Machine Vision and Applications
https://doi.org/10.1007/s00138-019-01022-y

ORIGINAL PAPER

xSDL: stroboscopic differential lighting eye tracker with extended temporal support

Frank H. Borsato[1] · Antonio Diaz-Tula[2] · Carlos H. Morimoto[2]

Received: 3 June 2018 / Revised: 27 November 2018 / Accepted: 8 March 2019

Abstract
Eye tracking (ET) for gaze interaction in wearable computing imposes harder constraints on computational efficiency and illumination conditions than remote ET. In this paper we present xSDL, an extended temporal support computer vision algorithm for accurate, robust, and efficient pupil detection and gaze estimation. The robustness and efficiency of xSDL partly come from the use of stroboscopic differential lighting (SDL), an extension of the differential lighting pupil detection technique developed in the 90's. Due to the erratic behavior of eye movements, traditional computer vision tracking techniques (such as Kalman filters) do not perform well, so most ET techniques simply detect some eye feature (such as the pupil center) at every frame. Extended temporal support uses keyframes selected during eye fixations and a simple translation model of the pupil to further improve the computational performance of SDL. A prototype composed of two independent acquisition systems was developed to evaluate the performance of xSDL and other four state-of-the-art ET techniques under similar conditions. Our results show that xSDL outperforms those four algorithms, both in speed (close to 2000 Hz using 240 line frames) and accuracy.

Keywords Eye tracking · Stroboscopic differential lighting · Extended temporal support · xSDL

1 Introduction

Eye trackers are devices that can estimate the point of gaze on a computer screen [28] or in the scene in front of the user [20]. Such devices are commonly used in usability studies [17], marketing research [23,30], medical diagnosis [21,33], communication for people with disabilities [4], psychological and psychophysical studies [5], and virtual reality [32,35].

Current eye tracking systems are mostly feature-based [14], i.e., they use one or more video cameras to detect and track the eye features and estimate the point of gaze on a planar surface (typically the computer screen). Most methods detect and track the iris or the pupil center and use active near infrared (NIR) illumination to improve the tracking performance. The use of NIR light is also desirable because it creates a corneal reflection that can be used as a reference point for gaze estimation [28].

The point of gaze can be estimated by a function that maps eye features (such as the pupil center) onto the observed surface. For example, a second-order polynomial can be used where the coefficients can be computed using regression techniques with corresponding eye features and gaze positions obtained from a calibration procedure. Detecting eye features, such as the pupil or iris, in a video frame can be challenging due to noise, low image resolution, and motion blur.

When the eye is illuminated using a NIR light source placed away from the camera optical axis (off-axis), the pupil appears dark in the camera image. This facilitates the segmentation and contour detection of pupils within light-color irises, but the contrast between the pupil and iris is low for people with darker eyes. Segmentation and tracking of the iris or iris contour (also known as limbus tracking) is also possible but it is more likely to be affected by occlusions of the eyelids and lashes.

Differential lighting (DL) [8,26] was introduced to improve the robustness of pupil detection methods. DL relies on two NIR illuminators. One is placed very close to the camera optical axis (on-axis), and one off-axis. The on-axis

✉ Frank H. Borsato
frankhelbert@utfpr.edu.br

[1] Universidade Tecnológica Federal do Paraná, Via Rosalina Maria dos Santos 1233, Campo Mourão 87301-899, Brazil

[2] Universidade de São Paulo, Rua do Matão 1010, São Paulo 05508-090, Brazil

Published online: 15 April 2019

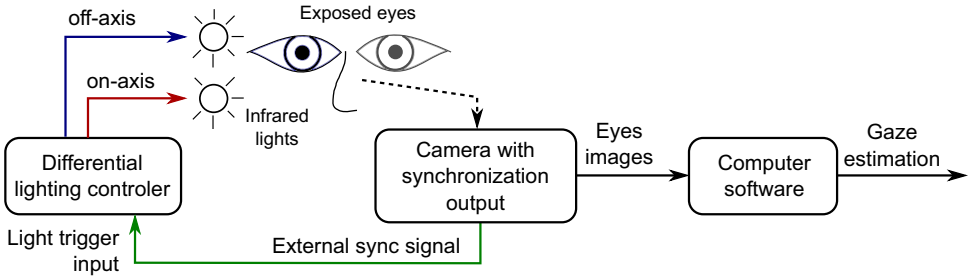

Fig. 1 Block diagram depicting the main components of the DL technique. The camera has a synchronization output which is used to trigger the light sources

illuminator generates bright pupil images because the camera is able to capture the light reflected from the back of the eye. DL alternates the off and on-axis illumination, producing a sequence of dark and bright pupil images. By subtracting two consecutive frames (one dark and one bright), the overlap between the dark and bright pupils can be easily segmented as regions of high contrast.

Despite its advantages, DL requires camera synchronization with the NIR light sources to produce the sequence of dark and bright pupil images. Figure 1 depicts such arrangement. Unfortunately most low-cost consumer cameras today do not provide a synchronization output. This is particularly true for digital web cameras used with computers. This might be one of the reasons why current low-cost eye trackers built with off-the-shelf web cameras use a single off-axis NIR illumination to detect and track the pupil [10,11,22,37].

In this paper, we present a high-performance, low-cost, stroboscopic differential lighting eye tracking technique with extended temporal support (xSDL). Our technique can be used with virtually any digital camera and was particularly designed to be used with cameras without external synchronization. The dual NIR illuminators are synchronized by software.

The remaining of this paper is organized as follows. Section 2 presents the basic differential lighting technique developed for analog cameras. Section 3 describes how the use of stroboscopic lighting allows DL to be used with digital cameras without external synchronization output. In Sect. 4, we introduce the extended temporal support algorithm to improve the overall performance of SDL. Section 5 presents the evaluation of xSDL and its comparison with state-of-the-art algorithms in a typical gaze estimation experiment. Section 6 presents the results of the xSDL evaluation. In Sect. 7, we present a discussion about xSDL, and finally Sect. 8 concludes the paper.

2 Pupil detection using differential lighting

The differential lighting (DL) technique introduced by Ebisawa and Satoh [8] was developed as a robust pupil detection method to improve the performance of nonverbal communication tools for people with disabilities. In [7], Ebisawa shows several refinements to the basic DL such as noise removal using morphological operations and pupil brightness control.

Morimoto et al. [25] describe a pupil-corneal-reflection (PCR) gaze estimation technique using DL and a second-order polynomial for mapping the PCR vector to target coordinates. Their system used an analog 30 Hz NTSC camera with external synchronization output. An external electronic circuitry was used to synchronize the even and odd frames of one interlaced camera image to the on and off-axis lights. Because each interlaced image contained a dark and bright pupil image, the pupil could be detected at 60 fields per second (where the field has half the resolution of an image frame after de-interlacing). In [25], the pupil was computed as the center of mass of the blob detected from the differential image. Hennessey et al. [16] proposed, as a further DL refinement, the computation of the actual pupil contour in the bright or dark pupil images, since the differential image only provides the overlap region when the eye is moving.

Morimoto and Flickner [24] also use DL to detect the eyes in a wider area to detect and track multiple faces. Ji and Yang [18] describe a gaze and face pose tracking system for monitoring driver's vigilance using DL. Due to the simplicity and good overall performance of the method, several other refinements and applications have been suggested in the literature [15,19,38].

DL was developed in the 90's to be used with analog cameras and *low performance* computers—low performance when compared to regular desktop computers today. As computers got more powerful and cameras more affordable and easier to setup and use (digital plug-and-play cameras), other pupil detection and tracking methods that do not require custom external hardware and use only visible light, such as [13], are preferred in practice despite DL's good performance. Nonetheless, as computational power continues to increase and hardware continues to reduce in size, the rise of technologies such as mobile, ubiquitous, and wearable computing demands more restrictive requirements for energy consumption and computational efficiency. Because an eye tracker can be used as a wearable input device that is always on, DL might become a stronger alternative because the active lighting allows the method to work under differ-

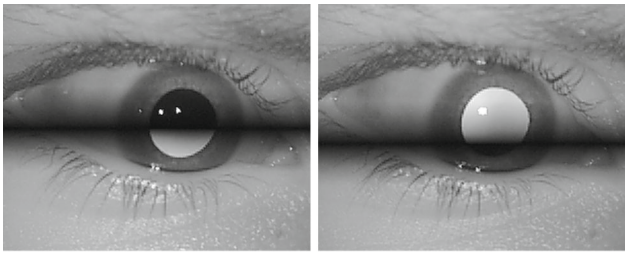

Fig. 2 Dark (**a**) and bright (**b**) pupil images with dark stripes created when the stroboscopic lights are not correctly synchronized with the camera frames. The stripes correspond to sensor lines that were not lit by the light pulses

ent lighting conditions, it is computationally very efficient, and its simplicity allows the method to be implemented in hardware [1].

Despite its advantages, DL using analog cameras requires the external lighting to be synchronized with the camera's odd and even fields [7,25]. A modern alternative would be the use of syncing-capable digital cameras with global-shutter, where all pixels are exposed within the same time window and a signal is generated to allow synchronization. While this option is attractive, such high-end cameras are still expensive. Most modern digital cameras (such as external webcams and cameras used in notebooks, tablets, and mobile phones) employ rolling shutters, i.e., each line of the frame is exposed to light at slightly different times. This sliding window creates image artifacts when fast moving objects, such as the eye, are present in the scene being captured. Also, because DL uses two light sources, it is possible that during an image scan part of the image is illuminated by one light and the rest of the image is illuminated by the other light source, as illustrated in Fig. 2.

To reduce these artifacts, we have proposed in [2] the use of stroboscopic differential lighting (SDL) controlled by software. This technique is described next.

3 Stroboscopic differential lighting (SDL)

In [2], we have described how stroboscopic lighting can be synchronized with rolling-shutter digital cameras. The idea is to fire one very short light pulse for every frame. The use of short light pulses allows low-end cameras to capture very sharp images (reduces motion blur) and reduces artifacts due to the rolling-shutter. Nonetheless, when the lighting is not correctly synchronized with the camera frames other artifacts such as those shown in Fig. 2 are created. The dark stripes correspond to sensor lines that were not lit by the light pulses.

Our method exploits the dark stripe artifacts to synchronize the lighting. In [2] we presented the computer vision algorithms to detect the stripe and to compute its spatial and temporal properties that are used to adjust the lighting parameters to conceal the stripes within hidden image lines. The basic idea is to first compute the position of the stripe by computing a vertical integral image, i.e., a column vector where each element corresponds to the integral of an image line. Once the stripe position is detected, the stroboscopic pulses are modulated to shift the stripe toward the hidden lines of the camera sensor.

One limitation of the method described in [2] was that it relied on the knowledge of sensor parameters. Because this information is not always available, the parameters had to be manually adjusted for some camera configurations used in the experiments. In [3], we presented a new solution that dynamically estimates the camera sensor exposure and number of lines from the dark stripe artifacts. Starting with a coarse estimation of the sensor parameters, the position and height of the stripe is computed using the vertical integral image. The stripe parameters are then used to refine the estimation of the sensor parameters before the adjustment of the firing of the illuminators, until a clear picture (without artifacts) is obtained.

While in previous papers we have focused on the SDL hardware and synchronization issues, the focus of this paper is on the computer vision software for accurate detection of eye features for gaze estimation. In the following subsections, we describe how the pupil and corneal reflections are detected using SDL.

3.1 Segmentation of pupil candidates

With the camera and light sources synchronized, the difference between two consecutive frames containing bright and dark pupil images can be used to detect pupil candidates by thresholding [27] as seen in Fig. 3.

The resulting dominant blob most likely corresponds to the overlap pupil region. Instead of using a fixed threshold value to compute the blob, we initially use an adaptive threshold technique similar to the procedure described in [22] to detect corneal reflections. The threshold is computed using (1) where $\mathbb{H}$ is the inverted cumulative histogram of the difference image and A is a geometric constraint used to eliminate noise, denoting the minimum expected pupil area in pixels.

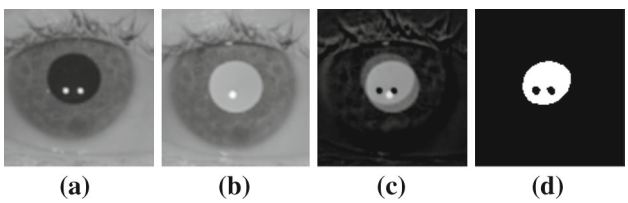

Fig. 3 Pupil candidate segmentation. **a** dark pupil image; **b** bright pupil image; **c** difference between bright and dark pupil images; **d** difference image thresholded

$$\text{threshold} = \arg\min_i |\mathbb{H}_i - A| \tag{1}$$

$\mathbb{H}$ is computed by (2), where i and j are bin numbers, h_j is the number of pixels that falls into the intensity interval defined by bin j, and k is the total number of bins.

$$\mathbb{H}_i = \sum_{j=i}^{k-1} h_j, \quad i = 0 \ldots k-1 \tag{2}$$

The algorithm varies the threshold from high to low values in large steps, speeding up the convergence. A new threshold is computed as the intensity which provides enough pixels to fill up an ellipse that encloses the contour of the largest area traced from the binarized image in the current iteration. We calculate the new threshold using (1) with A as the ellipse area.

Assuming that the pupil region corresponds to the largest elliptical blob, its area increases as the threshold is lowered. The search stops when the ratio between the largest blob and the remaining smaller blobs starts to decrease [22]. The threshold that maximizes the area A of the overlap region is selected. Additional geometric constraints regarding the expected size, shape, and position of the pupil are used to filter false candidates. Once this threshold is computed its value is used in future detections. The threshold is recomputed again only after long periods of miss or false pupil detections.

3.2 Segmentation of the corneal reflections

A similar adaptive threshold procedure based on the inverted cumulative histogram method is used to segment the corneal reflections (CRs) generated by the IR light sources. The reflections appear as bright small spots in the pupil image and they are commonly used to improve gaze estimation results [28]. Quite often they are within the pupil region and create artifacts in the difference images as seen in Fig. 3.

Geometric constraints such as the expected size and position of the CRs are used to filter some of the false positive candidates. Unlike the pupil though, other candidates might remain, particularly due to the tear layer near the eyelids, and near eyelashes.

To improve the robustness of the CR detection, each CR candidate g is modeled as a 5-tuple $(l_g, c_g, r_g, d_g, \hat{r}_g)$, where l_g is the number of iterations in which g is present; c_g is the coordinate of the reflection center, r_g is the radius of the enclosing circle; d_g is the distance to the pupil center; and $\hat{r}_g$ is the ratio between the number of segmented pixels within the circumference with radius r_g and the number of pixels within the circumference with radius $r_g + 1$, both centered at c_g. At each iteration of the adaptive threshold method, an ordered list of the corneal reflection candidates is stored. The candidates are sorted according to a quality function Q defined as

$$Q(g) = ((l_g + 1) \cdot r_g \cdot (1.0/(d_g + 0.1)) \cdot \hat{r}_g), \tag{3}$$

Therefore, the best quality CRs are those that are brighter $(l_g + 1)$, larger (r_g, within an expected range), closer to the pupil center $(1.0/(d_g+0.1))$, and shaped like a circumference ($\hat{r}_g$). The computation of the threshold stops when a number of appropriate CRs are detected and are stable between two iterations.

The position of the center (x_c, y_c) of one CR is estimated as the normalized center of mass that weights the pixels according to their intensities as follows:

$$(x_c, y_c) = \frac{1}{N} \sum_{i=1}^{N} (e^{\varrho \cdot (I(x_i, y_i) - 1)} \cdot (x_i, y_i)) \tag{4}$$

where N is the total number of segmented pixels that belong to the CR, the function $I(.)$ returns the pixel intensity normalized to $[0, 1]$ and ϱ is a weighting constant. Higher values of ϱ are used to select the center closer to the brightest pixel while with small values all pixels are considered. Two weighting constant values are used, one for the bright pupil images and one for the dark pupil images. The weighting was introduced to reduce biasing on the center estimation when the CR surrounding pixels are bright. (4) assumes that the CR intensity profile follows a symmetric bivariate Gaussian distribution such as the one described in [22].

3.3 Pupil refinement with subpixel accuracy

Tough the pupil overlap region can be used for gaze estimation [25], more accurate results can be achieved using the true pupil contour. Starting from the pupil candidate obtained from thresholding, its contour can be described by an ellipse (e_o), defined by the 4-tuple $(\mathbf{a}, \mathbf{b}, C, \hat{\theta})$, where $\mathbf{a}$ and $\mathbf{b}$ are the major and minor axes, $C = (c_x, c_y)$ is the ellipse center, and $\hat{\theta}$ is the rotation angle. The pupil refinement consists of projecting a number of rays $R_k, k = 1, \ldots, m$ outwards from the pupil center to detect the actual pupil edges in the current frame, similar to [29], as seen in Fig. 4. The length of each ray is proportional to the length of the pupil principal axis. The edge pixels with subpixel accuracy are then used to estimate the ellipse parameters.

Observe in Fig. 4 that rays intersecting the CRs are discarded when they are detected within the pupil region to avoid further interference in the computation of the pupil contour.

Each ray profile is transformed to a column vector as seen in Fig. 5 using bilinear interpolation. On each column (ray profile), a one-dimensional convolutional Gaussian derivative edge detector is applied (result seen in red). Though the

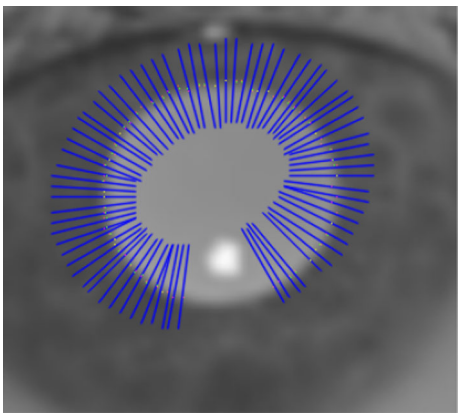

Fig. 4 Rays used to refine the edges of a bright pupil

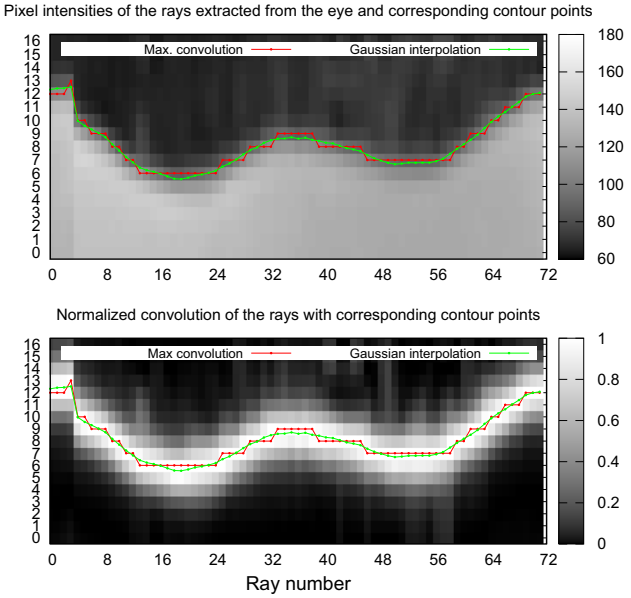

Fig. 5 The intensity profiles of rays computed from a bright pupil image. (Top) Rays resulting from bilinear interpolation of the pupil image along the ray trajectory. (Bottom) Results of the edge detector convolution along each ray. The red line shows the strongest filter response location, while the green represents the Gaussian interpolation with subpixel accuracy (color figure online)

peak response in general corresponds to the pupil–iris boundary, other strong responses might be caused by eyelashes and scene reflections. To lower the effect of such noisy rays, a normalized weight is used to sort the rays as follows:

$$W_k = \max_{j=1\ldots|R|} \left(\frac{R_k^*(j)}{\sum_{i=1}^{|R|} R_k^*(i)} \right) \tag{5}$$

where $R_k^*(j)$ denotes the edge response at position j of ray R_k, and W_k the weight attributed to ray R_k. Typically, 10% of rays with the lowest weight are discarded.

To further improve the robustness of the technique against outliers, a second filtering is performed, this time is based on

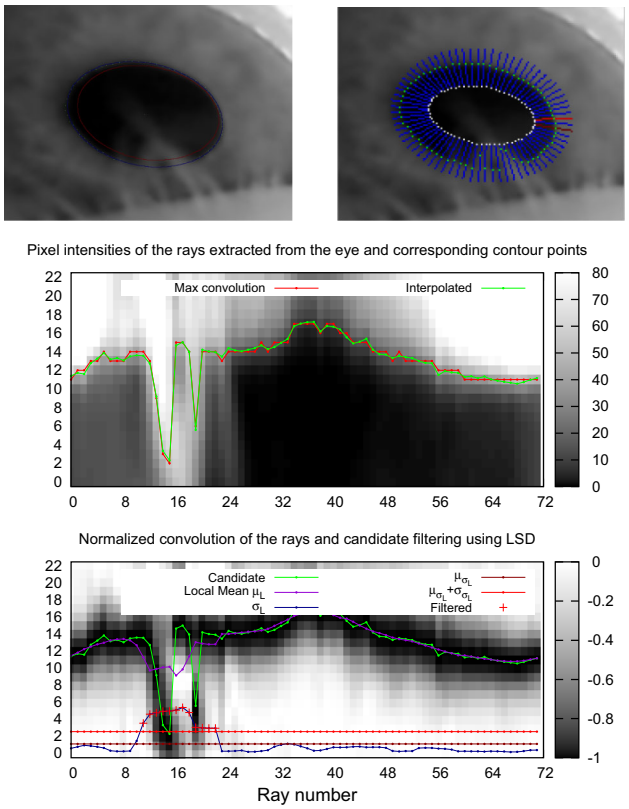

Fig. 6 Top: intensity profile of each ray extracted from a dark pupil image and corresponding convolutions. Bottom: the result of filtering using local statistics

local statistics of the spatial distribution of candidate positions. Consider the array E containing the pixel locations of the peak edge responses of each ray, sorted by the associated angle of projection. Each position in this array is a candidate to be part of the pupil contour and, therefore, we expect the values in E to vary smoothly. To remove outliers, we compute an array (σ_L) with the local standard deviation (LSD). Each value in σ_L contains the standard deviation (SD) of a neighborhood around the corresponding value in E (the neighborhood is typically of size 7). Outliers are expected to have high LSD, which are detected by thresholding as the candidates with values higher than the mean plus the SD over all LSD, i.e., threshold $= \mu_{\sigma_L} + \sigma_{\sigma_L}$. Figure 6 shows a pupil partially covered by eyelashes and the candidates filtered by this mechanism.

For the remaining rays, subpixel accuracy is obtained by interpolating the values around the pixel with strongest edge response similar to [6]. Figure 5 shows the intensity profiles of the rays computed from a bright pupil image and the corresponding strongest responses along with the results of the Gaussian interpolation (shown in green).

The computation of the interpolation is very efficient and can improve the accuracy of each pupil boundary estimation by 0.5 pixel. Considering a worst-case scenario where the

pupil center is shifted by 0.5 pixel due to quantization error, this could represent an improvement in gaze estimation accuracy of about 0.5°, when the magnitude of the PCR vector is small.

An ellipse is fitted to the remaining candidates based on the direct least-squares method [9].

Instead of using the overlap area to define the pupil candidate, the actual pupil countour is described by an ellipse e_o, defined by the 4-tuple:

$$e_o = (\mathbf{a}_K, \mathbf{b}_K, C_K, \hat{\theta}_K). \tag{6}$$

The index K denotes keyframes, i.e., frames where the pupil ellipse is fully computed using this subpixel process. The next section shows how keyframes can be used to reduce computational cost by exploiting natural eye behaviors.

When the ellipse e_o is larger or smaller than the expected size and shape of a pupil or e_o is not completely contained in the image, the refinement process stops and reports that no pupil was detected in the frame.

4 Extended temporal support

Basic eye movements can be classified into fixations, saccades, and smooth pursuits [31]. Because our vision is foveated, during a fixation the target of interest must be projected onto the fovea to be perceived at the highest resolution. The fovea only covers a small region of the retina, so the eye must stay somewhat stable during fixations. To perceive all the details of large objects, the eye must fixate on several locations. A saccade is a fast ballistic eye movement that moves the eye to different locations. Pursuits are eye movements used to foveate on moving objects, such as a flying bird.

Differential lighting works better when the overlap between the bright and dark pupil images is large. During saccades, the overlap region between two consecutive frames may become quite small for slow frame-rate cameras. Fast frame-rate cameras are used when saccades are required to be tracked. Nonetheless, slow frame-rate cameras are still adequate when tracking saccades are not important, e.g., for most gaze interaction applications.

Regardless of camera frame-rate, a DL eye tracker typically detects the pupil at every frame, considering the overlap from the previous frame. Though the pupil position could be predicted during pursuits by regular computer vision tracking algorithms, the erratic behavior of the eye during saccades makes pupil positions hard to predict.

Instead of using consecutive frames to estimate the pupil positions, our extended temporal support (xTS) technique computes most pupil displacements from selected keyframes, speeding up the computation of the new pupil position.

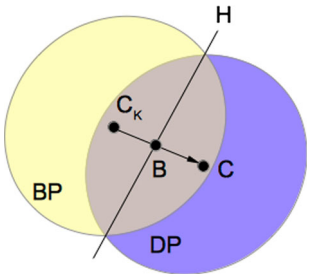

Fig. 7 xTS pupil position estimation process using a bright pupil (BP) as keyframe. A dark pupil (DP) center C is computed as a function of the keyframe pupil center C_k and the blob ellipse center B

The basic idea is shown in Fig. 7. Consider as keyframe a bright pupil (BP) image. When any dark pupil (DP) image with sufficient overlap is grabbed, the new pupil position C can be estimated as a function of the position of the pupil C_K in the keyframe and the center B of the overlap blob, computed as the center of mass of the blob. Assuming that BP and DP are about the same size, C can be computed as

$$C = B + (B - C_K). \tag{7}$$

The xTS algorithm maintains both a dark and a bright pupil keyframes. The algorithm for pupil tracking using extended temporal support with SDL (xSDL) uses those keyframes, whenever they are available, to compute the overlap region from the difference image of the most current frame and its appropriate keyframe (dark or bright). When the overlap is significant (typically, the minimum overlap width is set to be at least half the length of the keyframe's pupil minor axis), the new pupil center used for gaze estimation is computed using (7). Otherwise, xSDL tries to detect the pupil in the current frame using the previous frame. If successful, the two consecutive frames are considered as a new pair of keyframes (dark and bright). In case no pupil is detected and no keyframe is available, the algorithm continues to detect pupils using consecutive frames until keyframes are once again available. Long periods without any pupil being detected is possible during blinks for example.

Algorithm 1 gives a pseudocode overview of xSDL.

Observe the control variable `until_reset` in Algorithm 1 that is used to reset xSDL after K consecutive frames. The idea is to avoid long periods of gaze estimation without recomputing the accurate pupil contour (keyframe) to improve the overall accuracy of the system and limit the accumulation of errors. In Sect. 5, we investigate the influence of K in the system's accuracy and processing time.

```
Cur = GrabFrame(); /* bright or dark pupil      */
continue = True; /* set by external events     */
while continue :
    Prv = Cur;
    Cur = GrabFrame();
    P, CR = detectFullSDL(Prv, Cur);
    /* see section 3.1 to 3.3                  */
    estimateGaze(P, CR);
    if good( P, CR ) :
        keyfs = setKeyframes( Prv, Cur );
        until_reset = K;
        /* resets after K frames               */
        while good(P, CR, keyfs) and until_reset :
            Prv = Cur;
            Cur = GrabFrame();
            P, CR = detectXTS( Cur, keyfs);
            estimateGaze(P, CR);
            until_reset - = 1 ;
```

Algorithm 1: Overview of the xSDL.

5 xSDL empirical evaluation

We compare the performance of xSDL with the following four eye tracking algorithms: Starburst [22], ExCuSe [10], ElSe [11], and the method proposed by Świrski et al. [37].

Li and Parkhurst [22] introduced Starburst, a hybrid method that integrates feature-based and model-based approaches to detect the pupil. Starting from a point within the pupil, rays are projected radially to detect pupil–iris boundary points, i.e., where the derivative is larger than a threshold. Those feature points are used to fit an ellipse using RANSAC. The process is repeated starting at each feature point and projecting rays on the opposite direction. To reduce possible bias from the selected initial point, the process is iterated replacing the start point by the average location of all feature points until convergence, i.e., the averaged center differs less than a given threshold from the starting point location. An ellipse is fitted to the feature points using RANSAC, followed by an image-aware model-based optimization used to improve the fitting. The algorithm also finds the CR using adaptive thresholding and removes it using interpolation.

ExCuSe (Exclusive Curve Selector), developed by Fuhl et al. [10], is based on edge filtering and oriented histograms calculated via the Angular Integral Projection (AIP) function. The method follows different processing flows depending on the normalized image histogram. When a bright peak is detected, the pupil is estimated using an edge-filtering approach. The edge-filtering comprehends several steps applied to the Canny-edge image. The first step discards single pixels, small rectangles, and straight lines. The next step selects the curve that most likely encloses the pupil (the darkest area). Finally, an ellipse is fitted using all points in the selected curve with a direct least-squares method. In case a peak is not detected in the normalized image histogram, a coarse pupil position is estimated using AIP functions on a thresholded image followed by a refinement step. Four AIP are computed 45°apart, and the pupil position is assumed to lie in the intersection of the strongest function responses. The pupil center is then refined by ellipse estimation similar to the one employed by the Starburst method [22].

ElSe (Ellipse Selector), from Fuhl et al. [11], is also based on edge filtering. After Canny-edge filtering, edges are filtered out (using edge thinning followed by edge straightening) and separated so that every edge has elliptical shape. The next step selects the best ellipse fitted to the edges, according to their area, shape, and intensity ratio between the inside and outside area. In case no ellipse is found (e.g., due to motion blur), a coarse estimation of the pupil position is computed using convolution in the downscaled image, followed by a refinement step. As this second part of the method always computes an ellipse, a validation is performed as a last step to avoid returning a pupil position for a closed eye.

The last method used in the evaluation is the one proposed by Świrski et al [37]. The method uses a Haar-like feature detector to initially find a rough estimation of the pupil location. The intensity histogram of a region around the coarse position is clustered using k-means to refine the pupil center. Finally, the algorithm estimates the pupil contour with an image-augmented RANSAC ellipse fitting. Robustness and accuracy are improved by employing a support function that weights inliers according to their gradient direction and magnitude, preferring sets of points which agree with the ellipse gradient.

All these algorithms have a C or C++ source code available for download. The Starburst implementation used is available online [22] in both C and MATLAB versions. The C version lacks the model-based optimization step which we have added for completeness. The experiments with Starburst, ElSe, and ExCuSe, used their default parameters, as provided by their source codes. The experiments with Świrski et al. algorithm also used default parameters, except for the pupil minimum and maximum radii, for which the technique showed to be quite sensitive. This parameter was adjusted once per user.

The experiment was designed to compare the performance of each method in a typical gaze estimation task. Ten people (4 females) from 30 to 59 years old (mean 36.2) volunteered for the experiment. All participants had normal vision without the need of any correction lenses.

The participants' task was to fixate their gaze on 35 small circular targets (7 pixels in diameter) arranged in a 5×7 grid. Targets were displayed one at a time in random order for about 3 seconds. Only the 800 ms interval 500 ms before the next target presentation were considered for analysis. A 1680×1050 resolution, 60 Hz, 22" monitor was used, with a 20 pixel border at each side of the monitor. The distance between the monitor and the participants' eyes was about 70 cm.

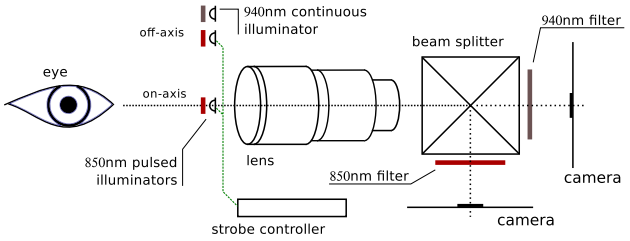

Fig. 8 System prototype used to compare eye tracking methods based on xSDL and dark pupil-only methods

5.1 Apparatus

The xSDL eye tracker prototype was built using a low-cost, off-the-shelf Play Station 3 video camera [36]. Because all algorithms except xSDL work with the dark pupil image, our prototype actually included 2 identical cameras and two independent lighting systems: one to capture the sequence of dark and bright pupil images using structured illumination needed for xSDL, and the other to capture only the dark pupil images for the other methods. Figure 8 depicts this setup.

The structured illumination system (used by the xSDL method) was composed of one on-axis and one off-axis light sources (sets of 850 nm LEDs) controlled by an Arduino board.

The continuous illumination system responsible for producing dark pupil images only was built using 940 nm LEDs. A flat-convex lens of 20 mm focal length was used to improve the LED efficiency. This source was also filtered using a narrow bandpass filter centered on the emitter wavelength.

After passing through the objective lens, the light was separated into two paths by a custom made beam splitter. Each path projected light to a different video camera: one for xSDL algorithm and the other for dark pupil-only algorithms. To avoid light interference between the different illumination systems, bandpass filters were placed in front of the camera sensors: a 850 nm pass filter for the xSDL camera and a 940 nm filter for the other camera. Thus, the eye image was captured from the same perspective using both illumination systems (the stroboscopic technique and constant illumination), as shown in Fig. 9.

To reduce camera biasing due to different spectral responses, such as increased noise at 940 nm due to a reduced sensitivity in this wavelength, both cameras were initially adjusted to the same fixed settings, and then the power of each illuminator was calibrated to generate images of the same brightness.

5.2 Data analysis

We used 9 (out of the 35) points to calibrate a second-order polynomial function. This function was used to estimate the gaze position at all of the 35 points to compute the gaze estimation error.

We use three metrics to evaluate the methods: pupil detection robustness, gaze estimation accuracy and precision, and processing time.

Pupil detection robustness refers to the proportion of frames that a method is able to detect the pupil. All images collected were manually inspected and only those containing the pupil image were used for evaluation. For xSDL, the sequence of frames contained bright and dark pupil images. For all other methods, the sequence of frames contained only dark pupil images. Because Świrski's method and Starburst always return a pupil boundary even though these methods did not accurately detect the pupil, we decided to consider frames with a gaze estimation error above 5° as no pupil detection.

The accuracy of each method is defined as the average gaze estimation error over all 35 target locations. Error in gaze estimation is the difference between the actual target position and the estimated gaze position in degrees of visual angle. The precision is given by the standard deviation of each participant's error.

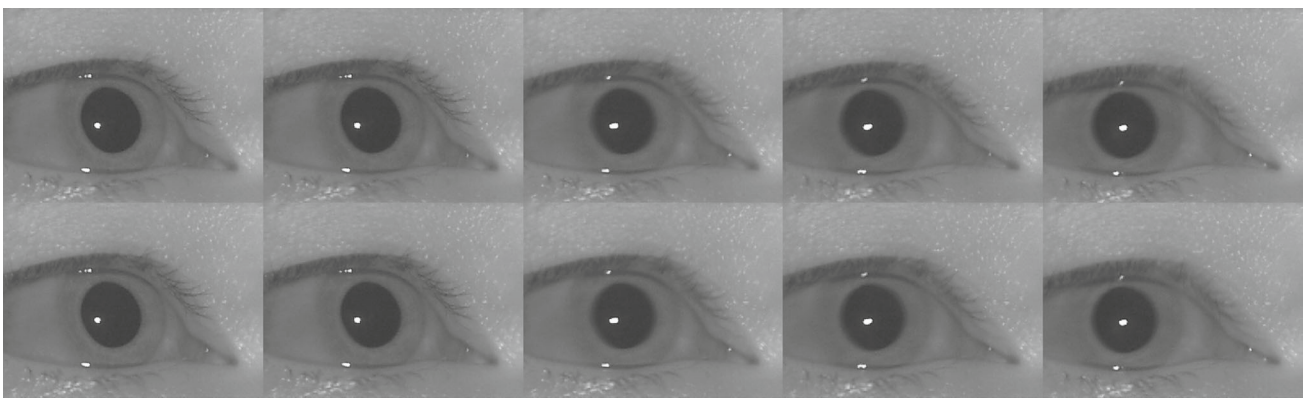

Fig. 9 Example capture of a participant with an ongoing saccade. On top, xSDL camera images and on bottom, continuous 940 nm illuminated images

The processing time is computed as the average time a method took to process each frame. All methods were timed using the same computer platform, a notebook equipped with an Intel i7-7700HQ Processor with 8 logical cores running at 2.8 GHz and 16 GB of RAM.

6 Results

This section presents the experimental results of pupil detection robustness, gaze estimation accuracy and precision, and processing time.

6.1 Influence of the reset rate (parameter *K*)

The performance of xSDL can be controlled by the reset rate parameter K. For small K values, we expect xSDL to be as accurate as SDL and, as K increases, the accuracy might drop to improve speed. In this section, we will consider $K = 0$ to be the performance of SDL (i.e., the pupil contour is computed at every frame with subpixel accuracy) and $K = \infty$ to be the fastest xSDL, i.e., a new keyframe is computed only when the overlap becomes small. In our experiments, the pupil contour for $K = \infty$ is recomputed when the overlap becomes smaller than half the radius of the pupil. We will also consider the intermediate values for $K \in \{5, 15, 30, 100\}$.

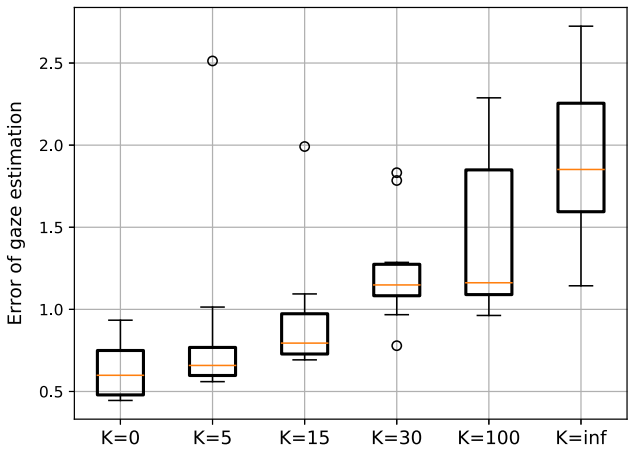

Fig. 10 Boxplot of accuracy for each K computed with data from the ten participants

Figure 10 shows a boxplot of the accuracy of xSDL for each K value, where the gaze estimation error was computed for all frames with a successfully detected pupil. We run a Friedman test and found a significant effect of K on accuracy ($\chi^2(6) = 50.54$, $p < 0.01$). A post-hoc Wilcoxon signed rank test with Holm correction showed that $K = 0$ (grand median 0.60°) was significantly more accurate compared to $K = 15$ (grand median 0.80°), $K = 30$ (grand median 1.04°), $K = 100$ (grand median 1.27°), and $K = \infty$ (grand median 1.70°), $p < 0.05$ in all cases. We also found a significant difference between $K = 15$ and $K = 100$ and $K = \infty (p < 0.05)$.

The distributions of the average error for all participants and K values are shown in the histograms in Fig. 11. Observe that, as expected, the number of pixels with large error increases as K becomes larger.

Figure 12 shows the boxplot of the processing time per frame for each K value, considering all ten participants. The histograms in Fig. 13 show that the processing time distribution is bimodal when K is greater than zero, as expected since some frames only compute the overlapping region. We can also observe that the first mode (that correspond to fast frame processing) increases with higher values of K.

We run a Friedman test and found a significant effect of K on processing time ($\chi^2(6) = 55.48$, $p < 0.01$). A post-hoc Wilcoxon signed rank test with Holm correction showed that the only computation times to differ statistically were $K = 15$ (grand median 0.52 ms) to $K = 30$ (grand median 0.48 ms), $K = 30$ (grand median 0.48 ms) to $K = 100$ (grand

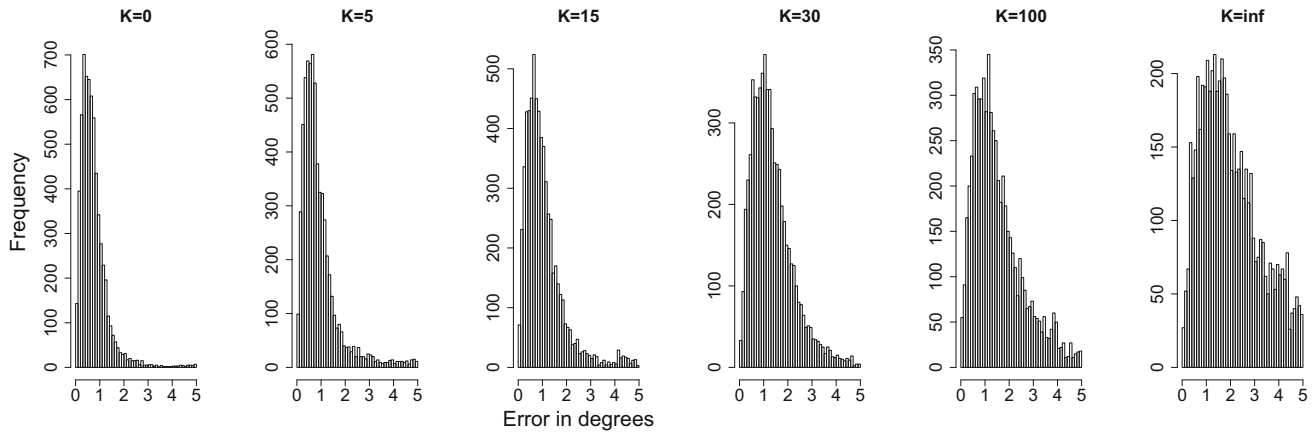

Fig. 11 Histograms of the gaze estimation error for all participants and K values

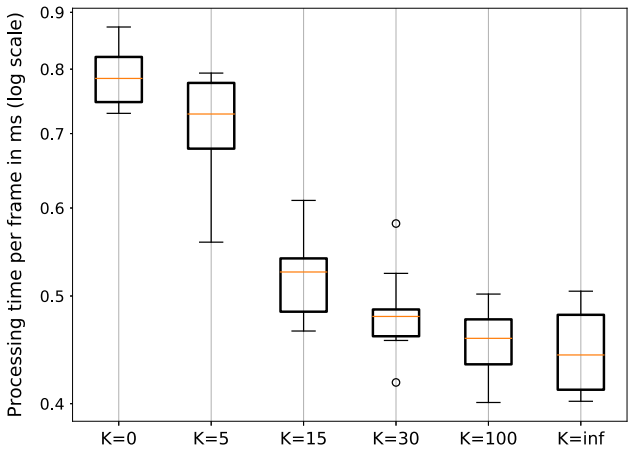

Fig. 12 Boxplot of processing time per frame for each K computed with data from the ten participants. Note that the vertical axis has a logarithmic scale to facilitate visualization

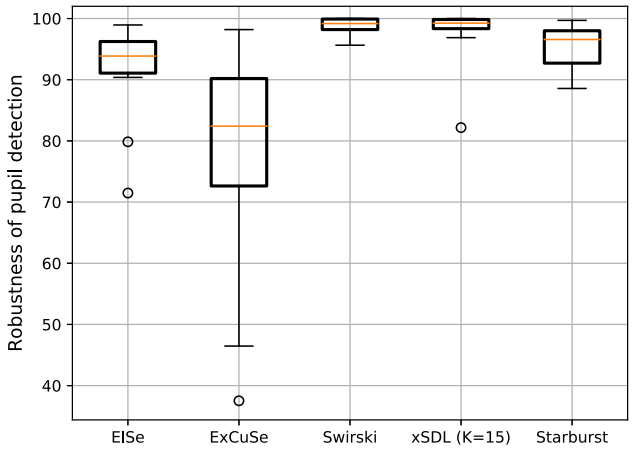

Fig. 14 Boxplot of pupil detection robustness for each method computed with data from the ten participants

median 0.46 ms) and $K = \infty$ (grand median 0.44 ms), and $K = 100$ to $K = \infty (p < 0.05)$.

Due to the good tradeoff between accuracy (less than 1°) and speed (close to 2 KHz in the computer used in the experiments), in the remainder of this section we will use $K = 15$ to compare the performance of xSDL with other state-of-the-art eye tracking methods.

6.2 Pupil detection robustness

For each participant × method in all captured frames, we computed the proportion of frames where the pupil was detected (i.e., with a gaze estimation error below 5°). Figure 14 shows the boxplot for all methods.

Because data are not normally distributed, as can be observed in Fig. 14, we ran the non-parametric Friedman test. Results showed a significant effect of algorithm ($\chi^2(4) = 24.32$, $p < 0.01$). A post-hoc Wilcoxon signed rank test with Holm correction showed that xSDL (grand median 99.25%) was significantly different than ElSe (grand median 93.87%) and ExCuSe (grand median 82.41%). The method of Świrski (grand median 99.18%) was significantly different than ElSe, ExCusE, and Starburst (grand median 96.57%), $p < 0.05$ in all cases. All other combinations were not statistically significant.

6.3 Gaze estimation accuracy and precision

For each method, the gaze estimation error was computed for all frames with a successfully detected pupil. For all methods, the error distribution was right-skewed, as shown in Fig. 15, though ElSe and ExCuSe had a larger dispersion. Because of the different distributions, we computed the median error of each participant for each method. Figure 16 shows the boxplot of gaze estimation error for the ten participants for each method.

As can be observed in Fig. 16, xSDL (grand median 0.79°) and Świrski (grand median 0.8°) had a very similar accuracy, followed by Starburst (grand median 1.1°). ElSe (grand

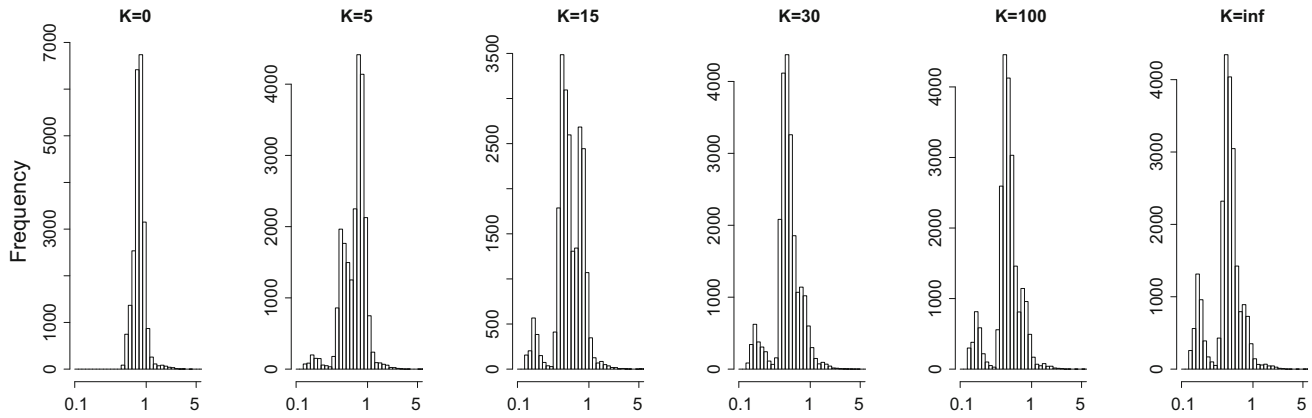

Fig. 13 Histograms of the processing time per frame for all values of K and all participants. Time in the horizontal axis is in log scale

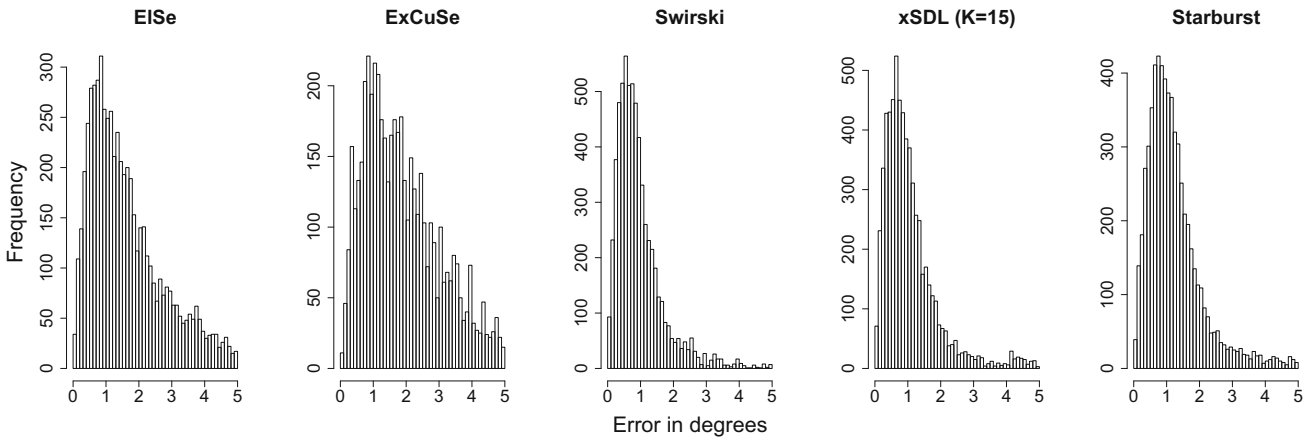

Fig. 15 Histograms of the gaze estimation error for all participants and methods

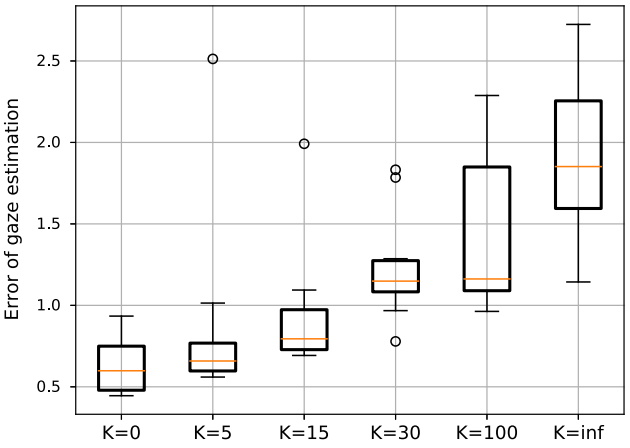

Fig. 16 Boxplot of accuracy for each method computed with data from the ten participants

median 1.5°) and ExCuSe (grand median 1.91°) had the larger error. We run a Friedman test and found a significant effect of algorithm on accuracy ($\chi^2(4) = 30.48$, $p < 0.01$). A post-hoc Wilcoxon signed rank test with Holm correction showed that xSDL was significantly more accurate compared to ElSe and ExCuSe, $p < 0.05$ in all cases. Świrski's difference compared to ElSe, ExCuSe, and Starburst was also significant ($p < 0.05$).

6.4 Processing time

We measured the time (in milliseconds) to process each frame. The histograms in Fig. 17 show that the processing time distribution varied largely among methods. For Świrski's, the distribution was more scattered compared to all other methods. Świrski, ExCuSe, and xSDL had a bimodal distribution that reflects the existence of different execution paths in their algorithms. Processing time of ElSe and Starburst had a unimodal distribution.

We computed the median processing time per frame for each participant and method. Figure 18 shows the boxplot of processing time for the ten participants. Because of the large difference in processing time among methods, we used a logarithmic scale in Fig. 18 to facilitate visualization. As can be observed, xSDL had the smaller processing time (grand median 0.52 ms) and also the smaller variation (IQR 0.057 ms). Starburst had the second lowest average running time (grand median 1.29 ms, IQR 0.46 ms). On the other hand, ElSe had the larger processing time (grand median 14.54 ms), followed by ExCuSe (grand median 6.71 ms, IQR 2.27 ms), and Świrski (grand median 4.61 ms, IQR 3.23 ms). Świrski had the larger variation. A Friedman test showed a significant effect of algorithm ($\chi^2(4) = 36.88$, $p < 0.01$). A post-hoc Wilcoxon signed rank test with Holm correction showed that processing time per frame differed significantly between all methods ($p < 0.05$ in all cases), with the exception of Świrski and ExCuSe ($p = 0.77$).

7 Discussion

Experimental results support that xSDL requires less processing power compared to the other state-of-the-art methods that were evaluated, while still presenting good robustness, accuracy and precision. For smaller K values, the xSDL can outperform all the tested methods in robustness, accuracy and precision while still preserving the computational efficiency.

The improved pupil detection robustness of xSDL could be attributed to the differential lightning technique, since it takes advantage of the sequence of dark and bright pupil images generated by the alternating light sources. The overlapping pupil area computed by the difference of consecutive (dark–bright or bright–dark) images serves as a robust estimator of the pupil location in the image, thus discarding false positives. Dark pupil image-based methods rely on histogram

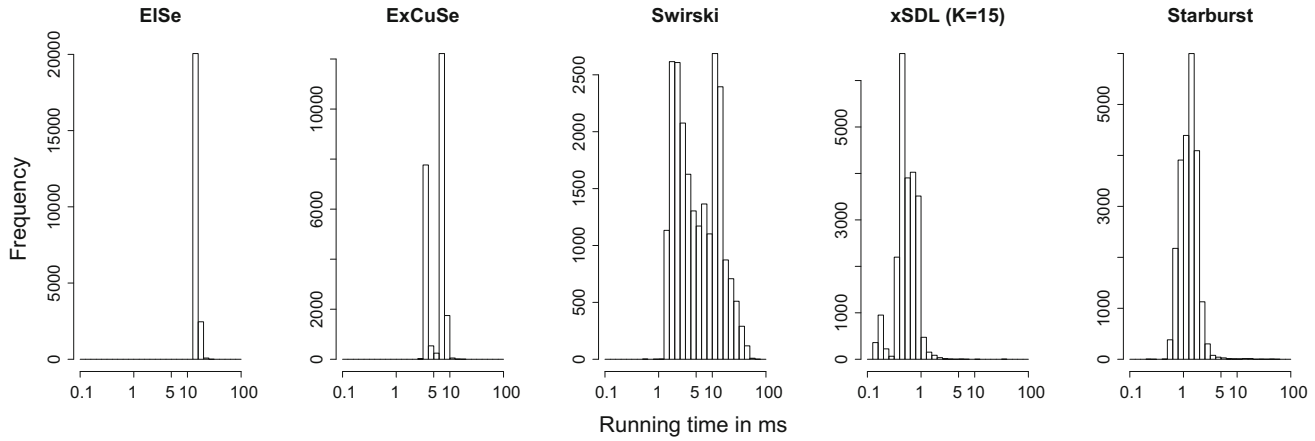

Fig. 17 Histograms of the processing time per frame for all methods

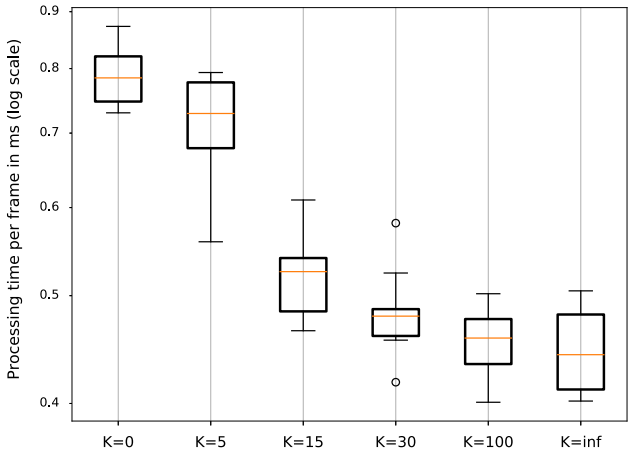

Fig. 18 Boxplot of processing time per frame for each method computed with data from the ten participants. Note that the vertical axis has a logarithmic scale to facilitate visualization

analysis (such as ExCuSe and Świrski) or high contrast areas (borders detected by Canny-edge filter or derivative) as a rough estimation of the pupil location. These feature-based approaches could result in areas not corresponding to the pupil to be detected as such, e.g., a black blob in the captured image. Among these methods, Świrski was the only one that had a robustness close to xSDL, but at the cost of a much larger running time and hence computational power.

xSDL was also among the three most accurate and precise methods. Accurate gaze estimation is very important for many applications, such as to point at small visual targets in a gaze-controlled interface such as a virtual keyboard. More accuracy allows to include more objects in an interface, e.g., punctuation, accents, and control commands in a virtual keyboard. Accuracy also reduces the number of selection errors. Precision is also very important, since gaze estimation is more consistent, improving the quality of gaze data. Better quality implies a better usability in gaze-controlled interfaces, and also more accurate results in medical applications that use gaze information.

Computational efficiency is another advantage of xSDL, as shown by the average processing time of each method. Not only xSDL is the fastest method, but it also holds the smallest dispersion among all methods. Our prototype was tested at 60 frames per second to keep infrared radiation within safe limits, as two independent illumination sources were in use. By reducing the infrared radiation emission (i.e., by using our technique alone), it is possible to run the prototype at 187 frames per second with the same low-cost PS3 video camera. By using faster cameras it would be possible to run xSDL at almost 2000 Hz with a similar computer processor, given the short average processing time of 0.59 ms per frame. It is noteworthy that ElSe and ExCuSe had a bimodal distribution processing time. This could be explained because those two algorithms have 2 possible execution paths. If the main execution path fails, then those methods use a simpler approach in an effort to detect the pupil. Though we did not verified whether ElSe and ExCuSe indeed ran different execution flows in our experiment, this hypothesis explains the bimodal distribution in their processing time histograms. Świrski et al. method possess the largest variation, with a mean processing time of 8.02 ms (IQR 3.23 ms).

7.1 Limitations and future work

Similar to any feature detection-based eye tracker using near IR lighting, xSDL will not perform well in bright natural illuminated scenarios. It can be also affected by occlusion of the pupil and corneal reflections by glasses and eyelashes. The ellipse fitting might also fail to detect very distorted pupil images generated when the camera is placed highly off the optical axis of the eye, typical in some head mounted configurations [12,34].

Though xSDL can be used with any digital off-the-shelf camera, it requires additional custom hardware to control the firing of the stroboscopic lighting. Our prototype is constituted of a simple Arduino board that, though simple to use, might require some knowledge and experience to build. Nonetheless, xSDL arises as a low-cost, high-performance, and robust eye tracker alternative for gaze-based applications.

We are currently working on an autonomous version of xSDL. The idea is to create a camera "attachment" that eliminates the physical connection of the computer to the xSDL hardware. The attachment would turn any digital camera into a high-quality portable eye tracker.

8 Conclusion

Differential lighting (DL) is a robust and computationally efficient pupil detection technique developed in the 90's for analog cameras. We have shown, in previous papers, that the use of stroboscopic differential lighting (SDL) allows DL to be used with any modern and low-cost rolling-shutter digital camera. We believe that its improved performance will compensate the trouble of building the extra synchronization hardware (in case it is not already available) in gaze-enhanced wearable computers.

This paper introduced the extended temporal support computer vision algorithm that further improves the performance of SDL, we called xSDL. Traditional computer vision tracking techniques are not used for eye tracking in general because they are either computationally expensive (such as correlation-based methods) or do not perform well due to the unpredictable eye movements during fixations. SDL detects the pupil at every frame by simple image differencing and works better when the overlap between bright and dark pupil images is large, i.e., it works well during fixations. xSDL selects keyframes (dark and bright pupil images) to serve as reference images. The position of the pupil in keyframes is computed with high accuracy. This position is then refined by xSDL using the translation of the overlap region computed from the difference image. Our results show that xSDL configured to recompute the pupil contour at least every 15 frames ($K = 15$) can achieve high speed and accuracy.

To compare xSDL performance against other state-of-the-art eye tracking algorithms available in the literature, we have created a two-camera apparatus that simultaneously collects videos of the eye that are appropriate for xSDL (that requires bright and dark pupil images), and the other methods, that only use dark pupil images. Therefore, the methods were compared using videos showing exactly the same eye behaviors and with the same frame rate (and other video properties).

Our results show that xSDL outperforms those four algorithms. Using 240 line images and $K = 15$, xSDL is able to process close to 2000 fps, with an accuracy of about 0.7°and detecting the pupil practically all the time. This level of accuracy and robustness is similar to the method of Świrski et al. [37], though this alternative is much more computationally expensive.

Acknowledgements This work has been supported by the Fundação Araucária (DINTER Project UTFPR/IME-USP) and FAPESP Grant Number 2012/04426-0, 2016/0446-2, 2016/10148-3 and by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Amir, A., Zimet, L., Sangiovanni-Vincentelli, A., Kao, S.: An embedded system for an eye-detection sensor. Comput. Vis. Image Underst. **98**(1), 104–123 (2005). https://doi.org/10.1016/j.cviu.2004.07.009
2. Borsato, F., Aluani, F., Morimoto, C.: A fast and accurate eye tracker using stroboscopic differential lighting. In: Proceedings of the IEEE International Conference on Computer Vision Workshops, pp. 110–118 (2015)
3. Borsato, F.H., Morimoto, C.H.: Building structured lighting applications using low-cost cameras. In: 2017 30th SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI), pp. 15–22 (2017). https://doi.org/10.1109/SIBGRAPI.2017.9
4. Diaz-Tula, A., Morimoto, C.H.: AugKey: increasing foveal throughput in eye typing with augmented keys. In: Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems, CHI '16, pp. 3533–3544. ACM, New York (2016).https://doi.org/10.1145/2858036.2858517
5. Diaz-Tula, A., Morimoto, C.H., Ranvaud, R.D.: A mathematical model of saccadic reaction time as a function of the fixation point brightness gain. Atten. Percept. Psychophys. **77**(6), 2153–2165 (2015). https://doi.org/10.3758/s13414-015-0902-9
6. Dvornychenko, V.: Bounds on (deterministic) correlation functions with application to registration. IEEE Trans. Pattern Anal. Mach. Intell. **5**(2), 206–213 (1983)
7. Ebisawa, Y.: Improved video-based eye-gaze detection method. IEEE Trans. Instrum. Meas. **47**(4), 948–955 (1998). https://doi.org/10.1109/19.744648
8. Ebisawa, Y., Satoh, S.: Effectiveness of pupil area detection technique using two light sources and image difference method. In: Proceedings of the 15th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, pp. 1268–1269 (1993). https://doi.org/10.1109/IEMBS.1993.979129
9. Fitzgibbon, A., Pilu, M., Fisher, R.B.: Direct least square fitting of ellipses. IEEE Trans. Pattern Anal. Mach. Intell. **21**(5), 476–480 (1999)
10. Fuhl, W., Kübler, T., Sippel, K., Rosenstiel, W., Kasneci, E.: Excuse: robust pupil detection in real-world scenarios. In: Azzopardi, G., Petkov, N. (eds.) Computer Analysis of Images and Patterns: 16th International Conference, CAIP 2015, Valletta, Malta, pp. 39–51. Springer (2015). https://doi.org/10.1007/978-3-319-23192-1_4

11. Fuhl, W., Santini, T.C., Kübler, T., Kasneci, E.: ElSe: ellipse selection for robust pupil detection in real-world environments. In: Proceedings of the Ninth Biennial ACM Symposium on Eye Tracking Research & Applications, ETRA '16, pp. 123–130. ACM, New York (2016). https://doi.org/10.1145/2857491.2857505
12. Fuhl, W., Tonsen, M., Bulling, A., Kasneci, E.: Pupil detection for head-mounted eye tracking in the wild: an evaluation of the state of the art. Mach. Vis. Appl. **27**(8), 1275–1288 (2016). https://doi.org/10.1007/s00138-016-0776-4
13. George, A., Routray, A.: Fast and accurate algorithm for eye localisation for gaze tracking in low-resolution images. IET Comput. Vis. **10**(7), 660–669 (2016). https://doi.org/10.1049/iet-cvi.2015.0316
14. Hansen, D.W., Ji, Q.: In the eye of the beholder: a survey of models for eyes and gaze. IEEE Trans. Pattern Anal. Mach. Intell. **32**(3), 478–500 (2010). https://doi.org/10.1109/TPAMI.2009.30
15. Haro, A., Flickner, M., Essa, I.: Detecting and tracking eyes by using their physiological properties, dynamics, and appearance. In: Proceedings IEEE Conference on Computer Vision and Pattern Recognition. CVPR 2000 (Cat. No. PR00662), vol. 1, pp. 163–168 (2000). https://doi.org/10.1109/CVPR.2000.855815
16. Hennessey, C., Noureddin, B., Lawrence, P.: A single camera eye-gaze tracking system with free head motion. In: Proceedings of the 2006 Symposium on Eye Tracking Research & Applications, pp. 87–94 (2006)
17. Jacob, R.J.K., Karn, K.S.: Eye tracking in human–computer interaction and usability research: ready to deliver the promises. In: The Mind's Eye: Cognitive and Applied Aspects of Eye Movement Research, pp. 573–603 (2003). https://doi.org/10.1016/B978-044451020-4/50031-1
18. Ji, Q., Yang, X.: Real-time eye, gaze, and face pose tracking for monitoring driver vigilance. Real-Time Imaging **8**(5), 357–377 (2002). https://doi.org/10.1006/rtim.2002.0279
19. Kapoor, A., Picard, R.W.: A real-time head nod and shake detector. In: Proceedings of the 2001 Workshop on Perceptive User Interfaces, PUI '01, pp. 1–5. ACM, New York (2001). https://doi.org/10.1145/971478.971509
20. Kassner, M., Patera, W., Bulling, A.: Pupil: an open source platform for pervasive eye tracking and mobile gaze-based interaction. In: Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct Publication, pp. 1151–1160. ACM (2014)
21. Kempinski, Y.: System and method of diagnosis using gaze and eye tracking (2016). US Patent App. 14/723,590. https://www.google.com/patents/US20160106315. Accessed 14 Nov 2017
22. Li, D., Parkhurst, D.J.: Starburst: A Robust Algorithm for Video-Based Eye Tracking, p. 6. Elsevier, Amsterdam (2005)
23. Menon, R.V., Sigurdsson, V., Larsen, N.M., Fagerstrøm, A., Foxall, G.R.: Consumer attention to price in social commerce: Eye tracking patterns in retail clothing. J. Bus. Res. **69**(11), 5008–5013 (2016). https://doi.org/10.1016/j.jbusres.2016.04.072
24. Morimoto, C., Flickner, M.: Real-time multiple face detection using active illumination. In: Proceedings Fourth IEEE International Conference on Automatic Face and Gesture Recognition (Cat. No. PR00580), Grenoble, France, pp. 8–13 (2000). https://doi.org/10.1109/AFGR.2000.840605
25. Morimoto, C., Koons, D., Amit, A., Flickner, M., Zhai, S.: Keeping an eye for HCI. In: Proceedings. XII Brazilian Symposium on Computer Graphics and Image Processing, 1999, pp. 171–176 (1999). https://doi.org/10.1109/SIBGRA.1999.805722
26. Morimoto, C.H., Koons, D., Amir, A., Flickner, M.: Pupil detection and tracking using multiple light sources. Image Vis. Comput. **18**(4), 331–335 (2000)
27. Morimoto, C.H., Koons, D., Amir, A., Flickner, M.D.: Pupil detection and tracking using multiple light sources. Image Vis. Comput. **18**(4), 331–335 (2000)
28. Morimoto, C.H., Mimica, M.R.: Eye gaze tracking techniques for interactive applications. Comput. Vis. Image Underst. **98**(1), 4–24 (2005). https://doi.org/10.1016/j.cviu.2004.07.010
29. Ohno, T., Mukawa, N., Yoshikawa, A.: FreeGaze: a gaze tracking system for everyday gaze interaction. In: Proceedings of the 2002 Symposium on Eye Tracking Research & Applications, pp. 125–132 (2002)
30. Oliveira, D., Machín, L., Deliza, R., Rosenthal, A., Walter, E.H., Giménez, A., Ares, G.: Consumers' attention to functional food labels: insights from eye-tracking and change detection in a case study with probiotic milk. LWT Food Sci. Technol. **68**, 160–167 (2016). https://doi.org/10.1016/j.lwt.2015.11.066
31. Oyster, C.: The Human Eye: Structure and Function. Sinauer Associates, Sunderland (1999)
32. Piumsomboon, T., Lee, G., Lindeman, R.W., Billinghurst, M.: Exploring natural eye-gaze-based interaction for immersive virtual reality. In: 2017 IEEE Symposium on 3D User Interfaces (3DUI), pp. 36–39 (2017). https://doi.org/10.1109/3DUI.2017.7893315
33. Samadani, U., Zahid, A.B., Lockyer, J., Dammavalam, V., Grady, M., Nance, M., Scheiman, M., Master, C.L.: Eye tracking a biomarker for concussion in the paediatricpaediatric population. Brit. J. Sports Med. **51**(11), A5 (2017)
34. Santini, T., Fuhl, W., Kasneci, E.: Purest: robust pupil tracking for real-time pervasive eye tracking. In: Proceedings of the 2018 ACM Symposium on Eye Tracking Research & Applications, ETRA '18, pp. 61:1–61:5. ACM, New York (2018). https://doi.org/10.1145/3204493.3204578
35. Soler-Dominguez, J.L., Camba, J.D., Contero, M., Alcañiz, M.: A proposal for the selection of eye-tracking metrics for the implementation of adaptive gameplay in virtual reality based games. In: Lackey, S., Chen, J. (eds.) Virtual, Augmented and Mixed Reality: 9th International Conference, VAMR 2017, pp. 369–380. Springer, Vancouver (2017). https://doi.org/10.1007/978-3-319-57987-0_30
36. Sony, C.: PlayStation Eye Camera [Online 14 Jan 2014] (2014). http://us.playstation.com/ps3/accessories/playstation-eye-camera-ps3.html
37. 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)
38. Zhu, Z., Ji, Q.: Robust real-time eye detection and tracking under variable lighting conditions and various face orientations. Comput. Vis. Image Underst. **98**(1), 124–154 (2005). https://doi.org/10.1016/j.cviu.2004.07.012

Frank H. Borsato received his B.Sc. and M.Sc. in Computer Science at the State University of Maringá (UEM), and his Ph.D. in Computer Science from the University of São Paulo (USP). He is currently an associate professor at the Department of Computer Science at the Federal University of Technology - Paraná (UTFPR). His main areas of interest include Computer Vision, Machine Learning, and Embedded Systems.

Antonio Diaz-Tula holds a BSc and MSc in Computer Science from Universidad de Oriente, Cuba (2007 and 2010, respectively) and a PhD in Computer Science from the University of São Paulo, Brazil (2015). He has experience in the area of Computer Science, work-

ing mainly with Eye Tracking, Human Computer Interaction, Software Development, and Statistical Data Analysis. He currently works as software developer at Dr. TIS company and acts as statistician consultant.

Carlos H. Morimoto received his B.Sc. and M.Sc. in Electrical Engineering at the University of São Paulo (USP), and his Ph.D. in Computer Science from the University of Maryland at College Park. He is currently an associate professor at the Department of Computer Science at Institute of Mathematics and Statistics (IME) at USP. His main areas of interest include Computer Vision and Human Computer Interaction, with particular focus on eye tracking and gaze interaction.