10] IR images are thresholded using a symmetric mass center algorithm. Image thresholding and mass center calculation is also performed in [12], whereas Valenti et al. [19] use isophotes curvature estimation and select the maximum isocenter as pupil center. In [21] the image is thresholded and the curvature of the threshold border is calculated. Despite the above approaches, most eye tracking vendors employ their own pupil detection algorithms, which are specifically tailored for their devices and mostly unpublished. To the best of our knowledge, the above approaches have been evaluated on rather small data sets.

We propose a novel algorithm, **Ex**clusive **Cu**rve **Se**lector or *ExCuSe* for short that is well suited for real-world eye-tracking applications, providing high detection rates and robustness in images where other algorithms fail. Our algorithm is based on oriented histograms calculated via the Angular Integral Projection Function [11]. The coarse pupil center estimation is then refined by ellipse estimation similar to Starburst [7]. The algorithm is evaluated on the Swirski data set [15] as well as nine other data sets that were collected during an on-road driving experiment [5] and eight data sets that were collected during a supermarket study [14]. The evaluation data set consists of overall 38,401 images, where the pupil position was labeled manually on each image. Thus, despite our algorithmic contribution we provide an enormous evaluation data set that may serve as ground truth data for further research.

2 Method

Input to our algorithm are 8-bit gray-scale images. The work-flow of the algorithm for pupil detection is depicted in Figure 1. Each step is described in detail in the following subsections. Furthermore, this paper is accompanied by a supplementary video, which demonstrates the main idea behind each step of ExCuSe and the processing result on real-world eye videos of different subjects.

2.1 Normalization and Histogram Analysis

The peripheral regions of the input image (i.e., 10% in our application) are excluded from further processing in order to avoid the frame of eyeglasses. Furthermore, we assume that on images with an overall bright intensity and similar gray values a reflection on eyeglasses or a bright illumination spot is present. Using an intensity threshold approach to extract the pupil is hard in such cases, since the pupil can not be expected to appear dark and is likely to contain a broad range of intensity values.

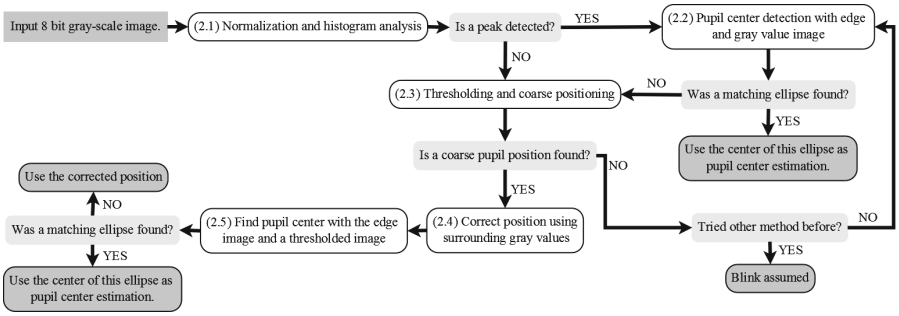


Fig. 1. The algorithmic work-flow in ExCuSe. Light gray boxes represent decisions, dark gray boxes stand for termination points, and white boxes represent processing steps.

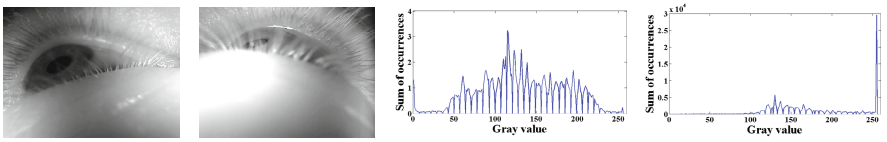
Thus, in a first step, the input image is normalized (range 0 to 255) and a histogram of the image is calculated. Then the algorithm checks whether the histogram contains a peak in the bright area (Figure 2(d)) with a gray value above a threshold th_1 (i.e., $th_1 = 200$ chosen empirically). The peak is detected if a bin in the histogram is higher than a multiple mu_1 of the average image intensity (we chose $mu_1 = 10$ empirically). If such a peak was detected, the pupil can be found based on edge-filtering.

2.2 Pupil Center Detection on Edge and Gray Value Image

We assume that the pupil appears as a curved edge encapsulating the darkest intensity values of the image. To find such an edge, four processing steps are performed on the Canny-edge-filtered image, Figure 3.

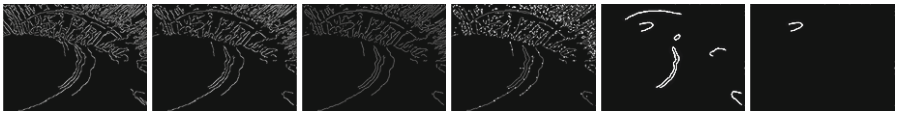
2.2.1 Filtering the Edge Image

Figure 3(a) shows the edge-filtered image from Figure 2(b), which appears cluttered and contains many edges that are not relevant for pupil detection. The pupil edges are difficult to detect since they are crossed by the eyelashes. In a



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

Fig. 2. Figures 2 (a) and (b) show two images from a dataset introduced by Swirski et al. [15] and their corresponding intensity histograms in (c) and (d). Figure 2 (b) shows a pupil with a high range of gray values. Eyelashes cover parts of the pupil and reflect the light.



(a) (b) (c) (d) (e) (f)

Fig. 3. (a) A Canny edge filter is applied to the image from Figure 2(b). (b) all edge pixels with less than two neighbors and angels between all neighbors $\leq 90°$ are removed. (c) the remaining connected edge pixels represent lines. They are thinned and pixels connecting two lines orthogonally are removed. (d) for each line the centroid (shown as white point) is inspected and lines close to their centroid are removed (e). (f) the longest line which contains the darkest pixels is assumed to encapsulate the pupil.

first filtering step, thin edge lines (i.e., 1 pixel thickness) and pixels of small rectangular surfaces (2×2 pixels) are removed. More specifically, the above criterion is fulfilled by neighboring pixels which (considered as vectors) have angels greater than 90° between each other. The remaining edge pixels represent lines which are straight, curved, or consist of both straight and curved parts. The separation step is here of particular interest, e.g., the edge of the eyelashes in the pupil are straight and connected to the curved edge of the pupil. To distinguish between connected line parts that have to be separated and those that do not, the connection point between such parts has to be examined in detail. The assumption is that line parts that have to be separated are orthogonal to each other at the connection point.

To separate lines consisting of curved and straight parts into the corresponding curved and straight segments, the morphologic operations shown in Figure 4 are applied. If one of these patterns matches to the edge pixels (shown in gray), that pixel is deleted. Pixels marked black in the Figure are added. After thinning, lines can be separated into segments by deletion of just one pixel. However, there are still pixels which prevent the patterns from Figure 4(d), (e), or (f) to match. Therefore, lines are straightened using the patterns shown in Figure 4(b) and (c). Now the connection points of line parts which are orthogonal to each

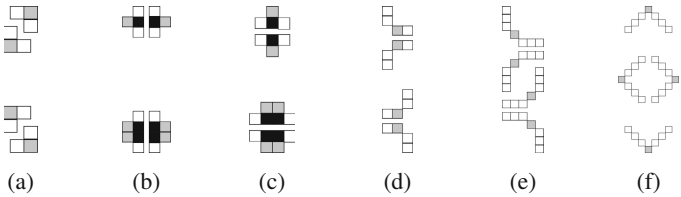


Fig. 4. Morphologic pixel manipulation patterns. White and gray boxes represent pixels that were detected as edges in the image. If the pattern matches an edge segment, gray pixels are removed and black pixels are added to the edge image. Operand (a) thins lines. Operands (b) and (c) are used to straighten lines. (d), (e) and (f) separate straight parts of a line from curved parts.

other can be separated using the patterns in Figure 4(d),(e), and (f). The result of this step is shown in Figure 3(c).

2.2.2 Remove Straight Lines

The next step is to detect and remove straight lines. Since the pupil is expected to be encapsulated in a curved line, straight lines are of no interest. Therefore, all remaining edge pixels are combined to lines based on their connection to neighboring edge pixels. The steps that are performed to calculate lines from the edge image are the following:

1. Find edge pixels that do not belong to any line yet
2. Create a new line with the edge pixel
3. Add all direct neighbor edge pixels to the line
4. Repeat Steps 3 + 4 for all added neighbor pixels

Calculate the line centroid for each line. If the pixel distance between the centroid and at least one point of the line segment is smaller than a threshold di_1, the line is assumed to be straight (in our application we chose $di_1 = 3$ empirically). Figure 3(d) shows all lines with their centroid (white point) and Figure 3(e) shows the remaining, curved lines after the removal of straight segments. We expect that one of the remaining lines belongs to the pupil.

2.2.3 Choose Curved Line

We assume the pupil to be a dark spot in the intensity image. Therefore, the pupil candidate with the darkest area contained in it is most likely to be the pupil. To calculate an intensity value for the contained area we choose the pixel with a distance of di_2 pixels for each line point which have the smallest euclidean distance to the line's centroid (i.e., $di_2 = 2$ chosen empirically). For these pixels, the mean gray value is calculated. It is possible that there is more than one curved line belonging to the pupil. We choose the longest line found with the darkest inboard area. To ensure that larger lines are not discarded, we choose a

range ra_1 in which the mean gray value deemed to be equal (i.e., $ra_1 = 5$). The chosen line is shown in Figure 3(f). All points on this line are collected and the center is estimated using ellipse fitting.

2.2.4 Fit Ellipse

There are basically three ways of fitting an ellipse to a set of points. First, the direct least squares method, which is highly affected by pixels not belonging to the border of the ellipse [2]. The other two possibilities are vote- and search-based (e.g., RANSAC [1]). These are more robust to in- and outliers, yet computationally expensive [20]. We fit the ellipse based on the direct least squares method. It is fast to calculate and also used as abort criterion on failure for the step (2.2) Figure 1.

2.3 Thresholding and Coarse Positioning

If the gray value histogram does not contain a peak (Figure 2(c)), we extract the pupil based on a threshold th_2. Each pixel with a gray value lower than th_2 is set to 255 as shown in Figure 5(b). In highly scattered images the pupil may consist of a range of different intensity values. The threshold th_2 is chosen dependent on the scattering in the image as half the standard deviation of the image intensity. In this step we aim at determining a coarse pupil position. It is not necessary to extract the whole pupil. Therefore a conservative threshold that reduces noise at the potential cost of cutting part of the pupil is preferable. The coarse pupil position is estimated utilizing the Angular Integral Projection Function (AIPF) [11] on the thresholded image. The AIPF allows the calculation of the Integral Projection Function (IPF) for any specified angle.

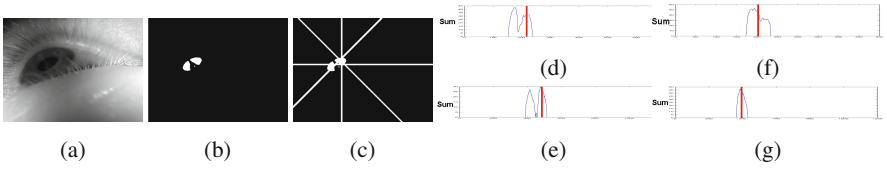


Fig. 5. (a) An image from the Swirski et al. [15] data set and (b) its corresponding thresholded image. In (c) the coarse positioning (white lines) of the four orientations from the Angular Integral Projection Function (AIPF) [11] are shown. The results of the AIPF calculated on the threshold image for the orientations 0°, 45°, 90° and 135° are shown in the histograms (d), (e), (f), and (g) in corresponding order. The chosen positions are shown as red lines and correspond to the white lines in (c) with (d) defining the vertical white line in (c), (e) the line from the right bottom to the top left corner, (f) to the horizontal line and (g) to the line from the left bottom to the top right corner.

$$IPF_h(y) = \int_{x_1}^{x_2} I(x,y)\ \mathrm{d}x \tag{1}$$

$$IPF_v(x) = \int_{y_1}^{y_2} I(x,y)\ \mathrm{d}y \tag{2}$$

With $I(x,y)$ as the gray value at the location (x,y) equation (1) (as found in [11]) defines the IPF_h (Integral Projection Function horizontally) for the interval $[x_1, x_2]$ and equation (2) define the IPF_v (Integral Projection Function vertically) for the interval $[y_1, y_2]$. The IPF calculates the sum of the intensity values of an image in one direction. For example, outgoing form the x-axes for each row (pixel line from the bottom of the image to the top) the pixel values are summed up and represent one bin in the resulting histogram. Those histograms do not rely on shape and our assumption is that the region with the highest response is the pupil. We used the AIPF because it allows to calculate IPFs for different orientations, it is known to be robust and is fast to calculate. Two well known IPFs are the horizontal (IPF_h) and the vertical (IPF_v) IPF. The IPF_v corresponds to the AIPF with angle 0° and the IPF_h corresponds to the AIPF with angle 90° [11].
With $I(x,y)$ as the gray value at location (x,y) equation (3) (as found in [11]) defines the AIPF. Θ is the angle of the line to the x-axis from which the integration rotated by 90° takes place, p is the position on the line or the bin of the corresponding histogram, h is the number of pixels to be integrated and (x_0, y_0) is the position of the start point of the line along which the integration rotated by 90° takes place. We used the orientations 0°, 45°, 90° and 135° for the AIPF to calculate the histograms shown in Figure 5(d),(e),(f), and (g).

$$\begin{aligned} AIPF(\Theta,p,h) = \frac{1}{h+1} * \int_{j=-\frac{h}{2}}^{\frac{h}{2}} I\Big(& (x_0 + p\cos\Theta) \\ & +(j\cos(\Theta+90°)), (y_0 + p\sin\Theta) \\ & +(j\sin(\Theta+90°))\Big)\,\mathrm{d}j \end{aligned} \tag{3}$$

In these four histograms the coarse pupil location is assumed at a wide and high response area. The minimum length of the area is specified by ar_1 and the number of consecutive bins allowed to be low is specified by ar_2 (in our application we chose $ar_1 = 7$ and $ar_2 = 5$ empirically). This is done to eliminate single high responses in the histogram.

Areas of high response are defined by a threshold th_3 which is a percentage of the maximum of the histogram (in our application we chose $th_3 = 0.5$ empirically). If there is more than one acceptable area in a histogram, our assumption is that the pupil can be found at the center of the image. Therefore, the midpoint of the area which is closest to the bin in the histogram corresponding to the center of the image is chosen as the pupil position. The white lines in

Figure 5(c) represent the angle of the AIPF for each histogram rotated by 90° (angle of the integration) and are the chosen positions. Therefore, these white lines correspond to the red lines from Figures 5(d), (e), (f), and (g) drawn to the threshold image shown in Figure 5(b). The pupil position is estimated based on the intersection of these lines. Our assumption is that the intersection of those lines orthogonal to each other are close to or hit the pupil. This way, up to two intersection points are considered. The pupil position is assumed as the point between these intersections. In the case that no intersection was found, branch *2.2* of the algorithm will take over. If this branch fails to detect the pupil as well, a blink is assumed.

2.4 Correct Position Using Surrounding Gray Values

Once a coarse pupil center estimation has been established, it has to be improved because it is possible that the coarse position lays outside or on the boarder of the pupil. It can be refined within a small area ar_3 around the estimation without being dependent on the shape or color of the pupil (in our application we chose $ar_3 = 0.1$ empirically, which represents 10% of the width and height of the image in each direction). The only assumption made is that pixels belonging to the pupil are surrounded by brighter or equally bright pixels. This step is important for images in which the pupil is especially hard to detect.

$$PS(x,y) = \sum_{x_i=x_1}^{x_2} \sum_{y_i=y_1}^{y_2} \begin{cases} I(x,y) - I(x_i,y_i), I(x_i,y_i) < I(x,y) \\ 0, I(x_i,y_i) >= I(x,y) \end{cases} \tag{4}$$

For each pixel the sum $PS(x,y)$ of gray value differences to its neighbors is calculated. Only gray values lower than the value of the pixel under consideration are taken into account. For the neighborhood area the square root of the diagonal of the area specified by ar_3 is used. The mean of the pixel positions with the lowest sum value is the new corrected position. In equation (4) $PS(x,y)$ is the sum calculated for the pixel at position (x,y), $[x_1,x_2]$ is the interval on the x-axis of the neighborhood area, $[y_1,y_2]$ is the interval on the y-axis and $I(x,y)$ is again the gray value at position (x,y).

2.5 Find Pupil Center with the Edge and Threshold Image

This step does not require the corrected position to be the accurate pupil center, however it is required to lie inside of the pupil. The concept of using a threshold image to improve the edge image and refine finding the pupil edges with rays outgoing from this position is described in the following chapter. Only the region ar_4 around the corrected point is of interest for finding the pupil (in our application we chose $ar_4 = 0.2$ empirically, which means 20% of the width and height of the image in each direction). We use only eight rays because for too many rays it is more likely that rays hit edges not belonging to the pupil if the edges belonging to the pupil are not consistently present. Therefore rays missing the pupil edges can hit other edges that do not belong to the pupil thus making the pupil center detection incorrect.

2.5.1 Improve Edge Image with Threshold Image

First, an edge-filter of the image region is calculated, as shown in Figure 6(a). The calculated threshold image from step *2.3* is not useful here because the threshold chosen was for coarse positioning and there was no need to extract the whole pupil. In this step, the threshold th_2 (chosen as half the standard deviation) is increased to the full standard deviation to calculate the new threshold image (Figure 6(b)). In this step it is important that no part of the pupil gets cut off by a too conservative threshold. Edges of the edge image are preselected by overlay with the threshold image. Only edges close to the border of the threshold region (Figure 6(c)) are considered relevant. This border is calculated by accepting only white pixels in the threshold image which have black direct neighbors. Only edges close to the border region are considered, see Figure 6(d). To calculate this the surrounding area ar_5 of each threshold border pixel is inspected (in our application we chose $ar_5 = 5$ empirically which means 5 pixels in each direction outgoing from the threshold border pixel). If an edge pixel lies within the ar_5 region of a threshold border pixel it is accepted. Then the border refinement steps described in *2.2.1* and *2.2.2* are carried out (the result is shown in Figure 6(e)).

(a) (b) (c) (d) (e) (f)

Fig. 6. The edge-filtered eye image (a) of the region where the pupil is expected. A threshold image (b) is calculated to determine the threshold border (c). Only edge pixels close to this border are used for further calculations (d). After the edge refinement steps explained in *2.2.1* and *2.2.2* the remaining edges are used for edge selection (e): rays are sent out from the corrected point (white point in the middle of (f)) into all directions with an angle step of 45°. If a ray hits an edge (white points on the ellipse in (f)) the line belonging to this edge is supposed to belong to the pupil.

2.5.2 Find Edges that Represent the Pupil Border

In the resulting edge image, rays from the corrected position (small white point in the middle of Figure 6(f)) are sent in all directions with an angle step of 45° until they hit an edge (white points on the elliptic line in Figure 6(f)), similar to the method used by the Starburst algorithm [7]. The intersection points between the rays and the edges are used to collect points. All edge pixels connected to a hit edge pixel and iteratively all that are connected to those pixels are used to fit an ellipse.

3 Experimental Evaluation

3.1 Data

We evaluated our approach on eighteen data sets. The first data set was published by Swirski et al. [15]. Nine data sets were recorded during an on-road driving experiment [5] using a head-mounted camera system (Dikablis Mobile Eye Tracker by Ergoneers GmbH). The remaining eight data sets were recorded during a supermarket search task [14]. These data sets are highly challenging, since illumination conditions change often and rapidly. Furthermore reflections on eyeglasses and contact lenses occur. The data reflect the results of standard eye-tracking experiments out of the laboratory and was recorded for other studies that did not focus on pupil detection. Pupil position was hand labeled for all of the above data sets. The data is available for download under https://www.ti.uni-tuebingen.de/Pupil-detection.1827.0.html?&L=1.

3.2 Results

The runtime of ExCuSe (C++ implementation) was 7ms per image (averaged over all images (384 × 288 pixel and 620 × 460 pixel), no multithreading, CPU: i5-4570 3.2GHz). We compared the performance of our algorithm (without changing any parameter for all data sets) to two state-of-the-art approaches, namely Swirski et al. [15] (Version June 1st, 2014 runtime:14ms) and to the Starburst algorithm [7] (Version 1.1.0 from OpenEyes website runtime not comparable because of matlab version). The performance was measured in terms of detection rate for different pixel errors based on the Euclidean distance between the hand-labeled ground-truth and the pupil center as reported by each algorithm. Table 1 shows the results for each algorithm on each data set with a pixel error up to 5.

Note that the performance of the pupil tracking software provided by the eye-tracker manufacturer (Ergoneers GmbH) was very poor on all the above data sets (without Swirski data) with a detection rate of less than 0.027 % for 15 error pixels. Therefore, the results of the manufacturer software will not be

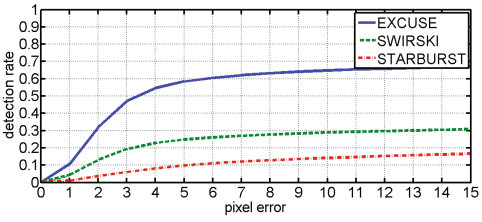


Fig. 7. Average pupil detection rates for different pixel errors achieved by ExCuSe (solid), the Swirksi et al. algorithm (dashed), and Starburst (double dashed) on all data sets from Table 1.

presented here. In addition, Figure 7 shows the average performance of all three algorithms over the entire evaluation data, i.e., on about 39,000 hand-labeled images. As shown in the Figure, ExCuSe outperforms the competitor algorithms with detection rates nearly twice as good as the Swirski et al. algorithm and is therefore a suitable choice for pupil detection on such difficult data.

Table 1. Detection rate of Starburst, Swirski et al. and ExCuSe on each evaluation data set. For each data set, the table shows the number of hand-labeled images and a description of the challenges faced by the image processing as seen by the authors. The last three columns are the results up to an pixel error of 5.

Data Set	Images	Challenges	Starburst	Swirski et al.	ExCuSe
Swirski	600	Highly off axis, eyelashes	21%	78%	**86%**
I	6.554	Reflections	5%	5%	**70%**
II	505	Reflections, bad illumination	2%	24%	**34%**
III	9.799	Reflections, recording errors, bad illumination	1%	6%	**39%**
IV	2.655	Contact lenses, bad illumination	4%	34%	**81%**
V	2.135	Shifted contact lenses	14%	**77%**	**77%**
VI	4.400	Bad illumination, Mascara	18%	19%	**53%**
VII	4.890	Bad illumination, mascara, eyeshadow	2%	39%	**46%**
VIII	630	Bad illumination, Eyelashes	8%	41%	**56%**
IX	2.831	Reflections, additional black dot	12%	23%	**74%**
X	840	Bad illumination, pupil at image boarder	53%	29%	**79%**
XI	655	Reflections, bad illumination, additional black dot	26%	20%	**56%**
XII	524	Bad illumination	61%	70%	**79%**
XIII	491	Bad illumination, Eyelashes	43%	61%	**70%**
XIV	469	Bad illumination	21%	52%	**57%**
XV	363	Shifted contact lenses	8%	**62%**	52%
XVI	392	Mascara, eyelashes	8%	18%	**49%**
XVII	268	Bad illumination, eyelashes	0%	68%	**78%**

4 Conclusions

We presented a pupil detection algorithm, ExCuSe, for application in real-world eye-tracking experiments. The focus was primarily on the robustness of the algorithm with respect to frequently and rapidly changing illumination conditions, off-axial camera position, and other sources of noise. We evaluated our algorithm on a total of 39,001 eye images in comparison with two state-of-the-art approaches. Our method showed high robustness and clearly outperformed the competitor algorithms. Since robust pupil position tracking under real-world illumination conditions is a crucial prerequisite towards online analysis of eye-tracking data in different applications, e.g., driving, where such information can be used to determine the visual attention focus of the driver [4,16,17], we encourage the application of ExCuSe in such tasks. In our future work, we will integrate ExCuSe in visual search tools (e.g., Vishnoo [18]) to make it available for pupil detection in search tasks under laboratory conditions.

References

1. Fischler, M.A., Bolles, R.C.: Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Communications of the ACM **24**(6), 381–395 (1981)
2. Fitzgibbon, A., Pilu, M., Fisher, R.B.: Direct least square fitting of ellipses. IEEE Transactions on Pattern Analysis and Machine Intelligence **21**(5), 476–480 (1999)
3. Goni, S., Echeto, J., Villanueva, A., Cabeza, R.: Robust algorithm for pupil-glint vector detection in a video-oculography eyetracking system. In: Pattern Recognition. ICPR 2004, vol. 4, pp. 941–944. IEEE (2004)
4. Kasneci, E.: Towards the Automated Recognition of Assistance Need for Drivers with Impaired Visual Field. Ph.D. thesis, University of Tübingen, Wilhelmstr. 32, 72074 Tübingen (2013)
5. Kasneci, E., Sippel, K., Aehling, K., Heister, M., Rosenstiel, W., Schiefer, U., Papageorgiou, E.: Driving with Binocular Visual Field Loss? A Study on a Supervised On-road Parcours with Simultaneous Eye and Head Tracking. Plos One (2014). doi:10.1371/journal.pone.0087470
6. Keil, A., Albuquerque, G., Berger, K., Magnor, M.A.: Real-time gaze tracking with a consumer-grade video camera
7. Li, D., Winfield, D., Parkhurst, D.J.: Starburst: a hybrid algorithm for video-based eye tracking combining feature-based and model-based approaches. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition-Workshops, 2005. CVPR Workshops, pp. 79–79. IEEE (2005)
8. Lin, L., Pan, L., Wei, L., Yu, L.: A robust and accurate detection of pupil images. In: 2010 3rd International Conference on Biomedical Engineering and Informatics (BMEI), vol. 1, pp. 70–74. IEEE (2010)
9. Liu, X., Xu, F., Fujimura, K.: Real-time eye detection and tracking for driver observation under various light conditions. In: IEEE Intelligent Vehicle Symposium, 2002, vol. 2, pp. 344–351. IEEE (2002)
10. Long, X., Tonguz, O.K., Kiderman, A.: A high speed eye tracking system with robust pupil center estimation algorithm. In: 29th Annual International Conference of the IEEE on Engineering in Medicine and Biology Society. EMBS 2007, pp. 3331–3334. IEEE (2007)
11. Mohammed, G.J., Hong, B.R., Jarjes, A.A.: Accurate pupil features extraction based on new projection function. Computing and Informatics **29**(4), 663–680 (2012)
12. Peréz, A., Cordoba, M., Garcia, A., Méndez, R., Munoz, M., Pedraza, J.L., Sanchez, F.: A precise eye-gaze detection and tracking system
13. Schnipke, S.K., Todd, M.W.: Trials and tribulations of using an eye-tracking system. In: CHI 2000 extended abstracts on Human factors in computing systems, pp. 273–274. ACM (2000)
14. Sippel, K., Kasneci, E., Aehling, K., Heister, M., Rosenstiel, W., Schiefer, U., Papageorgiou, E.: Binocular Glaucomatous Visual Field Loss and Its Impact on Visual Exploration - A Supermarket Study. PLoS ONE **9**(8), e106089 (2014)

15. Świrski, L., Bulling, A., Dodgson, N.: Robust real-time pupil tracking in highly off-axis images. In: Proceedings of the Symposium on Eye Tracking Research and Applications, pp. 173–176. ACM (2012)
16. Tafaj, E., Kasneci, G., Rosenstiel, W., Bogdan, M.: Bayesian online clustering of eye movement data. In: Proceedings of the Symposium on Eye Tracking Research and Applications, ETRA 2012, pp. 285–288. ACM (2012)
17. Tafaj, E., Kübler, T.C., Kasneci, G., Rosenstiel, W., Bogdan, M.: Online classification of eye tracking data for automated analysis of traffic hazard perception. In: Mladenov, V., Koprinkova-Hristova, P., Palm, G., Villa, A.E.P., Appollini, B., Kasabov, N. (eds.) ICANN 2013. LNCS, vol. 8131, pp. 442–450. Springer, Heidelberg (2013)
18. Tafaj, E., Kübler, T., Peter, J., Schiefer, U., Bogdan, M., Rosenstiel, W.: Vishnoo - an open-source software for vision research. In: Proceedings of the 24th IEEE International Symposium on Computer-Based Medical Systems, CBMS 2011, pp. 1–6. IEEE (2011)
19. Valenti, R., Gevers, T.: Accurate eye center location through invariant isocentric patterns. Transactions on pattern analysis and machine intelligence **34**(9), 1785–1798 (2012)
20. Yuen, H., Illingworth, J., Kittler, J. Ellipse detection using the hough transform. In: Alvey Vision Conference, pp. 1–8 (1988)
21. Zhu, D., Moore, S.T., Raphan, T.: Robust pupil center detection using a curvature algorithm. Computer methods and programs in biomedicine **59**(3), 145–157 (1999)