ElSe: Ellipse Selection for Robust Pupil Detection in Real-World Environments

Wolfgang Fuhl[*]
Perception Engineering
University of Tübingen

Thiago C. Santini[†]
Perception Engineering
University of Tübingen

Thomas Kübler[‡]
Perception Engineering
University of Tübingen

Enkelejda Kasneci[§]
Perception Engineering
University of Tübingen

Abstract

Fast and robust pupil detection is an essential prerequisite for video-based eye-tracking in real-world settings. Several algorithms for image-based pupil detection have been proposed in the past, their applicability, however, is mostly limited to laboratory conditions. In real-world scenarios, automated pupil detection has to face various challenges, such as illumination changes, reflections (on glasses), make-up, non-centered eye recording, and physiological eye characteristics. We propose *ElSe*, a novel algorithm based on ellipse evaluation of a filtered edge image. We aim at a robust, inexpensive approach that can be integrated in embedded architectures, e.g., driving. The proposed algorithm was evaluated against four state-of-the-art methods on over 93,000 hand-labeled images from which 55,000 are new eye images contributed by this work. On average, the proposed method achieved a 14.53% improvement on the detection rate relative to the best state-of-the-art performer. Algorithm and data sets are available for download: *ftp://emmapupildata@messor.informatik.uni-tuebingen.de* (password:*eyedata*).

Keywords: Pupil detection, Eye tracking, Pupil data set

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

1 Introduction

Over the last years head-mounted, mobile eye trackers enabled the assessment of the human viewing behavior in real-world and dynamic tasks. Such eye trackers map the gaze point of the scene based on the center of the automatically detected pupil in the eye images. Whereas tracking can be accomplished successfully under laboratory conditions, many studies report the occurrence of difficulties when eye trackers are employed in natural environments, such as driving [Kasneci 2013; Liu et al. 2002] or shopping [Kasneci et al. 2014b; Sippel et al. 2014]. The main source of noise in such experimental tasks is a non-robust pupil signal that is mainly related to challenges in the image-based detection of the pupil. [Schnipke and Todd 2000] summarized a variety of difficulties occurring when using eye trackers, such as changing illumination, motion blur, recording errors and eyelashes covering the pupil. Rapidly changing illumination conditions arise primarily in tasks where the subject is moving fast, e.g., while driving, or rotates relative to unequally distributed light sources. Furthermore, in case the subject is wearing eye glasses or contact lenses, further reflections may occur. Another issue arises due to the off-axial position of eye camera in head-mounted eye trackers. Therefore, studies based on eye tracking outside of laboratory constantly report low pupil detection rates [Kasneci et al. 2014a; Liu et al. 2002; Trösterer et al. 2014]. As a consequence, the data collected in such studies has to be manually post-processed, which is a laborious and time-consuming procedure. Furthermore, this post-processing is impossible for real-time applications that rely on the pupil monitoring (e.g., driving or surgery assistance). Such real-time applications also impose harsh constraints on the algorithm, making the use of computer-intensive methods impracticable and leading to the prevalence of threshold-based methods.

Several algorithms address image-based pupil detection under laboratory conditions. For example, [Goni et al. 2004] use a histogram-based threshold calculation on bright pupils. A similar approach was introduced by [Keil et al. 2010], where a corneal reflection detection was performed on top of a histogram-based method. Such algorithms can be applied to eye images captured under infrared light as in [Lin et al. 2010] and [Long et al. 2007]. In [Long et al. 2007] and [Peréz et al. 2003], the center of the pupil is estimated based on a threshold and a center of mass calculation. Another threshold-based approach was presented in [Zhu et al. 1999], where the pupil is detected based on the calculation of the curvature of the threshold border. A similar approach is also used in the recently published algorithm SET [Javadi et al. 2015], which first extracts pupil pixels based on a luminance threshold. Afterwards, the shape of the thresholded area is extracted and compared against a sine curve. An isophotes curvature-based approach is presented by [Valenti and Gevers 2012] using the maximum isocenter as pupil center estimation. Despite recent developments the most popular algorithm in this realm is probably Starburst, introduced by [Li et al. 2005]. Starburst sends out rays in multiple directions and collects all positions where the difference of consecutive pixels is higher than a threshold. The mean position is calculated and this step is repeated until convergence. [Świrski et al. 2012] proposed an algorithm starting with a coarse positioning using Haar-like features. The intensity histogram of the coarse position is clustered using k-means followed by a modified RANSAC ellipse fit. [Fuhl et al. 2015] proposed ExCuSe, which was designed with the aforementioned challenges that arise from real-world scenarios in mind. Based on an intensity histogram analysis the algorithm decides whether the input image has reflections or not. On images with reflections, the edge image is filtered and the best curve is selected. Otherwise, a coarse position is estimated and then refined.

In this paper, we present a novel algorithm for pupil detection named *Ellipse Selector* (*ElSe* for short) based on edge filtering, ellipse evaluation, and pupil validation. We evaluated ElSe on over 94,000 images collected during eye-tracking experiments in real-world scenarios. ElSe proved high detection rates, robustness, and a fast runtime in comparison to state-of-the-art algorithms ExCuSe [Fuhl et al. 2015], SET [Javadi et al. 2015], Starburst [Li et al. 2005], and [Świrski et al. 2012]. As an additional contribution, both data set and the annotated pupil centers are openly accessible to support further research.

[*]wolfgang.fuhl@uni-tuebingen.de
[†]thiago.santini@uni-tuebingen.de
[‡]thomas.kuebler@uni-tuebingen.de
[§]enkelejda.kasneci@uni-tuebingen.de

ETRA 2016, March 14 - 17, 2016, Charleston, SC, USA
ISBN: 978-1-4503-4125-7/16/03
DOI: http://dx.doi.org/10.1145/2857491.2857505

2 Method

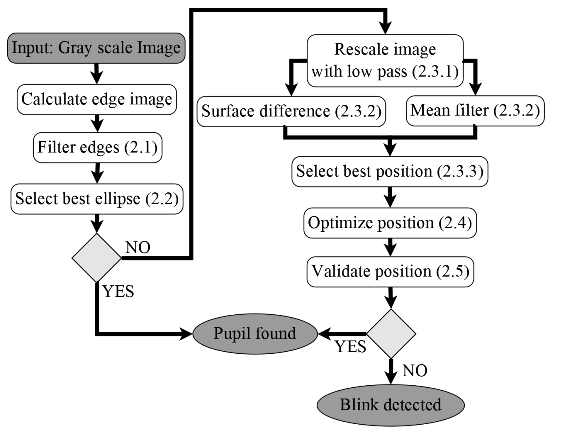

Figure 1: *Flowchart of the proposed algorithm. Light gray boxes represent decisions, dark gray ellipses termination points, and white boxes represent processing steps.*

ElSe operates on gray scale images. After normalization, a Canny edge filter is applied to the eye image (Figure 1).

In the next algorithmic step (Step 2.1 in Figure 1), edge connections that could impair the surrounding edge of the pupil are removed. Afterwards, in Step 2.2, connected edges are collected and evaluated based on straightness, inner intensity value, elliptic properties, the possibility to fit an ellipse to it, and a pupil plausibility check. If a valid ellipse describing the pupil is found, it is returned as the result. In case no ellipse is found (e.g., when the edge filtering does not result in suitable edges), a second analysis is conducted. To speed up the convolution with the surface difference (Step 2.3.2) and mean filter (Step 2.3.2), the image is downscaled (Step 2.3.1). After applying the surface difference and mean filter to the rescaled image, the best position is selected (Step 2.3.3) by multiplying the result of both filters and selecting the maximum position. Choosing a pixel position in the downscaled image leads to a distance error of the pupil center in the full scale image. Therefore, the position has to be optimized on the full scale image (Step 2.4) based on an analysis of the surrounding pixels of the chosen position. In the following sections, each of the above mentioned steps is described in detail.

2.1 Filter edges

Edges are split up at positions that do not occur in an ellipse, e.g., orthogonal connectors and edge points with more than two neighbors. Additionally, edges are thinned and straightened in order to improve the breaking procedure based on two approaches (morphologic and algorithmic). Both approaches lead to comparable results. In the provided implementation, ElSe uses the morphologic approach since it requires less computational power.

2.1.1 Morphologic approach

The employed morphologic operations in Figures 2b, 2c, 2d, and 2e are similar to those introduced in ExCuSe [Fuhl et al. 2015]. However, in contrast to ExCuSe, no preprocessing based on deletion of edges with low angle is performed. Furthermore, we introduce a stable thinning procedure (Figure 2a) and deletion of edges with too many neighbors. The morphologic processing starts with edge-thinning using the pattern shown in Figure 2a. Figure 2j presents the result of thinning applied on the Canny edge image from Figure 2i. Afterwards, the direct neighborhood of each edge pixel is summed up. If this neighborhood is > 2, the edge pixel is deleted because it has joined more than two lines. Applied to the result from the thinning step, Figure 2k shows the remaining edge pixels. Next, a refinement step is performed by applying the straightening patterns in Figure 2b and 2c, yielding the edges in Figure 2l. Then, the patterns shown in Figure 2d, 2e, 2f, and 2g are applied; as a result, the orthogonal connections in consecutive edge pixels are separated by deleting the connecting pixel, resulting in Figure 2m.

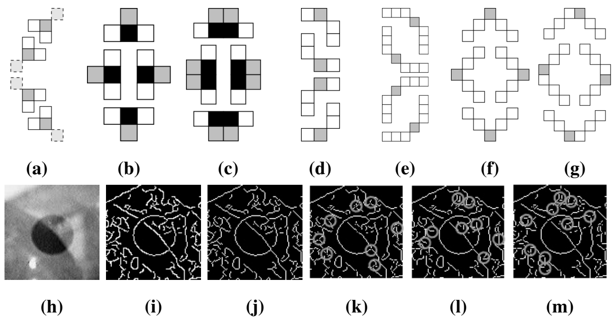

Figure 2: *Morphologic patterns for edge manipulation. White and dark gray boxes represent pixels that have to remain edge pixels. Light gray boxes with dashed borders (a) represent pixels that have to be removed. If the pattern matches a segment in the edge image, pixels under dark gray boxes are removed from, and pixels under black boxes are added to the edge image. The pattern in (a) thins lines, whereas patterns (b) and (c) straightens lines. The patterns (d), (e), (f), and (g) are applied to break up orthogonal connections. (h) input image, (i) edge filtered results, (j) edges after thinning using the morphologic pattern from (a). (k) remaining edges after deleting all edges with too many neighbors. (l) result of edge straightening by applying the operations shown in (b) and (c). (m) result after deleting edge pixels that connect orthogonal by means of the morphologic patterns shown in (d), (e), (f), and (g).*

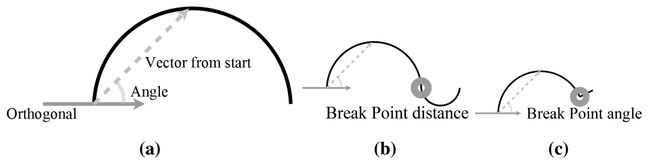

Figure 3: *In (a), the gray arrow is the calculated orthogonality, whereas the dashed gray arrow is the vector between the starting and the current point. The black line represents the processed edge. As the gray dashed arrow moves along the edge, the angle to the orthogonal decreases, whereas the length of the vector increases. (b) distance breaking and (c) angle breaking condition is triggered.*

2.1.2 Algorithmic approach

The algorithmic approach to filtering the edge image is based on the idea of breaking up lines at positions where the line course cannot belong to a common ellipse. Prerequisites here are edge-thinning, breaking up lines with too many neighbors, and line straightening as described previously. The algorithm starts with calculating the vector orthogonal to the first two points of a line (solid arrow in Figure 3a). For each following point, the vector from the starting point is calculated (dashed arrow in Figure 3a). Afterwards, the angle and distance between the orthogonal and the calculated vector is computed. For an ellipse, this angle has to shrink from 90° to 0°. Once the angle has reached 0°, it has to grow back to 90° whereas the distance has to shrink. If this is not the case in the beginning, the orthogonal vector has to be turned over. In case the shrinking and growing do not apply to the behavior of the line, a point where the edge has to be split is found (Figure 3b and 3c). This is shown in more detail in the provided pseudocode in Algorithm 1.

```
Data: LINE
DIR=0;SET(START);SET(ORTHO);SET(ANGLE);SET(DIST);
for IDX to Size(LINE) do
    VEC=CALC(START,LINE(IDX));
    if DIR=0 then
        if ANGLE>=ANGLE(ORTHO,VEC) AND DIST<=|VEC| then
            ANGLE=ANGLE(ORTHO,VEC);
            DIST=|VEC|;
        else
            BREAK_LINE();
        end
        if ANGLE=DIRECTION then
            DIR=90;
        end
    else
        if ANGLE<=ANGLE(ORTHO,VEC) AND DIST>=|VEC| then
            ANGLE=ANGLE(ORTHO,VEC);
            DIST=|VEC|;
        else
            BREAK_LINE();
        end
    end
end
```

Algorithm 1: Separation of lines collected from the edge image.

2.2 Select best ellipse

In this step, all consecutive edge pixels in the edge image are collected. For the morphologic approach this is done by combining all connected edge pixels into a line. In the algorithmic approach, closed lines can be excluded to decrease runtime. Therefore, open lines (start and end pixel have only one neighbor) and closed lines have to be separated. Open lines are collected by starting new lines only on pixels with one neighbor and closed lines are collected by starting at any pixel not accessed in the first step. These lines are evaluated based on their shape, the resulting shape after an ellipse fit, and the image intensity enclosed by the ellipse.

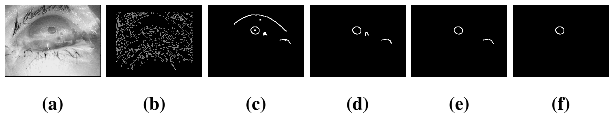
(a) (b) (c) (d) (e) (f)

Figure 4: *(a) input image and (b) edge-filtered result. For each line, we analyze whether it is curved based on the centroid of its pixels. The result is shown in (c) with the mean positions as bright dots. Then the algorithm fits an ellipse to the line. In case of success, the ellipse is further analyzed. Remaining lines after this fitting step are shown in (d). The first evaluation of the ellipse filters stretched ellipses by comparing the ratio of the two ellipse radii. The result is shown in (e). For the pupil area restriction a maximum and minimum percentage of pixels in the image is used as parameters. Picture (f) shows the remaining contour after this step.*

2.2.1 Remove straight lines

Since pupil contours exhibit a round or elliptical shape, straight lines have to be removed. We analyze whether each line is straight or curved based on the mean position of all pixels that belong to the line. If the shortest distance of a line pixel to the mean position is below an empirically set threshold $min\_mean\_line\_dist$, the line is straight. Note that this decision is taken for both x and y dimensions. An example of this step is shown in Figure 4c, where the mean position is represented by a white dot.

2.2.2 Ellipse fitting

There are several ways to fit an ellipse to a set of coordinates. In case of an online scenario (such as driving), where the information about the pupil position is used as input to other systems (e.g., driver assistance), we are interested in very low latencies. Therefore, we employ the least squares ellipse fit as in [Fitzgibbon et al. 1999] for efficient ellipse fitting. An exemplary result is shown in Figure 4d.

2.2.3 Ellipse evaluation

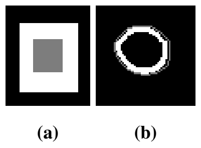
(a) (b)

Figure 5: *Calculation of the difference between the inner and outer area of an ellipse.*

In this step, we exclude ellipses that are unlikely to describe the pupil by considering their area, shape, and gray value ratio between the area inside and outside of the ellipse. The first restriction pertains the shape of the pupil by restricting the ratio between the two ellipse radii. The rationale is that the pupil position relative to the eye tracker camera can only distort the pupil ellipse eccentricity to a certain point. In our implementation, we chose $radi_{ratio} = 3$ empirically. The second restriction regards the pupil area in relation to the image size, since the eye tracker camera has to be positioned at a restricted distance from the eye (neither to close nor to far), which is reflected on the ratio of the image area occupied by the pupil. We used two thresholds, namely $min_{area} = 0.5\%$ and $max_{area} = 10\%$ of the total image area. Due to the eye physiology, the last evaluation step expects the pupil to be darker than its surroundings. Figure 5a shows the calculated pattern based on the radius of the ellipse. To reduce computation time, we consider only a portion of the minimum enclosing, unrotated rectangle, instead of the whole ellipse, as shown in Figure 5a. Pixels within the gray box in Figure 5a contribute to the pupil intensity value and those within the black box contribute to the surrounding intensity. The size of the gray box is $\frac{1}{2}$ of the width and height of the enclosing rectangle. The white box in Figure 5a has the size of the enclosing rectangle and the surrounding black box has $\frac{3}{2}$ of this size.

To evaluate the validity of an ellipse, the surface difference of the pupil box and the surrounding box is calculated (as shown in Figure 5a). This difference is compared against a threshold. In our implementation, we used a $validity_{threshold} = 10$, implying that we expect the surface difference to exceed 10 in order to be valid.

2.2.4 Rate ellipse

All found ellipses have to be compared against each other. For this, the inner gray value of each ellipse is computed by calculating a vector between each point of the line and the center of the ellipse. This vector is shortened by multiplying it stepwise from 0.95 to 0.80 with a step size of 0.01. Figure 5b shows the line pixels in gray and all pixels contributing to the inner gray value ($gray_{value}$ in Equation (1)) in white. Note that each pixel can only contribute once to the inner gray value. This value is normalized by the sum of all contributing pixels.

$$eval(el) = gray_{value} * (1 + |el_{width} - el_{height}|) \quad (1)$$

The best of all remaining ellipses is chosen by selecting the ellipse with the lowest inner gray value and the roundest shape. Equation (1) shows the formula for calculating the rank of an ellipse, where el is the ellipse, and el_{width}, el_{height} are the radii of the ellipse. If el_{width} and el_{height} are equal the ellipse is round. The variable $gray_{value}$ in Equation (1) is the calculated inner gray value as specified before. The ellipse with the lowest value calculated based on Equation (1) is chosen. If there is more than one ellipse with the lowest value, the one with the most edge points and therefore clearest contour is chosen.

2.3 Coarse positioning

A different approach is chosen if the algorithm cannot find a good pupil edge – e.g., due to motion blur, the pupil being located in a dark spot, or the pupil being hidden by eyelashes. More specifically, we apply an additional method that tries to find the pupil by first determining a likely location candidate and then refining this position. Since a computationally demanding convolution operation is required, we rescale the image to keep run-time tractable (see Section 2.3.1). This rescaling process contains a low pass procedure to preserve dark regions and to reduce the effect of blurring or eyelashes. Afterwards, the image is convolved with two different filters separately: 1) a surface difference filter to calculate the area difference between an inner circle and a surrounding box, and 2) a mean filter. The results of both convolutions are multiplied, and the maximum value is set as the starting point of the refinement step.

2.3.1 Rescale image with low pass

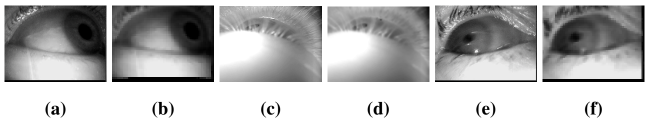

Figure 6: *(a), (c) and (e) show input images taken from the data set proposed by [Fuhl et al. 2015] and [Świrski et al. 2012]. The respective results of the downscaling operation by a factor of six using the mean between zero and the mean of the input image region influencing a pixel are shown in (b), (d) and (f).*

There are several methods to downscale an image, e.g., based on nearest neighbor, bilinear or bicubic interpolations, based on Lanczos kernel or more advanced downscaling operations like content adaptive [Kopf et al. 2013] or clustering based [Gerstner et al. 2012] downscaling. In case the edge detection part of the algorithm could not find a good edge because of motion blur (Figure 6e) or eyelashes (Figure 6c), a downscaling operation that weights dark pixels stronger would be preferable. However, considering that the pupil could also be in a dark region of the image (as in Figure 6a), weighting dark pixels too strong can lead to a merging of the pupil and the surrounding dark region. We apply a fast method to calculate the intensity histogram and the mean (Equation (2)) of the pixels influencing the new pixel. Afterwards, the mean of the lower part of the histogram (defined as the part smaller than the mean of the whole histogram) is computed (Equation (3)). The resulting value is used as the intensity of the new pixel. This method weights dark pixels stronger based on the intensity distribution of the influencing area (specified by x_1, y_1, x_2 and y_2). $I(x_i, y_i)$ denotes the intensity value of a pixel.

$$Mean(x_1, y_1, x_2, y_2) = \frac{\sum_{x_i=x_1}^{x_2} \sum_{y_i=y_1}^{y_2} I(x_i, y_i)}{|x_1 - x_2| * |y_1 - y_2|} \quad (2)$$

$$MUM(x_1, y_1, x_2, y_2) = \frac{\sum_{x_i=0}^{Mean(x_1,y_1,x_2,y_2)} IH(x_i) * x_i}{\sum_{x_i=0}^{Mean(x_1,y_1,x_2,y_2)} IH(x_i)} \quad (3)$$

Equation 3 yields the mean neighborhood intensity of the dark neighborhood regions, where darkness is defined by the mean calculated in Equation (2). Therefore, it uses the intensity histogram of the region which is denoted as $IH(x_i)$ and the intensity index denoted as x_i. For our implementation we used overlapping regions with a window $radius_{scale} = 5$ (Figure 7a). The overlapping regions do not include the center of the other boxes, and therefore, $radius_{scale} = 5$ downscales an image by a factor of six (Figure 7b).

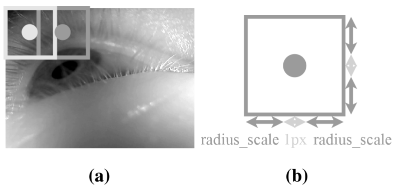

Figure 7: *(a) shows how neighborhood regions of pixels close to each other in the downscaled image can overlap (light gray box and dark gray box). Each gray box represents the pixels influencing the intensity of a pixel in the downscaled image. The circles represent the center of a region. In (b) the construction of the window based on the parameter $radius_{scale}$ is shown.*

2.3.2 Convolution filters

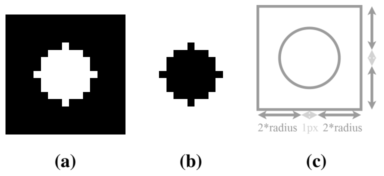

Figure 8: *(a) mean filter, where the white region's sum is one and the black region's is zero. (b) surface difference filter, where the black inner circle sums up to minus one and the surrounding white to one. Both kernels have the same size. (c) filter construction, where the radius is calculated based on the image resolution.*

The convolution filters used are a mean (Figure 8a) and a surface difference filter (Figure 8b). Because of the unknown shape and expected roundness of the pupil both filters contain the shape of a circle. Our algorithm expects the input image to contain the complete eye, and therefore, the expected pupil size depends on image resolution. To calculate the parameter $radius_{filter}$ we simply divide the resolution in the x and y dimension of the image by 100. Afterwards, the maximum of these two values is rounded up and used as the parameter $radius_{filter}$. The construction of the filters based on this value ($radius_{filter} = radius$) is shown in Figure 8c.

The diameter of such a circle in the real image is

$$(radius_{scale} + 1) * (radius_{filter} * 2 + 1), \quad (4)$$

which is expected to be larger than the real pupil. This is important for the surface difference filter (Figure 8b) because, on larger pupils, the result in the middle would be lower than the result closer to the border of the pupil.

2.3.3 Select best position

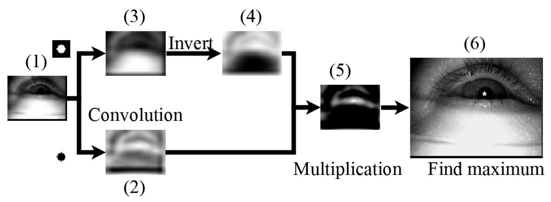

Figure 10: *Workflow after downscaling of the coarse positioning. The input (1) is the downscaled image. (2) result of the convolution with the surface difference filter (Figure 8a). (3) convolution result of the mean filter (Figure 8b) and (4) inverted image. The result (5) is the point wise multiplication of (2) and (4). The absolute maximum of (5) is represented by a white cross in the real image (6) taken from the data set proposed in [Fuhl et al. 2015].*

To find the best fitting position of the pupil, we first convolve the downscaled image with the surface difference filter (Figure 8b). All areas with low intensity values in the inner circle of the filter (black in Figure 8b) and high values in the surrounding area will have positive results (white in Figure 10(2)). The bigger this difference is, the higher the convolution response. The idea behind this is that the pupil is surrounded by brighter intensity values. Problems with this filter are that other areas respond also with positive values and the filter response does not include intensity information of the inner area (black in Figure 8b). We are searching for the pupil, which is expected to be dark, and, therefore, we use the mean filter (Figure 8a) to include the intensity response of the inner area (Figure 10(3)). To achieve this, the result of the convolution with the mean filter has to be inverted (Figure 10(4)). This is because the response of areas with low intensity is low, and, to use it as weight for the result of the surface difference filter, we want it to be high.

The weighting is done by pointwise multiplication of the two convolution responses of the inverted mean (Figure 10(4)) and the surface difference filter (Figure 10(2)). In the result of the weighting (Figure 10(5)) the maximum is searched and used as coarse position (white cross in Figure 10(6)).

By inverting the surface difference and not inverting the mean filter, this algorithm searches for a white blob. Additionally, by reducing the filter size and operating only on a small area surrounding the pupil center position, it can be used for cornea reflection detection.

2.4 Optimize position

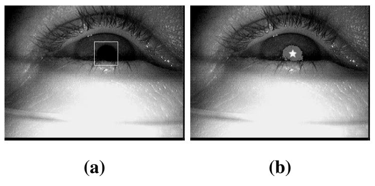

Figure 11: *In (a), the area, in which the optimization takes place is enclosed by a white box. (b) shows the pixels below the calculated threshold (dark gray area) and the resulting position (white cross).*

The coarse position is based on the downscaled image, and, therefore, one pixel error relative to the pupil center represents a distance of six pixels in the original image. For the optimization step, we expect the coarse position to be contained within the pupil and calculate a pupil intensity threshold using the neighborhood of the coarse position in the real image. In our implementation we choose $size_{neigbourhood} = 2$, meaning a pixel distance of two in each direction. The mean of this box is calculated, and the absolute difference to the pixel value of the coarse position is computed. This difference is added to the coarse position pixel value and used as a threshold. To optimize the position, a small window (Figure 11a, white box) surrounding the coarse position in the real image is thresholded. The dark gray area in Figure 11b shows this thresholded region. We chose the window size $radius_{filter} * radius_{filter}$ in each direction empirically. Afterwards, the center of mass of the thresholded pixels is calculated and used as pupil center position (white cross in Figure 11b).

2.5 Validate position

The second method will always find a pupil location, even if the eye is currently closed. Therefore, we have to validate the candidate location. This is done in the same way as for the ellipse validation shown in Figure 5a. For the two diameters of the ellipse we used $radius_{filter} * radius_{filter} * 2 + 1$. The parameter $validity_{threshold}$ is set to the value previously defined in Section 2.2.3 (i.e., 10).

3 Experimental Evaluation

ElSe was evaluated on over 94,000 hand-labeled eye images collected during eye-tracking experiments in real-world scenarios. These data sets and the performance of our algorithm will be presented in the following.

3.1 Data Sets

For our evaluation, we employed several data sets provided in related work as well as a new hand-labeled data set. More specifically, ElSe was evaluated on the data set presented by [Świrski et al. 2012], 17 data sets introduced by [Fuhl et al. 2015] (i.e., data sets I-XVII in Table 1), and 7 new hand-labeled data sets (i.e., data sets XVIII-XXIII in Table 1). Figure 9 shows an overview of these new data sets, where each column contains exemplary images of one data set. Among these, Data sets XVIII-XXII were derived from eye-tracking recordings during driving [Kasneci et al. 2014a]. Data sets XXIII and XXIV were recorded during in-door experiments with Asian subjects, where the challenge in pupil detection arises from eyelids and eyelashes covering or casting shadows onto the pupil (and, in one case, glasses reflections). Further challenges associated with Data set XXIV (Figure 9 last column) are related to reflections on eyeglasses. These reflections have low transparency and, therefore, affect the intensity value of the pupil in the image as well as the edge image.

The challenges in the eye images included in the Data sets XVIII, XIX, XX, XXI and XXII are related to motion blur, reflections, and low pupil contrast to the surrounding area. These challenges occur simultaneously in some images. For Data set XVIII (Figure 9 first column) most of the reflections have low transparency and form few areas. In contrast to that, reflections in Data set XIX (Figure 9 second column) also have low transparency but appear scattered in many areas. This leads to edge images where the pupil edges are very difficult to extract. For Data set XX, the reflections are also scattered in many areas but with more transparency. Data set XXI consists mainly of images where the main challenge is a dark region surrounding the pupil. This is shown in the fourth column

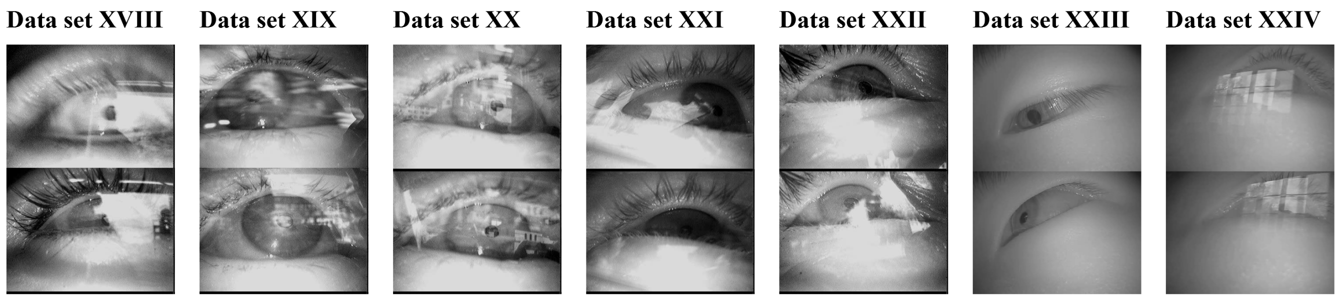

Figure 9: *New hand-labeled data sets with exemplary images showing the main challenges regarding the automated image processing. The first five data sets were collected in an on road driving experiment, the last two are collected during in-door experiments with Asian subjects.*

of Figure 9. All seven data sets were recorded with different subjects and contain overall 55,712 images (resolution 384x288 pixels). These data set can be downloaded at ftp://emmapupildata@messor.informatik.uni-tuebingen.de (password:*eyedata*).

3.2 Results

Data set	SET(%)	Starburst(%)	Swirski(%)	ExCuSe(%)	ElSe(%)
Swirski	63.00	19.33	77.17	***86.17***	82.00 | *81.83* | *20.83*
I	10.27	5.48	5.11	70.95	***85.52*** | *85.26* | *13.25*
II	43.76	4.16	26.34	34.26	***65.35*** | *20.39* | *68.71*
III	12.23	1.71	6.81	39.44	***63.60*** | *63.25* | *9.39*
IV	4.03	4.44	34.54	81.58	***83.24*** | *83.05* | *2.52*
V	18.08	14.66	77.85	77.28	***84.87*** | *81.59* | *37.89*
VI	10.30	19.14	19.34	53.18	***77.52*** | *77.34* | *4.40*
VII	2.19	2.41	39.35	46.91	***59.51*** | *59.01* | *7.23*
VIII	36.67	9.52	41.90	56.83	***68.41*** | *52.06* | *63.17*
IX	10.20	13.88	24.09	74.60	***86.72*** | *86.40* | *10.77*
X	57.62	51.07	29.88	***79.76***	78.93 | *77.38* | *32.02*
XI	23.51	27.79	20.31	56.49	***75.27*** | *74.80* | *1.37*
XII	56.11	64.50	71.37	79.20	***79.39*** | *79.19* | *11.64*
XIII	33.40	46.64	61.51	70.26	***73.52*** | *72.70* | *15.47*
XIV	46.27	22.81	53.30	57.57	***84.22*** | *82.51* | *7.67*
XV	38.29	7.71	***60.88***	52.34	57.30 | *54.82* | *39.39*
XVI	57.14	8.93	17.86	49.49	***59.95*** | *55.61* | *14.79*
XVII	***91.04***	0.75	70.90	77.99	89.55 | *86.56* | *52.61*
XVIII	1.32	1.92	12.39	22.24	***50.86*** | *50.11* | *4.66*
XIX	4.75	5.25	9.03	26.45	***33.04*** | *32.52* | *5.29*
XX	3.20	3.73	17.93	52.37	***67.90*** | *67.54* | *7.58*
XXI	2.29	2.41	8.09	***43.54***	41.47 | *38.69* | *7.89*
XXII	1.91	5.91	1.98	27.93	***48.98*** | *47.42* | *8.58*
XXIII	55.43	8.03	***96.54***	93.86	94.34 | *92.45* | *54.55*
XXIV	0.94	1.87	44.43	45.21	***52.97*** | *49.73* | *28.82*

Table 1: *Performance comparison of SET (best result of both parameter settings), Starburst, Swirski, ExCuSe and ElSe in terms of detection rate up to an error of five pixels. The best performance on each data set is shown in bold. For ElSe the table provides results of the complete algorithm | best edge selection(2.2) | coarse positioning and refinement (2.3, 2.4 and 2.5).*

We compared our algorithm to four state-of-the-art approaches on the above data without adjusting its parameters, i.e., ElSe was applied with one fixed parameter setting to all data sets. The competitor algorithms are SET [Javadi et al. 2015], Starburst [Li et al. 2005], [Świrski et al. 2012], and ExCuSe [Fuhl et al. 2015]. For SET, we applied two parameter combinations *ThresholdLuminance* = 30, *ThresholdRegion* = 600 and *ThresholdLuminance* = 80, *ThresholdRegion* = 800 with the MATLAB version from the eyego eyetracker website (https://sites.google.com/site/eyegoeyetracker/, accessed on June, 1 2015). Starburst [Li et al. 2005] was used in its MATLAB Version 1.1.0 as provided by the OpenEyes website (http://thirtysixthspan.com/openEyes/software.html, accessed on June 1, 2015) without any changes in the parameter setting. The starting location of the algorithm was set to the center of the image. The algorithm proposed by Swirski et al. [Świrski et al. 2012] was used with the parameter settings provided by the authors (https://www.cl.cam.ac.uk/research/rainbow/projects/pupiltracking/, accessed on June 1, 2015). ExCuSe [Fuhl et al. 2015] was used with the parameter setting provided by the authors (https://www.ti.uni-tuebingen.de/Pupil-detection.1827.0.html?&L=1, accessed on June 12, 2015). The performance was measured in terms of the detection rate for different pixel errors. The pixel error represents the Euclidean distance between the hand-labeled center of the pupil and the pupil center reported by the algorithm. Note that we do not report performance measures related to the gaze position on the scene, since this also depends on the calibration. We focus on the pupil center position on the eye images, where the first source of noise occurs. Table 1 summarizes the performance results for each of the competing algorithms in terms of the detection rate up to an error of five pixels. For each data set, the best result is shown in bold. In addition, Figure 12 presents the performance of ElSe and its competitors in terms of the detection rate for different pixel error rates (0-15 pixels). More specifically, Figure 12(a) shows the detection rate as the percentage of correctly detected pupil centers for the 94,713 images, whereas Figure 12(b) depicts the detection rate as the average over all data sets with each data set weighted equally (due to different data set sizes). In both evaluations, ElSe clearly outperformed all competitor algorithms. For Data sets X, XV, XVII, XXI, XXIII, and [Świrski et al. 2012], ElSe has not the best detection rate but is always close to the best result (on average 2% worse than the best performer in this cases).

For the [Świrski et al. 2012] approach, we measured a runtime of 8 ms per image on an i5-4570 (3.2GHz) CPU. ExCuSe [Fuhl et al. 2015] needed 6 ms per image, whereas ElSe (without parallelization) 7 ms. The algorithms SET [Javadi et al. 2015] and Starburst [Li et al. 2005] are not comparable in terms of runtime because we used their MATLAB version.

3.3 Parameter sensitivity

Decisions in ElSe are taken based on several parameters. To quantify the impact of these parameters on ElSe's performance we conducted four tests. The results are shown in Figure 13. The first test regards the parameter $min\_mean\_line\_dist$, which is used to decided whether a line is straight or curved. If the value of this parameter is too high, the probability to remove correct lines increases. A too low value does not have a significant effect on the result but leads to an increase in runtime (1-2 ms per image). In the second test, the ellipse restrictions for radius relation and area were removed. This reduces the detection rate by about 1%, and therefore, these parameters are unlikely to have a big impact on detection results. For the third test, the validation threshold $validity_{threshold}$

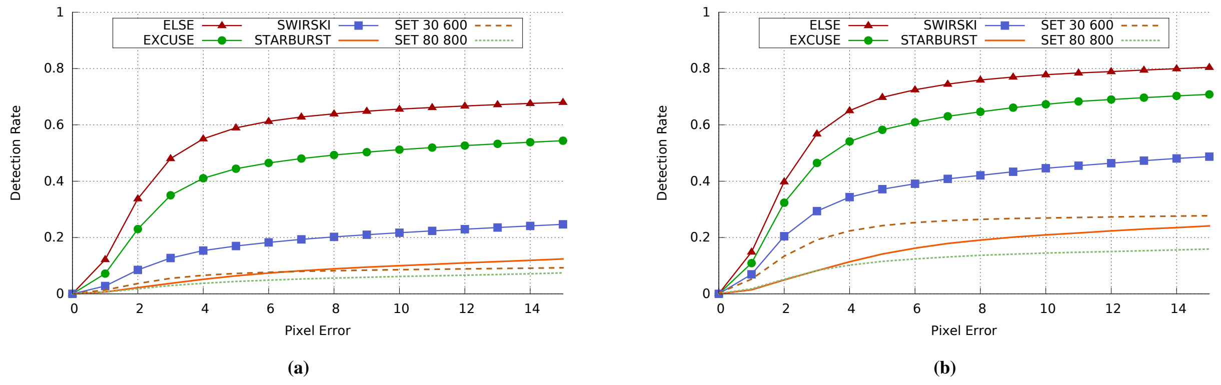

Figure 12: *Average detection rates at different pixel distances on all data sets. In (a) the result for each data set is weighted by the number of images in the corresponding data set. (b) presents the mean (unweighted) detection rate over all data sets. SET [Javadi et al. 2015] is shown with two different settings: 1) luminance=30, area threshold=600 and 2) luminance=80, area threshold=800.*

in the algorithmic steps *Select best ellipse 2.2* and *Validate position 2.5* in Figure 1 was changed. For lower values of this parameter more ellipses get accepted, leading thus to an increase in detection rate but also to a higher false positive rate. This means that more blinks are falsely accepted as pupils. Higher values decrease the detection rate but reduce the false positive rate too. The last test regards the coarse positioning by changing the parameter $radius_{scale}$, which effects the rescaling. Too high and too low values for this parameter reduce the detection rate of ElSe by about 1-2%. The main effect of this parameter is on the run-time because of the convolution step. For high values, the run-time decreases, whereas the run-time increases for low values. Nonetheless, a comparison of the results in Figure 13 with those shown in Figure 12 (a) reveals that even for these parameter changes ElSe has a higher detection rate than its competitors.

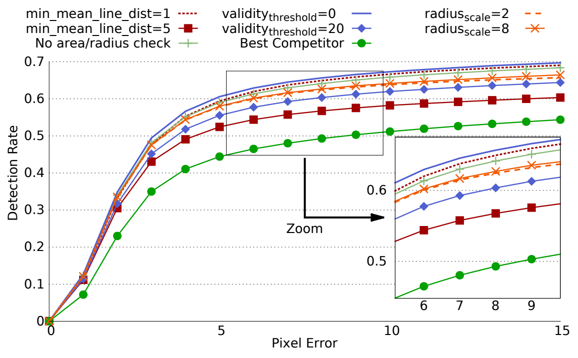

Figure 13: *The detection rates of ElSe with different parameter settings for all data sets (94,713 images). The impact of the $min\_mean\_line\_dist$ parameter on the detection rate is shown in red, the impact of the $validity_{threshold}$ in blue, the impact of the rescaling factor $radius_{scale}$ in orange, and the performance when removing the ellipse area and radius relation check in pale grey. For reference, the best competitor (ExCuSe) is also displayed.*

In addition, Figure 14 shows the results of ElSe in comparison to different line combining approaches. The first and second approach, i.e., ELSE LINECOMB 75 and ELSE LINECOMB 25, use the mean line positions. If the distance between both is less than 75 pixels (25 for the second approach), the lines are combined. The runtime for these approaches on one i5-4570 (3.2GHz) CPU core is ∼ 300ms and ∼ 60ms, respectively. The overall performance is however poorer than that of ElSe, since many ellipses are created,

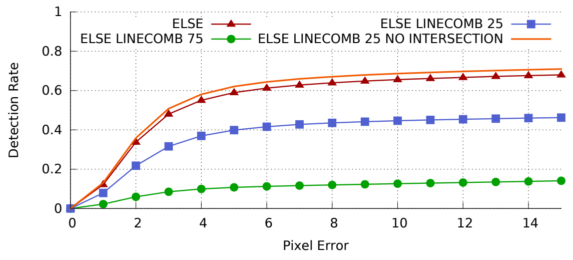

Figure 14: *The detection rates of ElSe for different line combining approaches.*

which may be rounder than the valid ellipse (Formula 1) or valid ellipses out of the edge image bypassing the second part (Section 2.3) of the algorithm are created. The last approach ELSE LINECOMB 25 NO INTERSECTION uses the mean distance threshold of 25 pixels. Additionally, both lines are not allowed to lay between the two mean positions (runtime ∼ 40ms).

3.4 Limitations of ElSe

ElSe relies on the Canny edge filter. Thus, if the edge selection fails, the convolution approach makes the critical assumption that the pupil has a low intensity value surrounded by higher intensity values. For input images where the pupil is partially covered by many small reflections (e.g., Figure 15 (a)) the algorithm fails. Such reflections not only destroy the edge filter response (Figure 15 (e)) but also lower the convolution filter result (Figure 15 (i)), leading thus to a wrong coarse positioning. Another example where ElSe fails, is shown in Figure 15(b). Here the pupil is surrounded by a very dark iris and the skin below the eye is very bright. The dark iris leads to a good edge filter response (Figure 15(f)). The edge selection, however, fails at the validation check for the correct ellipse. Afterwards, the convolution approach (Figure 15(j)) is applied, but due to the bright skin below the eye, the coarse positioning fails.

A last failure example are reflections covering most of the pupil (Figure 15(c)). In such cases the pupil edge is scattered due to the high magnitude response of the edges belonging to the reflection (Figure 15 (g)). Figure 15 (k) shows the multiplied convolution response for the coarse positioning. Here the dark pupil part (Figure 15 (c)) has only a low response. Since most of the pupil is bright, it will lead to a negative surface difference and, therefore, to a low weight through the mean filter.

The images Figure 15 (d), (h) and (l) show cases where ElSe (white bar) is outperformed by the algorithms ExCuSe [Fuhl et al. 2015] (green dot) and [Świrski et al. 2012] (red star). For the eye image in (d), ElSe fails because it does not find a valid edge and the convolution response at the eye corner is highest. In (h), the selected edge is misshaped, resulting thus in an invalid ellipsis. In this case, the RANSAC ellipse fitting as employed by [Świrski et al. 2012] performs better. In the last example (l), ElSe fails due to an invalid pupil edge, which is caused by eyelashes. This edge connection is not broken, leading thus to an invalid invalid. In consequence, ElSe selects a wrong line close to the eyelid, which leads to a wrong result. Despite these limitations, ElSe showed high robustness in comparison to related approaches.

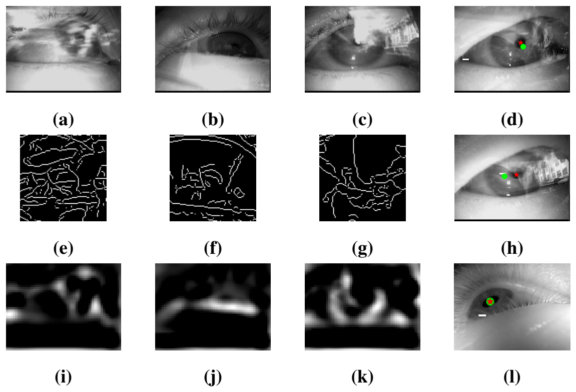

Figure 15: *Failure cases of ElSe on the input images (a), (b) and (c). (e), (f) and (g) show the corresponding morphologic filtered edge images surrounding the pupil. (i), (j) and (k) show the convolution response corresponding to the input image below at the top row. The images (d), (h) and (l) are examples were the algorithms ExCuSe [Fuhl et al. 2015](blue dot) and Swirski [Świrski et al. 2012](red star) perform better than ElSe(white bar).*

4 Conclusions

We presented a novel pupil detection algorithm, ElSe, for real-time eye-tracking experiments in outdoor environments. ElSe was evaluated on 94,713 challenging, hand-labeled eye images in which reflections, changing illumination conditions, off-axial camera position and other sources of noise occur. We compared our approach against four state-of-the-art methods and showed that ElSe outperformed the competitors by far. The implementation and data sets are available for download. ElSe is currently used online in two projects with a mobile eye tracker recording at 60 Hz. For higher sampling rates, each step can be parallelized. For avoidance of parameter tuning the implementation of ElSe rescales the input image width to 384 pixels keeping the aspect ratio.

References

FITZGIBBON, A., PILU, M., AND FISHER, R. B. 1999. Direct least square fitting of ellipses. *IEEE TPAMI 21*.

FUHL, W., KÜBLER, T., SIPPEL, K., ROSENSTIEL, W., AND KASNECI, E. 2015. Excuse: Robust pupil detection in real-world scenarios. In *CAIP*, Springer, 39–51.

GERSTNER, T., DECARLO, D., ALEXA, M., FINKELSTEIN, A., GINGOLD, Y., AND NEALEN, A. 2012. Pixelated image abstraction. In *Proceedings of the Symposium on NPAR*, 29–36.

GONI, S., ECHETO, J., VILLANUEVA, A., AND CABEZA, R. 2004. Robust algorithm for pupil-glint vector detection in a video-oculography eyetracking system. In *ICPR*, 941–944.

JAVADI, A.-H., HAKIMI, Z., BARATI, M., WALSH, V., AND TCHEANG, L. 2015. Set: a pupil detection method using sinusoidal approximation. *Frontiers in neuroengineering 8*.

KASNECI, E., SIPPEL, K., AEHLING, K., HEISTER, M., ROSENSTIEL, W., SCHIEFER, U., AND PAPAGEORGIOU, E. 2014. Driving with Binocular Visual Field Loss? A Study on a Supervised On-road Parcours with Simultaneous Eye and Head Tracking. *Plos One 9*, 2, e87470.

KASNECI, E., SIPPEL, K., HEISTER, M., AEHLING, K., ROSENSTIEL, W., SCHIEFER, U., AND PAPAGEORGIOU, E. 2014. Homonymous visual field loss and its impact on visual exploration: A supermarket study. *TVST 3*, 6.

KASNECI, E. 2013. *Towards the Automated Recognition of Assistance Need for Drivers with Impaired Visual Field*. PhD thesis, University of Tübingen.

KEIL, A., ALBUQUERQUE, G., BERGER, K., AND MAGNOR, M. A. 2010. Real-time gaze tracking with a consumer-grade video camera.

KOPF, J., SHAMIR, A., AND PEERS, P. 2013. Content-adaptive image downscaling. *ACM TOG 32*.

LI, D., WINFIELD, D., AND PARKHURST, D. J. 2005. Starburst: A hybrid algorithm for video-based eye tracking combining feature-based and model-based approaches. In *CVPR Workshops 2005*, 79–79.

LIN, L., PAN, L., WEI, L., AND YU, L. 2010. A robust and accurate detection of pupil images. In *BMEI 2010*, vol. 1, IEEE.

LIU, X., XU, F., AND FUJIMURA, K. 2002. Real-time eye detection and tracking for driver observation under various light conditions. In *IEEE Intelligent Vehicle Symposium*, vol. 2.

LONG, X., TONGUZ, O. K., AND KIDERMAN, A. 2007. A high speed eye tracking system with robust pupil center estimation algorithm. In *EMBS 2007*, IEEE.

PERÉZ, A., CORDOBA, M., GARCIA, A., MÉNDEZ, R., MUNOZ, M., PEDRAZA, J. L., AND SANCHEZ, F. 2003. A precise eye-gaze detection and tracking system.

SCHNIPKE, S. K., AND TODD, M. W. 2000. Trials and tribulations of using an eye-tracking system. In *CHI'00 ext. abstr.*, ACM.

SIPPEL, K., KASNECI, E., AEHLING, K., HEISTER, M., ROSENSTIEL, W., SCHIEFER, U., AND PAPAGEORGIOU, E. 2014. Binocular Glaucomatous Visual Field Loss and Its Impact on Visual Exploration - A Supermarket Study. *PLoS ONE 9*, 8, e106089.

ŚWIRSKI, L., BULLING, A., AND DODGSON, N. 2012. Robust real-time pupil tracking in highly off-axis images. In *Proceedings of the Symposium on ETRA*, ACM, 173–176.

TRÖSTERER, S., MESCHTSCHERJAKOV, A., WILFINGER, D., AND TSCHELIGI, M. 2014. Eye tracking in the car: Challenges in a dual-task scenario on a test track. In *Proceedings of the 6th AutomotiveUI*, ACM.

VALENTI, R., AND GEVERS, T. 2012. Accurate eye center location through invariant isocentric patterns. *TPAMI 34*.

ZHU, D., MOORE, S. T., AND RAPHAN, T. 1999. Robust pupil center detection using a curvature algorithm. *CMPB 59*.