Fast Detection of the Pupil Centre in Stable Light Conditions

Krzysztof Wolski*
Supervised by: Radosław Mantiuk †

West Pomeranian University of Technology, Szczecin
Poland

Abstract

We present a simple and efficient technique of the pupil centre detection. This technique is addressed to the video eye tracking solutions, in which pupil centre must be found in image of the human eye. In contrary to previous work, we assume stable light conditions that provide a correct eye image. Such conditions can be achieved in many eye tracking applications but our solution is especially addressed to the scientific activities related to the perceptual experiments. We introduce a novel cross spread technique in which it is assumed that pupil shape is similar to ellipse. In this strategy, a parallel algorithm can be applied to detect the pupil centre, which enables accurate operation in less than 2 milliseconds. We present the OpenCL-based implementation of the cross spread algorithm and its application in the real-world eye tracker. The paper shows the results of the experimental measurement of this eye tracker accuracy performed for a number of human observers. The achieved accuracy close to 1.5 degree of the visual angle is comparable to the commercial devices.

Keywords: pupil centre detection, cross spread, eye tracing, eye tracking accuracy, perceptual experiments

1 Introduction

Detection of the pupil centre in the images of the eye is a basic function of the video-based eye trackers. This type of eye trackers consists of the infrared camera and the infrared light source, which are directed towards the eye. The camera captures the image of the eye with the dark circle of the pupil (see example in Fig. 2). The pupil follows the gaze direction during eye movement. Location of its centre is used to estimate the gaze direction.

The field of view for both eyes spans more than 180° horizontally and 130° vertically, although, humans are able to see details only in the fovea — the 2° patch of the retina located in the middle of the macula. The eye muscles enable fast gaze shifting to orient the eye such that the object of interest is projected onto the fovea. There are four types of eye movements: vergence movements, vestibular ocular movements, smooth pursuit, and saccadic movements [6]. From the eye tracking perspective, the most important is the latter one. Fast (up to 900°/s) and short (10-100 milliseconds [1]) saccades move the eye to a new area of interest. They should be captured with at least 200 Hz frequency to allow accurate registration of the gaze direction. Otherwise, eye tracker can register a gaze location in the middle of the saccadic movement making its identification challenging.

In this paper we propose a novel pupil tracking algorithm called the *cross spread*. We assume that the pupil shape is similar to ellipse. Then, we use basic image processing operations to process the image of the eye and find the pupil centre. The algorithm consists of three steps: thresholding and binarization, noise reduction using median filter, and the core cross spread algorithm, which detects the pupil centre. These operations are sufficient to achieve the high accuracy of detection. However, a correct image of the eye must be delivered from the camera to avoid image analysis errors. The most important is a good visibility of the pupil on the iris background. A correct image of the eye can be taken in the stable light conditions. Such conditions may be provided e.g. during the perceptual experiments, that use eye trackers.

We implemented parallelised version of the cross spread algorithm based on the OpenCL library, which detects the pupil centre in less than 2 milliseconds. This implementation was tested with the Do-It-Yourself (DIY) eye tracker - the custom-built head-mounted eye tracking system [4]. The paper shows the results of the experimental measurement of DIY accuracy performed for a number of human observers. The achieved accuracy close to 1.5 degree of the visual angle is comparable to the commercial eye trackers.

In Sect. 2, an existing pupil detection techniques are outlined. The cross spread technique is introduced in Sect. 3, followed by the description of its parallel implementation and results of the performance tests. In Sect. 4 we present conducted experiments that measured the accuracy of the custom-build eye tracker equipped with our implementation of cross spread technique.

2 Previous work

Most algorithms for detection of the pupil centre binarise the image of the eye and filter out the noise to achieve the

*krwolski@wi.zut.edu.pl
†rmantiuk@wi.zut.edu.pl

best possible image of the pupil. Then, various scenarios are implemented to detect the centre of the ellipse, which shape reproduces the shape of the pupil.

The curvature detection algorithm [8] detects the edge of the pupil by tracing rays in all direction from the starting point located within the pupil area. For each ray, the location of the pupil edge is detected. Then, a heuristic curvature detection algorithm is used to eliminate the edge deteriorations caused by the eyelids, cilia, or corneal reflections. Finally, the parameters of ellipse, which best fits in the detected edge are computed using the least squares solution.

In the starburst algorithm [2] rather complex noise reduction technique based on Gaussian filtering, thresholding and morphological operation is used to achieve satisfactory image of eye. Then, similarly to the curvature detection technique, the rays are shot to find the pupil edge. Finally, the best fitting ellipse is determined using the RANSAC technique.

Other pupil centre detection algorithms are described in [7] and [5]. The goal of all these techniques is to achieve the best accuracy of detection. Much attention is paid to noise reduction and accurate ellipse fitting. In the following section we present simplified technique. We assume that correct image of the eye is delivered from the camera and more attention is paid to the processing speed. We propose the parallelised implementation of this algorithm adjusted to the GPU processing.

3 Cross spread technique

This section includes a detailed description of the cross spread algorithm and discussion on the drawbacks of this technique.

3.1 Detection pipeline

The whole algorithm consists of three steps presented in Fig. 1. As an input, the eye image is taken using the infrared camera. Then thresholding and median filtering is applied to binarise and denoise the image, respectively. Finally, the core algorithm is activated to detect the pupil centre and compute (x,y) coordinates of this point. The output values are expressed in pixels of the camera image.

Figure 1: The pupil centre detection pipeline.

The core cross spread algorithm is designed to search elliptic shapes. Any deviation of continuity of the edges or spatial coherence of the blob may reduce its accuracy. The possible deterioration depends mainly on the area of an artefact. Although the algorithm is designed to find the centre of ellipse, it can work correctly also for other convex shapes that are centrally symmetric.

Image thresholding

Thresholding is a simple image processing operation, which binarises the image colours. The infrared camera delivers image in the grey shades (see Fig. 2, left). After thresholding, the pupil ellipse becomes black and the rest of the pixels in an image become white. It is done based on the threshold value - empirically chosen grey value below and above which, the pixels are marked as black or white respectively. Fig. 2 (right) presents an example image of the eye after thresholding. The threshold value can be adjusted to the camera and lighting conditions. However, in some cases it must be tuned for an individual observer.

Figure 2: Left: image taken by the infrared camera. Right: the same image after thresholding.

Median filtering

The main task of median filter is to reduce the noise in the thresholded image. After thresholding, some black pixels can still exist in the pupil surrounding. These pixels are filtered out using the median filter. Filtration efficiency, i.e. the efficiency of impulse noise removal, depends on the size of this neighbourhood (see examples in Fig. 3). However, if one chooses too large surrounding the pupil shape can be distorted.

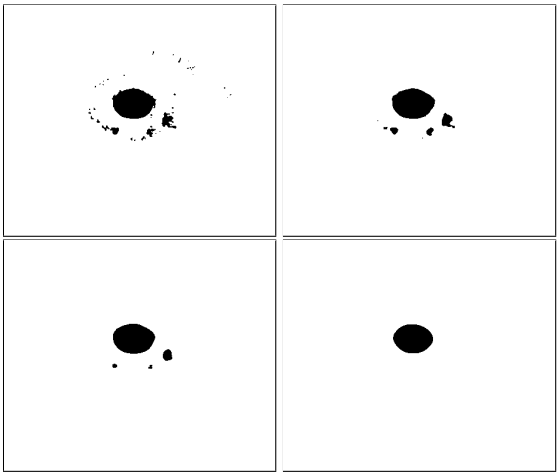

Figure 3: Results of the median filter for 3x3, 7x7, 15x15, and 31x31 pixel neighbourhoods (viewed from the top-left corner). The median filter removes pixels that not belong to the pupil.

The median filter is computationally complex operation for the colour images. It requires sorting of the pixels, which is particularly expensive for the large pixel neighbourhoods. For binary image, this operation is much aster, because the final value of the pixel depends on the number of white pixels in the surrounding. If this number is higher than the number of all pixels in the surrounding, the final value is set to 1 (white colour). Vice versa, the pixel value is set to 0 (black colour).

Cross spread algorithm

The term *cross spread* refers to the process of searching the boundary points. This technique is based on the human eye trait - circular shape of the pupil. Unlike pupils of other mammals, the human pupil is close to the circle. When the image of the eye is captured by camera, the pupil forms various elliptical shapes, depending on the location of the eyeball.

The cross spread technique is performed in the following steps:

1. Choose the starting point located in the area covered by the pupil (black pixels) (Fig. 4a).

2. Shoot 4 rays from the starting point in the horizontal (left, right) and vertical (top, down) directions (Fig. 4b).

3. Find the boundary points located on the pupil edge, that are closest to the starting point (Fig. 4c).

4. Create a new starting point by averaging the coordinates of the boundary points (Fig. 4d).

5. Iterate again from 1. until the location of the starting point is stabilised between iterations.

The boundary points are identified as points not belonging to the pupil, i.e. points with the high gradient between the current and the next position. Abscissa and ordinate of the new starting point are computed by averaging the coordinates of the horizontal and vertical boundary points, respectively.

Parallelisation

We implemented the parallelised version of the algorithm. We apply the regular grid of horizontal and vertical lines that covers the image. Candidate starting points are generated at the intersection these lines. The ones belonging to the pupil are passed to further processing (see Fig. 5a). Then, four rays from each starting point is traced in parallel and the boundary points for each ray are located (see Fig. 5b). The location of the most advanced points in each direction is stored (see Fig. 5c). These values are used to find the pupil centre (see Fig. 5d).

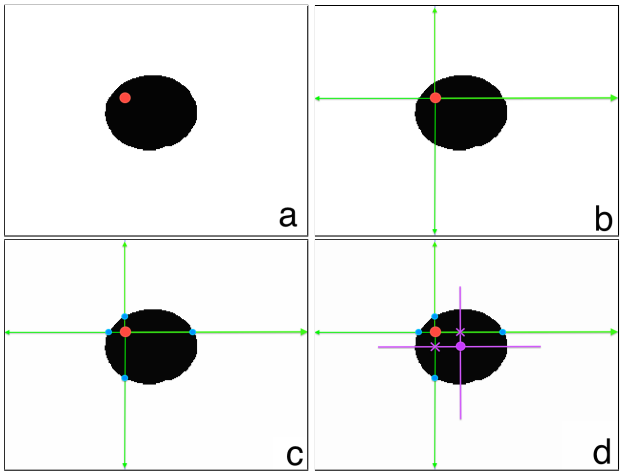

Figure 4: An iteration of the cross spread algorithm.

Corneal reflections

Corneal reflections are the small bright spots in the image of the eye (see Fig. 6). In the most eye tracking systems they are caused by infrared light sources placed near the camera.

If the corneal reflection is located within the pupil it interferes with the trivial implementation of the cross spread algorithm and can cause false detection of the boundary points. Therefore, after detecting a boundary point, we continue the search for the next few pixels. This tolerance depends on the size of the corneal reflection spots and should be set empirically.

3.2 Implementation and performance tests

We implemented the parallelised version of the algorithm using the OpenCL library [1]. This library allows choosing the computing device, which can be both CPU or GPU. We created three kernels responsible for thresholding, filtering, and the core cross spread algorithm. The kernels share the same device memory.

We measured the execution time of the cross spread algorithm using the profiling system provided by the OpenCL platform. It enables to separate time spent for individual kernels and also measures the overall execution time. The test was run 1000 times and then, the results were averaged. The final results are presented in Tab. 1.

Four different computation devices were evaluated: Intel Core i7-2670QM (2.20 GHz), Intel Core i7-3537U (2.00 GHz), NVIDIA GeForce GT 555M, and NVIDIA GeForce GT 740M.

As it was expected, the best results were achieved for GPUs. Both graphics processors can compute the cross spread in less than 2 ms, which is equivalent to processing of more than 500 frames per second. The data preparation phase is rather expensive for GPUs (this is 70% of the

[1] Open Computing Language, `https://www.khronos.org/opencl/`

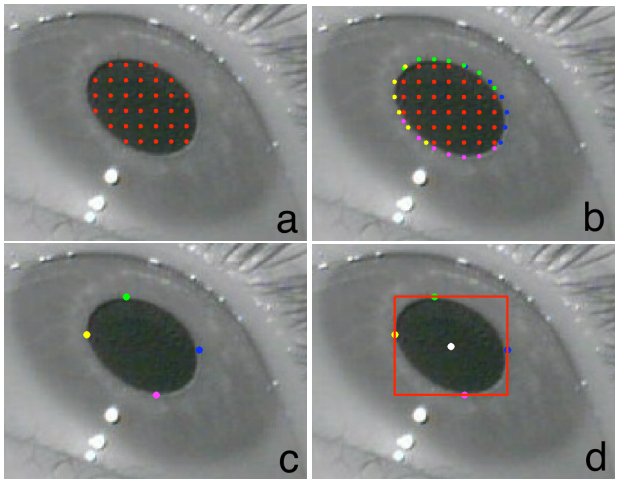

Figure 5: The parallel version of the cross spread algorithm. a: red dots depict the starting points. b: the blue, green, yellow, and pink dots are the boundary points. c: location of the extreme points. d: the white dot depicts the computed pupil centre surrounded by the red rectangle defining the boundaries of the pupil.

Figure 6: The two corneal reflection spots located with the pupil.

total execution time) because the camera image must be transferred to the GPU memory .

Interestingly, we achieved satisfactory results also for CPUs. The execution time of 4.4195 ms and 7.483 ms is equivalent to processing of 225 and 133 frames per second, respectively.

As can be seen in Tab. 1, the median filtering is the main bottleneck of the CPU implementation. It accounts for 60% of the overall execution time.

4 Accuracy evaluation

The goal of the experiment was to evaluate the accuracy of the cross spread algorithm. However, we measured this factor indirectly by testing the accuracy of the DIY eye tracker equipped with our software.

4.1 Do-It-Yourself eye tracker

The DIY eye tracker is a custom-built low-cost eye tracker of a basic construction [4] (see Fig. 7). It consists of two

Computing device	Intel Core i7-3537U 2.00 GHz	Intel Core i7-2670QM 2.20 GHz	NVIDIA GeForce GT 555M	NVIDIA GeForce GT 740M
Thresholding	0.9613	0.5080	0.0944	0.1388
Median filtering	4.4191	2.4243	0.1158	0.1776
Cross spread	0.3036	0.2001	0.2198	0.2128
Overall time	7.4830	4.4195	1.9031	1.8260
Data preparation	1.7989	1.2871	1.4731	1.2968

Table 1: Execution times of the paralleled version of the cross spread algorithm. Time in milliseconds.

main components: a modified safety goggles that act as a frame, and a typical web camera: Microsoft Lifecam VX-1000, working in 640x480 pixels resolution. The only change made to the camera is replacing the infrared light blocking filter with the visible light blocking filter to enable capturing images in the infrared light spectrum. The camera is mounted on the frame in 5 cm distance from the left eye. It is connected to computer via the USB cable. The eye is illuminated by three infrared LEDs placed close to the camera lens.

Figure 7: Head-mounted Do-It-Yourself video eye tracker [4].

Formerly, the DIY eye tracker was controlled by the ITU Gaze Tracker software. We reimplemented the whole eye tracking pipeline replacing all the tasks made by this software with our algorithms. Our implementation involves pupil detection based on the cross spread algorithm, but also communication with the camera, eye tracker calibration, and gaze position estimation.

The principle of the eye tracker operation is based on the observation that the pupil follows the gaze direction during eye movement. Therefore, the location of the pupil centre can be used to estimate the temporary gaze position/direction. The cross spread algorithm detects the pupil centre as the position in camera image coordinates (in pixels). These coordinates must be transformed from the camera space to the screen space to compute the gaze position on the screen. It is done using the mapping defined as the polynomial transformation citeRamanauskas06:

$$\begin{cases} s_x = a_0 + a_1 x + a_2 y + a_3 xy + a_4 x^2 + a_5 y^2, \\ s_y = b_0 + b_1 x + b_2 y + b_3 xy + b_4 x^2 + b_5 y^2, \end{cases} \quad (1)$$

where (s_x, s_y) depicts gaze position in the screen coordinates (in pixels). $a_{0\ldots5}$ and $b_{0\ldots5}$ are coefficient computed during the eye tracker calibration. Calibration is the mandatory part which precedes every eye tracking session. During calibration, people are asked to look at the target points displayed on the screen, so one can assume that locations of these target points are known (i.e. (s_x, s_y) for each target point is known). The pupil centre (x, y) is computed by the cross spread algorithm in the camera space. Then, the polynomial coefficients can be calculated using the Singular Value Decomposition technique (SVD) [3]. During actual eye tracking, the polynomial with known coefficients can be used to transform the centre of pupil location from the camera space to the screen space.

Stimuli and procedure

Observers sat in the front of the display in 60 cm distance and used the chin-rest adopted from an ophthalmic slit lamp. The experiment started with a 9-point calibration. This procedure took about 20 seconds and involved observation of the markers displayed in different areas of the screen. The data processing including computation of the calibration polynomial coefficients was performed by the custom software.

In validation phase, participants looked at the circle marker displayed for 2 seconds at 25 different positions located on the regular grid (see Fig. 9). These positions, called the target points, acted as known and imposed fixation points. The marker was moved between target points in random order. We noticed that smooth animation of the marker between target points allows for faster observer's fixation and reduces number of outliers. Additionally, the marker was minified when reaches its target position to focus observer's attention on a smaller area. The data recorded before 800 ms from the beginning of the marker movement was removed from the analysis. Also the data collected over the last 200 ms were filtered out. Thanks to this it was possible to avoid the errors arising from the gaze transfer between the reference points.

The experiment was performed in a darkened room. Images were displayed on LCD monitor with native resolution of 1920 x 1080 pixels.

Participants

We repeated the experiment for 11 volunteer observers (age between 20 and 22 years, 9 males and 2 females). They declared normal or corrected to normal vision and correct colour vision. The participants were aware what they should do, but they were naïve about the purpose of the experiment.

4.2 Results

The accuracy of eye tracker is quantified as the average distance between the physical target position and the measured gaze position. During experiment described in Sect. 4 we registered over 30 thousand gaze points with the known reference. Due to the eye tracking inaccuracies these points are located in a circular neighbourhood of the physical target points.

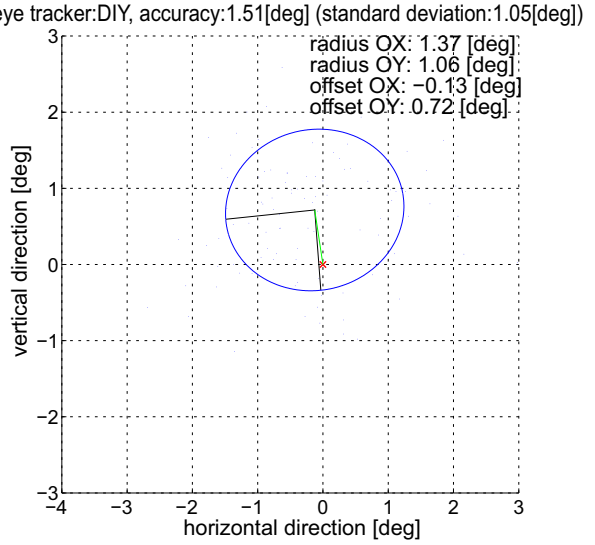

Figure 8: The average DIY eye tracker error.

We present the average eye tracker error as the covariance ellipses (see Fig. 8). The direction of the radii of ellipse corresponds to the eigenvectors of the covariance matrix and their lengths to the square roots of the eigenvalues. An eye tracker will have a good accuracy, if the distribution of the error will have the circular shape corresponding to the normal distribution and the centre of this circle will be located in (0,0) position. Additionally, the ellipse radii should be as small as possible.

As can be seen in Fig. 8, the average eye tracker error is closed to 1.51° of visual angle. The ellipse is noticeably shifted in vertical direction, which indicates the systematic error of the gaze estimation. We suspect that this error was caused by involuntary movements of the head during measurement. The DIY eye tracker is not immune to the head movements. We used the chin rest to stabilise the head but even small movements caused by e.g. swallowing could introduce some inaccuracies.

Fig. 9 shows covariance ellipse calculated for individual target points. The inaccuracies are larger for higher viewing angles. It is particularly evident for the target points located at the edges of the screen, for which the ellipses are distorted. We suspect that it was caused by occlusion of the pupil by the eye lids. Notice, that the distance from the centre of ellipse (i.e. average gaze location) is more meaningful for accuracy estimation than the radii of ellipses. For example, for (-10.1,5.72) [deg] target point lo-

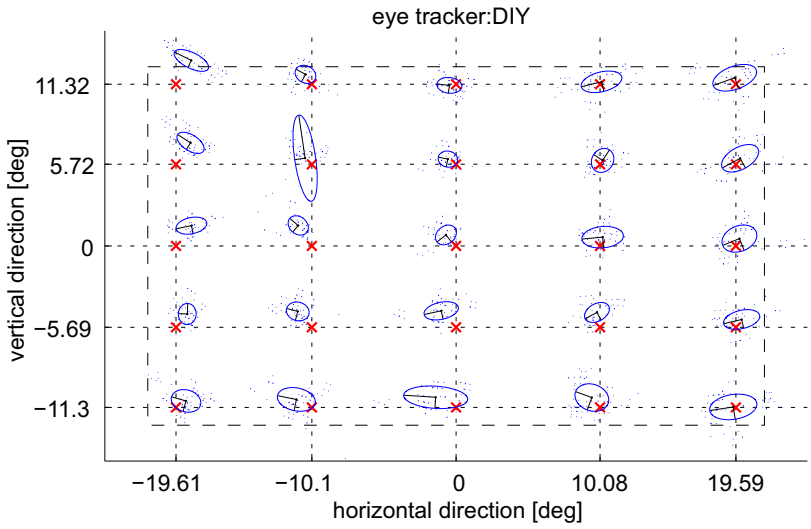

Figure 9: Covariance of the averaged locations of the gaze points recorded by the DIY eye tracker for each reference point.

cation, the distance is rather small despite the large size of the ellipse, which can be caused by some unfiltered gaze points which occurred during blinks.

5 Conclusions and future work

We have implemented a new pupil centre detection technique, which skips some time consuming operations performed by typical methods to increase the processing speed. The core of the technique is the cross spread algorithm, which detects the centre of the pupil by tracing only horizontal and vertical rays. This simple approach gives satisfactory detection accuracy for the pupil images of the centrally symmetric shape. This accuracy is comparable to ITU Gaze Tracker software results.

Our OpenCL-based implementation of the cross spread technique is executed in less than 2 milliseconds, which is equivalent to processing more than 500 frames per second. This frequency is satisfactory for accurate recording of the saccadic movements and only slightly worse than in the commercial high-end eye tracking systems.

We have integrated the cross spread method with the low-cost DIY eye tracker and tested the accuracy of this setup. The perceptual experiments have revealed satisfactory accuracy of the eye tracker equal to 1.5° per visual angle.

In future work we plan to conduct a detailed analysis of the accuracy of the cross spread algorithm. In particular, we are interested in challenging scenarios, in which the pupil is partially obscured.

References

[1] Richard A. Abrams, David E. Meyer, and Sylvan Kornblum. Speed and accuracy of saccadic eye movements: Characteristics of impulse variability in the oculomotor system. *Journal of Experimental Psychology: Human Perception and Performance*, 15(3):529–543, 1989.

[2] Li Dongheng, D. Winfield, and D.J. Parkhurst. Starburst: A robust algorithm for video-based eye tracking. *Proceedings of the IEEE Vision for Human-Computer Interaction Workshop*, September 2005.

[3] Andrew T. Duchowski. *Eye Tracking Methodology: Theory and Practice (2nd edition)*. Springer, London, 2007.

[4] R. Mantiuk, M. Kowalik, A. Nowosielski, and B. Bazyluk. Do-it-yourself eye tracker: Low-cost pupil-based eye tracker for computer graphics applications. *Lecture Notes in Computer Science (Proc. of MMM'12 Conference)*, 7131:115–125, 2012.

[5] A. Pérez, M.L. Córdoba, A. García, R. Ménde, M.L. Muñoz, J.L. Pedraza, and F. Sánchez. A precise eye-gaze detection and tracking system. *Journal of WSCG*, 2003.

[6] Dale Purves, George J. Augustine, David Fitzpatrick, Lawrence C. Katz, Anthony-Samuel LaMantia, James O. McNamara, and S. Mark William. *Types of Eye Movements and Their Functions. Neuroscience (2nd edition)*. Sunderland (MA): Sinauer Associates, 2001.

[7] T. Takegami, T. Goto, and Ooyama. G. An algorithm for model-based stable pupil detection for eye tracking system. *IEICE Trans. Information and Systems vol.J86-D-II(2)*, September 2003.

[8] D. Zhu, T. Raphan, and S. T. Moore. Robust pupil center detection using a curvature algorithm. *Computer Methods and Programs in Biomedicine*, June 1999.