An Adaptive Algorithm for Precise Pupil Boundary Detection Using the Entropy of Contour Gradients☆

Cihan Topal[a], Cuneyt Akinlar[a]

[a]*Department of Computer Engineering, Anadolu University, 26470 Eskisehir, Turkey*

Abstract

We present a robust pupil boundary detection method that is able to detect the pupil even in tough occlusive situations. The algorithm starts by detecting and validating all edge segments in an image with Edge Drawing Parameter Free (EDPF) algorithm. Next, a modal structure estimation for each edge segment is performed by analyzing their gradient vector distributions. In this step, we try to find a near-circular segment that traverses the entire boundary of the pupil by maximizing the entropy of the gradient distributions. Once a near-circular segment is found, elliptical arcs are extracted from only this segment. If no such segment exists, i.e., when the pupil is occluded by the eyelids or eyelashes; arcs from all segments in the image are extracted. Thereby, the algorithm adapts itself and reduces the computation time by trying to understand whether the pupil is entirely visible or occluded. Extracted arcs are then grouped to generate several pupil contour candidates, and one of them is chosen to be the final pupil contour by a cost function. The algorithm is also promising at detection of true negatives which is a useful property for many applications. The algorithm takes a mere 10 ms on non-occluded, and about 30 ms on occluded pupil images size of 640×480.

Keywords: Pupil detection, eye tracking, shape recognition, gaze estimation, human computer interfaces, real-time imaging.

1. Introduction

Eye tracking has emerged as an important research area with a diverse set of applications including human computer interaction, diagnosis of psychological, neurological and ophthalmological individuals, assistive systems for drivers or disabled people, marketing research, and biometrics.

Pupil boundary detection and center estimation is an essential step in many applications. In applications such as eye tracking and gaze estimation, the detection of pupil center is required for the solution. On the other hand, pupil boundary detection is a harder problem and its accurate detection is indispensable for biometrics and medical studies. For any kind of application, the detection of boundary or center has to be performed precisely. For instance, even one pixel precision loss on pupil center may cause an error of a few degrees in the gaze direction vector, which would result in a significant drift in the estimated gaze point.

In this study, we propose a new robust and fast pupil boundary detection method from eye images, which works by extracting arcs from the edge segments of the image and joining them to find the pupil boundary, hence the center. The organization of the paper is as follows; we give a comprehensive related work on pupil boundary and center detection in section 2, we explain our method with fine detail in Sec. 3, analyses of the method in terms of success and running time are presented with experimental results in section 4, and we finalize the paper with concluding remarks in section 5.

2. Related Work

The literature on pupil detection is very rich, and many different techniques have been proposed. In this section, our goal is to give a high-level picture of the proposed solutions for pupil boundary detection and/or pupil center estimation.

One of the most common pupil detection techniques is dark-/bright pupil differencing [1]-[3]. In general, this technique is useful to roughly detect the eye locations in a remotely taken image. First, two eye images are captured with on-axis and off-axis illumination. Due to the physical structure of human pupil, on-axis illumination causes a significant brightness inside the pupil. Then difference image of two successive frames is computed and thresholded. Morphological operations are usually performed on the binarized image to visually enhance the result.

Wang and Sung [5] detect iris instead of the pupil in their gaze estimation study. They threshold the eye image and apply morphological opening to improve compactness of the pixel sets. Next, they find vertical edges, select two longest edge segments and fit ellipse. Goñi et al. [6] adaptively threshold the eye image and apply morphological open/close operations. Then connected component analysis is applied to detect the blobs. Later, blobs are filtered according to their sizes, and the most compact blob is selected as the pupil. Finally, center of the

☆This work is supported by The Scientific and Technological Research Council of Turkey (TUBITAK) and Anadolu University Commission of Scientific Research Projects (BAP) under the grant numbers 111E053 and 1207F113, respectively.

Email addresses: cihant@andolu.edu.tr (Cihan Topal), cakinlar@andolu.edu.tr (Cuneyt Akinlar)

Table 1: A brief taxonomy for pupil boundary detection / center estimation algorithms.

REFERENCES	Downsampling	Bright / Dark Pupil Differencing	Image Thresholding / Binarization	Morphological Operations	Edge Detection	Blob Detection / Con. Comp. Ana.	Circle / Ellipse Fitting	Center of Mass Algorithm	Use of Temporal Information	Other / Comments
Ebisawa [1]		•	•	•				•		⌀
Morimoto et al. [2]		•	•						•	⌀
Hiley et al. [3]		•	•	iterative				•		⌀
Wang and Sun [5]			•	•	vertical		ellipse			♢
Gõni et al. [6]			adaptive	•				modified	•	⌀
Keil et al. [7]			•		for glint			•		⌀, ♤
Agustin et al. [8]			•				ellipse			
Long et al. [9]			•			•		symmetric		⌀
Lin et al. [10]	•		•	•	•			parallelogram		⌀
Ma et al. [11]			adaptive		•		Hough			
Mäenpää [13]			•	•				•		♢
Dey and Samanta [14]	•				•	for edges	circle			♤
Zhu et al. [15]			•			•	ellipse			△
Li et al. [16]					radial		ellipse			♧
Kumar et al. [19]				•	•	for edges				□

⌀ does not detect the pupil boundary, only estimates its center. ♢ performs iris detection.
♤ applies histogram back-projection or non-linear power transform on the image to make the pupil more salient.
△ before ellipse fitting, tries to determine the false pupil contour pixels with respect to their curvature values by a set of assumptions.
♧ requires removal of glints. Assumes that the initial point for ray casting is inside the pupil. Iterative algorithm.
□ performs Fast Radial Symmetry detection and Delaunay Triangulation. Removes unwanted artifacts such as glints by a set of morphological rules and assumptions.

pupil is calculated with a modified center of mass algorithm. Keil et al. [7] first compute edge map and look for glints. The pupil is then searched nearby the glints. Dark pixels composing the pupil are extracted with a histogram back-projection, and the centroid of these pixels is considered to be the pupil center. Agustin et al. [8] threshold the image and extract the points between pupil and iris. Then they fit ellipse to these points with RANSAC [17].

The occlusions caused by eyelids and eyelashes makes detection of pupil a lot more difficult and there are studies that specifically target this problem. Long et al. [9] capture only the eye vicinity to save time and threshold the acquired images with a user-set value. Then, blob detection is applied and the largest blob is selected as the pupil. Pupil center is computed using a symmetric mass center algorithm which is based on placing a parallelogram into the pupil contour with the assumption that the centroids of the pupil ellipse and the parallelogram will overlap. In a similar vein, Lin et al. [10] uses inscribed parallelograms to make a good estimate of the pupil center in occlusive cases. The image is thresholded and undesirable artifacts are removed by morphological operations. Next, pupil contour is extracted and three parallelograms are inscribed into it. If the centroids of these parallelograms are not overlapped, the pupil contour is rotated and the previous step is repeated.

Ma et al. [11] add up the intensity values of rows and columns, and take the lowest $[row, column]$ value as approximate pupil center in their iris recognition study. Then, adaptive thresholding is performed and a local image region is determined according to the approximated pupil center. Finally, edge detection followed by Hough circle transform [12] are applied.

Mäenpää [13] finds the pupil centre by thresholding and morphological opening in their iterative iris detection algorithm. Dey and Samanta [14] downsample image to reduce the computation time, and apply a non-linear power transform to increase the contrast of the pupil. Then Gaussian smoothing followed by edge detection are performed. Next, edge segments are extracted and bounding box of each edge segment is found as a modal heuristic. Finally, circle fitting is applied to each segment to determine the pupil center.

There are also methods using alternative ways in feature extraction for pupil detection. In [15] authors propose using curvature of pupil contour to detect pupil boundary. After thresholding and blob detection are applied, small blobs are eliminated by a pre-defined threshold. Next, contour of pupil's blob is extracted and its curvature is calculated. Finally, edge pixels of the pupil's contour are then classified by employing a set of rules (e.g. eyelids have positive curvature, etc.) and ellipse fit applied to the selected pixels.

Another study, Starburst, estimates pupil center by an iterative radial feature detection technique instead of finding all edges. The algorithm starts by locating and removing the glint. Then, rays are cast from an initial point (assumed to be inside the pupil) with a 20° of radial step. Each ray stops when the derivation is greater than a threshold value, i.e., when a sharp intensity change occurs. This operation is iterated a few times with an updated starting point and a group of feature points are collected at each step. Finally, an ellipse is fit to the collected feature points with RANSAC [17]. Although the algorithm dif-

fers by its feature extraction scheme, it is not clear how it performs in occlusive cases as all the experiments presented in the paper consist of images with the pupil in clear sight. Authors also present a study that tries to adapt the Starburst algorithm to elliptical iris segmentation problem [18].

A different approach on pupil detection is used in EyeSeeCam project [19]. The algorithm starts with a fast radial symmetry detection algorithm to obtain a rough estimate of pupil. Then edge detection followed by connected component analysis is applied to obtain edge segments. Thereafter, unwanted artifacts like glints and other reflections are deleted by applying a sequence of morphological and logical operations. Finally, Delaunay Triangulation is applied to remaining pixels and the pupil's boundary is detected assuming it is a convex hull. Although results of the algorithm are successful in occlusive cases, it has a complicated structure consisting of assumptions and morphological rules. Furthermore, there is no concrete information about the computation time requirements of the algorithm, except the authors' claim that all employed tools are computationally inexpensive.

In addition to above algorithms, there are also purely model-based methods which are mostly utilized in iris recognition studies. As an example, Daugman [20] proposes an integro-differential operator for detecting the pupil and iris boundaries aiming to maximize the contour integral value on the smoothed image derivative (Eq. 1).

$$\max_{(r,x_0,y_0)} \left| G_\sigma(r) * \frac{\partial}{\partial r} \oint_{(r,x_0,y_0)} \frac{I(x,y)}{2\pi r} ds \right| \qquad (1)$$

Arvacheh and Tizhoosh [21] developed an iterative algorithm based on an active counter model which is also capable of detecting near-circular shapes such as the iris and the pupil. These methods work fine; however, they require a full search of the image plane in order to find r, x_0, y_0 parameters that maximize the given model. This search operation is computationally expensive and highly vulnerable in cases where the pupil is not perfectly circular. Therefore, they cannot be employed in real-time eye tracking applications.

Table 1 gives a brief taxonomy of the above-mentioned pupil detection algorithms. Before we go over the table and describe relative merits or disadvantages of the employed methods, it is important to remind that even small errors in pupil center or boundary detection can cause significant inefficiencies in the medical or gaze applications. Therefore, downsampling the image to save computational time is obviously undesirable as it decreases the accuracy by wasting fiducial information. Next technique on the table is the bright/dark pupil differencing. Although this technique takes little computation and eases roughly locating the pupil, it has important disadvantages. Firstly, it needs additional hardware to obtain bright and dark pupil images consecutively in a synchronous manner. Secondly, it reduces the temporal resolution in half since it needs two frames to perform single pupil detection. Thirdly, it is very sensitive to motion; that is, if a large pupil displacement occurs between two consecutive pupil images, it becomes difficult to find the pupil. Ebisawa specifically addresses this problem in [4], and proposes various methods for positional compensation.

As seen from Table 1, thresholding is the most common technique in pupil detection. Despite thresholding can quickly discriminate image regions with different intensity values, it is highly vulnerable to lighting conditions and parameter selection. Furthermore, it fails on finding the exact location where intensity change occurs which decreases the efficiency and accuracy. Another common technique employed after image thresholding is morphological operations, i.e., erosion, dilation, opening, etc. These operations are applied on the thresholded binary image to improve modal structure of the image and eradicate the meaningless pixel formations. However, they may also affect the actual information on the image and cause significant errors on the result. Moreover, methods which utilize bright/dark pupil, thresholding and blob detection to find only a center point for pupil are obviously not capable of detecting the boundary. Hence, they cannot be applied on most biometrics or medical studies which requires precise detection of the boundaries of pupil and iris.

In this section we presented an overview of related studies covering both biometrics and eye tracking areas from the viewpoint of pupil detection problem. For interested readers, there are also comprehensive surveys that review the gaze estimation literature; in particular [22, 23].

3. Proposed Method

In this study, we propose an adaptive method for pupil boundary detection which is able to save computation by inferring whether an occlusion is the case or not. When the pupil is in clear sight, the computation takes very little time; when the pupil is severely occluded, the algorithm infers it and spends more effort to detect the pupil without compromising real-time applicability. The main strategy improving the method against the occlusive cases is extracting the elliptical arcs from the image and finding one arc or a group of arcs representing the pupil contour. In this way, relevant features from a partially visible pupil can be extracted and detection can be performed by fusion of separate features. Besides detecting the pupil boundary and center precisely, the algorithm can also identify if there is no pupil in the image as a very handy information for many applications.

The proposed method has a simple flow and consists of the processing pipeline shown in Fig. 1. The processing starts by the detection of the edge segments of the image by recently-proposed edge segment detector, the Edge Drawing (ED) algorithm. ED returns a set of edge segments, each of which is a contiguous chain of pixels. Having edge segments instead of a binary edge map makes further higher-level processing easier.

After the edge segments are extracted, the next step is to determine whether a near-circular segment exists that traces the entire boundary of the pupil. Such an edge segment would be found if the pupil is clearly visible with no or very little occlusion. To find whether an edge segment has a circular geometry, we devise a fast heuristic method which is based on the gradient distribution of the segments. On the condition that a near-circular segment is found, elliptical arcs are extracted only from that segment. If no near-circular segment is found, which

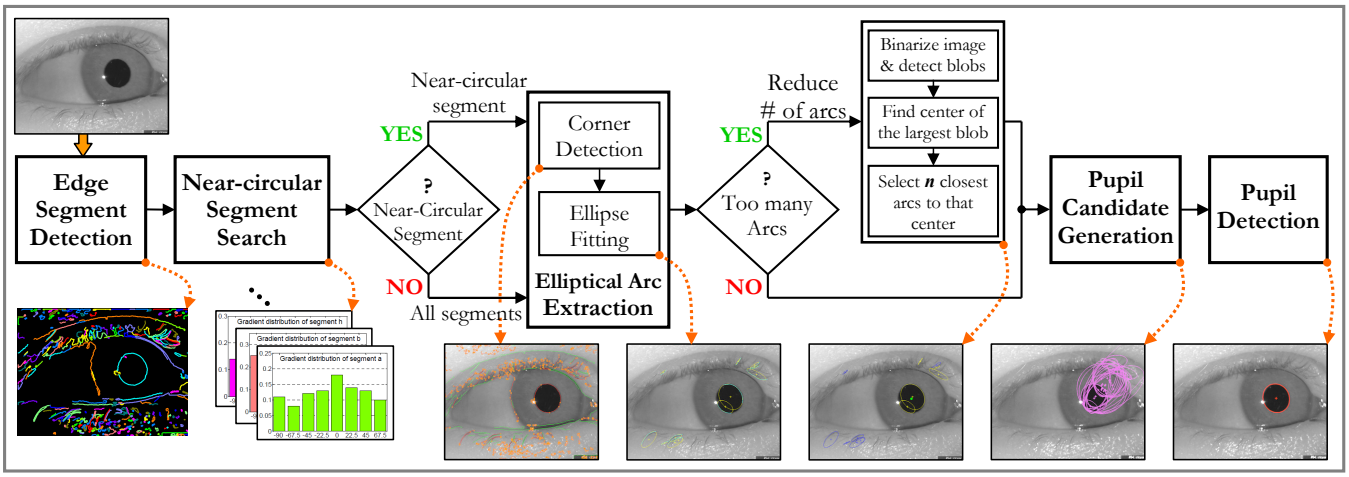


Figure 1: Processing pipeline of the proposed algorithm.

would be the case if the pupil is severely occluded by the eyelids or eyelashes, then arcs from all edge segments in the entire image are extracted.

Following the extraction of the elliptical arcs, the next step is to join the arcs to generate a set of ellipse candidates that trace the pupil boundary. Candidates are finally evaluated for their relevance to be the actual pupil contour and the best one, if it exists, from among the candidate ellipses is chosen.

In the following subsections, our goal is to elaborate each step of the proposed algorithm in detail, giving enough detail to make the discussion clear.

3.1. Edge Segment Detection

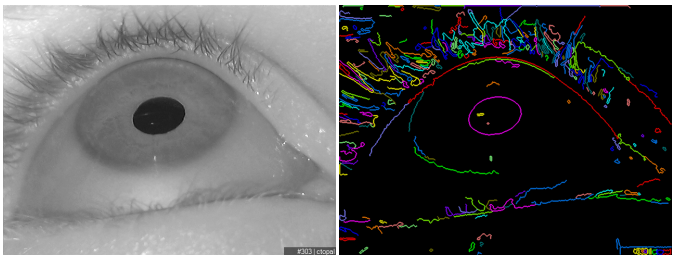
To detect all edge segments of an image, we employ our recently proposed, real-time edge/edge segment detector, Edge Drawing (ED) [24]-[27]. Unlike traditional edge detectors which work by identifying a set of potential edge pixels in an image and eliminating non-edge pixels through operations such as morphological operations, non-maximal suppression, hysteresis thresholding etc., ED follows a proactive approach. The ED algorithm works by first identifying a set of points in the image, called the anchors, and then joins these anchors using a smart routing procedure; that is, ED literally draws edges in an image. ED outputs not only a binary edge map similar to those output by traditional edge detectors, but it also outputs the result as a set of edge segments each of which is a contiguous and connected pixel chain [26]. This property of ED extremely eases the application of the algorithm to further detection and recognition problems.

ED has many parameters that must be set by the user, which requires the tuning of ED's parameters for different types of images. Ideally, one would want to have a real-time edge/edge segment detector which runs with a fixed set of internal parameters for all types of images and requires no parameter tuning. To achieve this goal, we have recently incorporated ED with the a contrario edge validation mechanism due to the Helmholtz principle [30, 31], and obtained a real-time parameter-free edge segment detector, which we name edge drawing parameter free (EDPF) [28, 29]. EDPF works by running ED with all ED's parameters at their extremes, which detects all possible edge segments in a given image with many false positives. We then validate the extracted edge segments by the Helmholtz principle, which eliminates false detections leaving only perceptually meaningful edge segments with respect to the a contrario approach with very little overhead of computation. Fig. 2 illustrates detected edge segments for an example eye image. In the figure, each color represents a different edge segment, which is one-pixel wide, contiguous chain of pixels.

3.2. Near-Circular Segment Search

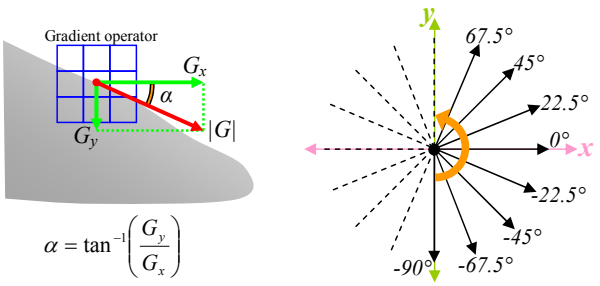
The main goal of this step is detecting the pupil in an easy and computation efficient way when it is entirely visible without any occlusions. Once we have the edge segments detected, we need to find the one that traverses the pupil boundary. The first thing that comes to mind is to apply a brute force search as follows: fit an ellipse to each edge segment, compute the ellipse fitting error, i.e., root mean square error, and pick the edge segment that has the smallest fitting error. Since the pupil has a near-circular geometry, its segment would result in the smallest error. This method might work only when the pupil is clearly visible, i.e., when the pupil is not occluded by IR-LED glints or eyelashes, however, fitting an ellipse and calculating the fitting error for each segment requires a lot of computational time.

To reduce this computational burden, we devise a faster method based on the analysis of gradient directions to find the near-circular segment, if one exists. Gradient information of the segments contains substantial information about the geometrical structure and is used in shape matching, retrieval and recognition problems [32]-[36]. Since we already have the vertical and horizontal derivatives of the image computed during the edge detection scheme, we can find the gradient directions with simple trigonometry in a computationally cheap way (see Fig. 3.a). The *arctan* function obviously results angle values in an interval $[-\pi/2, \pi/2]$, providing an angle range of 180°. Before examining the distribution of gradients, we quantize the angles with 22.5°, thus we divide the unit circle into 16 regions into 8 different directions (see Fig. 3.b).



(a) (b)

Figure 2: **(a)** Sample eye image, **(b)** 255 edge segments obtained by the EDPF algorithm. Each color represents a different edge segment.



(a) (b)

Figure 3: (a) Gradient computation with image derivatives, (b) Quantization of computed gradient directions.

Once we get the quantized gradient directions for all pixels in each segment, we try to infer the modal characteristics of each segment by analyzing their gradient distributions. It is easy to observe that any segment in near-circular form would have an even gradient distribution if the tangential gradients on its perimeter is sampled with a fixed angular step.

Fig. 4 shows the edge segments of an eye image, and illustrates the distributions of the gradient directions for the 9 edge segments labelled a-i. As seen from the figure, circular edge segments have an even distribution; whereas, straight edge segments have an uneven distribution, where a few values dominate.

Thus, to distinguish the segments having circular shapes, we need to select the segments that result in plain gradient distribution. To achieve this, we use the entropy function (Eq. 2) on the quantized gradient distributions of the segments.

$$E = -\sum_{i}^{n} p_i . \log(p_i) \tag{2}$$

Since entropy function maximizes for flat distributions where the frequency of each symbol is equal, we compute the entropy values of the gradient distributions of the segments and select the segments which maximize the entropy (Eq. 3).

$$\arg\max \left| \sum_{i}^{8} f_{G_i} . \log(f_{G_i}) \right| \tag{3}$$

where f_{G_i} is the frequency of the i^{th} gradient direction. The entropy values for the segments are maximized for a perfect circle and gets lower as the segment shape differs from being a circle (elliptic, etc.), and finally entropy value becomes zero for straight lines since a straight line has only one gradient direction along its geometry. Since the number of different symbols is 8; the maximum entropy value is $\log_2 8 = 3$ in our case. In Table 2, the lengths and the entropy values of the segments shown in Fig. 4 are listed. It is easy to observe that circular edge segments have higher gradient entropy values regardless their size, whereas straight edge segments have lower values as expected.

With this heuristic, we can discard the segments producing small entropy values than a certain threshold in a fast manner. When examining speed of the heuristic, we measure that computing the gradient entropy of an edge segment with available image derivatives is 150~200 times faster than ellipse fitting and error computation with respect to length of the segment. In this way, we avoid spending computation time on the segments which have irrelevant geometries.

Following the computation of the edge segment entropies, one of the segments is chosen to be the near-circular segment if it satisfies the following three conditions:

i) Must have a very high gradient entropy. Note that the theoretical upper-bound for the entropy is $\log_2 8 = 3$ for 8 different gradient directions. We choose the segments that have 2.9 or more gradient entropy values.

ii) Must have a small root mean square error, e.g., 2 pixels, when an ellipse is fit to the pixels of the segment.

iii) Must be a closed segment. In other words, the distance between the start and end points of the segment must be short. To avoid problems due to the small occlusions, such as glints, we consider a 15 pixels threshold for this distance.

Evaluation of the second condition requires ellipse fitting to a set of points, for which we employ the method proposed by Fitzgibbon et al. [37]. To compute the ellipse fitting error, there is no straightforward method; so, we developed a method based on [38]. We compute the distance between the ellipse and each point by solving the equations described in [38] with Newton-Raphson estimation method in a couple of iterations. Once we estimate all distance values between each point and the ellipse, we calculate the normalized rms distance to obtain a single scalar for the fitting error. For ellipse perimeter calculation, which also does not have an exact solution, we employ Ramanujan's second approximation [39]. In the event that more than one edge segment satisfies all three conditions given above, the one having the minimum ellipse fitting error is chosen to be the near-circular segment. While existence of a near-circular segment speeds up the computation, its not compulsory for the detection of pupil.

It is important to note that shape of a segment does not necessarily have to be near-circular to give high entropy values. In addition to segments with near-circular geometry, segments which have concave shapes or follow complex trajectories can also produce high entropy values. Therefore, we use entropy

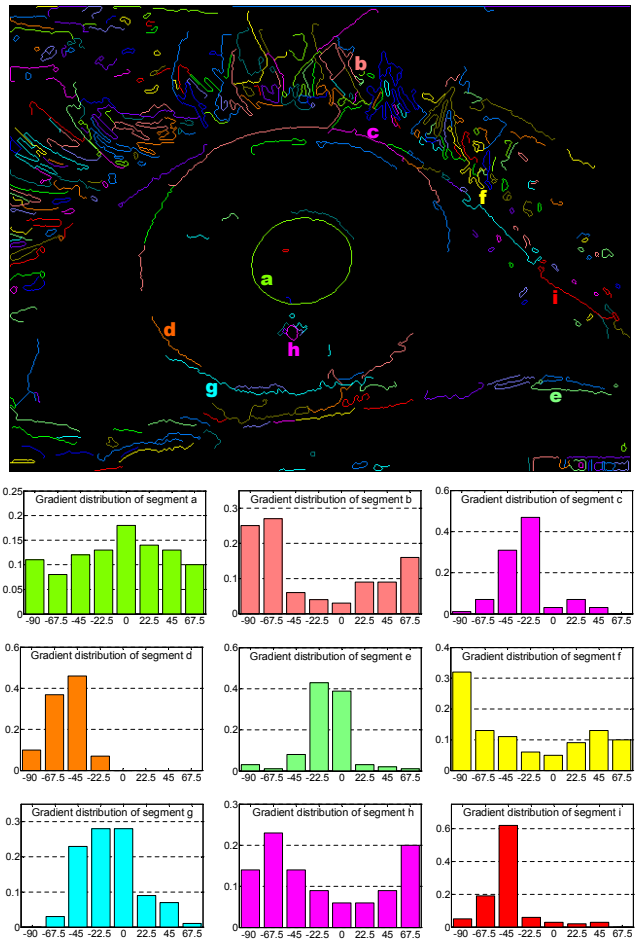


Figure 4: Detected edge segments displayed with different colors (top), Gradient distributions of the labeled segments in the image (a to i).

test as a prerequisite to accelerate the algorithm and make final decision about a segment by ellipse fitting.

3.3. Elliptical Arc Extraction

The next step of the algorithm is extracting the elliptical arcs (which will be referred to as *arc* hereafter) gathered in the edge segment detection. If a near-circular segment could be found at the previous step, arcs are only extracted from that segment. If no near-circular segment is found, then all segments are subjected to arc extraction process. In this manner, the algorithm adapts itself and requires less computation when the pupil contour is entirely visible.

In a previous work, we extract circular arcs by combining consecutive line segments in an image to detect circles [40, 41]. However, pupil's projection onto the camera plane can be in a more elliptical structure, hence we need to detect the elliptical arcs in this study. To solve this problem, we devise another strategy which also works fast. Finding the start and end points of a potential elliptical arc inside an edge segment is achieved by locating the corners along the segment. First, we detect corners on the segments with a fast curvature scale-space (CSS) method [42]. To gain speed, we used the same gradient information employed in previous steps to compute the turning angle curvature and detect corners. Afterwards, we fit an ellipse to the points lying in between two consecutive corners along each segment and obtain elliptical arcs.

Table 2: Lengths and gradient entropies of the segments shown in Fig. 4.

Segment ID.	Length (pix.)	Entropy
a	271	2.96
b	128	2.66
c	86	1.99
d	68	1.66
e	100	1.89
f	127	2.76
g	232	2.36
h	35	2.83
i	113	1.74

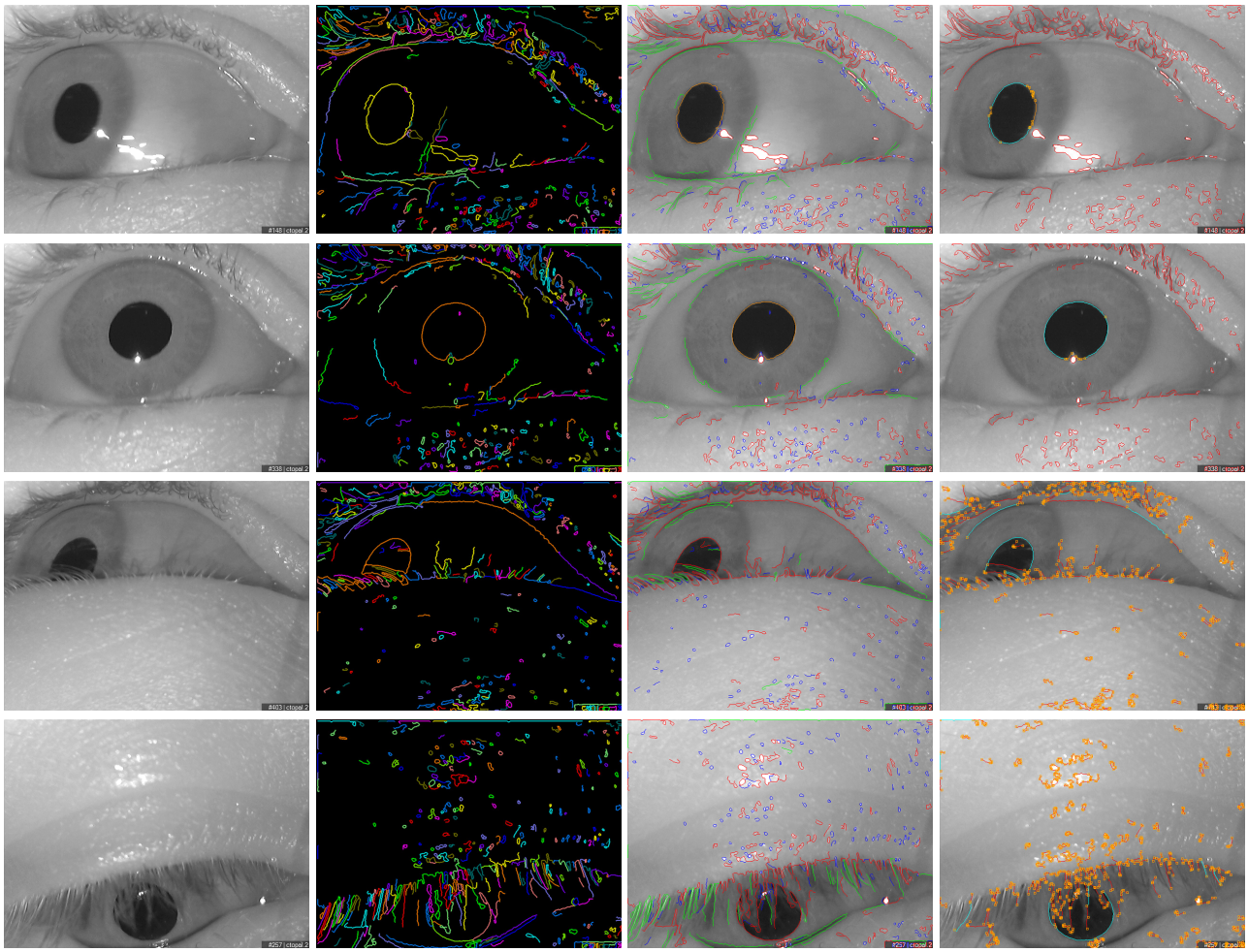
Fig. 5.c and Fig. 5.d presents the results of our arc extraction method for 4 test images. In the first two rows, the pupil is completely visible; hence, the near-circular segment (the orange colored segment) is detected. Therefore, arc extraction is applied only to this segment. When no near-circular segment is found, i.e., when the pupil is severely occluded by the eyelids or eyelashes, arcs are extracted from all segments to avoid missing any useful information (see 3^{rd} and 4^{th} rows of Fig. 5). To save further computation time, we omit the segments having low gradient entropy (i.e., < 2) because their straight geometry rarely contains elliptical arcs. We also neglect the short segments (i.e., < 25 pixels) since they do not provide any useful information. In the third column of Fig. 5, low gradient segments and short segments are indicated in green and blue, respectively, and the final segments that are subjected to arc extraction displayed in red. The last column of Fig. 5 shows the detected corners (orange colored squares) and extracted arcs (colored cyan) for all images.

It should be noted that even if the pupil is in clear sight, it may appear highly elliptical because of the view angle. In such a case, pupil's segment may result in a low gradient entropy and near-circular segment would not be detected. As a consequence, elliptical arc extraction would be applied to all segments even though there is no occlusion.

3.4. Generation of the Pupil Candidates

In this step, we generate candidate ellipses by grouping the extracted arcs. Thus we aim to detect the complete pupil boundary even if its boundary is partially visible. To generate pupil candidates, first we consider all subsets of extracted arcs. Excluding the empty set, there are $2^n - 1$ different arc combinations for n arcs. Due to the exponential grow rate, the number of generated pupil candidates increases excessively for the number of extracted arcs and testing all combinations becomes infeasible. In Fig. 6 an example is shown to illustrate the relation between number of selected arcs and generated pupil candidates. As seen from the example, the number of generated pupil candidates increases in a useless manner for higher number of selected arcs.

For the cases where the number of extracted arcs exceeds, we apply a fast selection method to reduce the number of arcs.

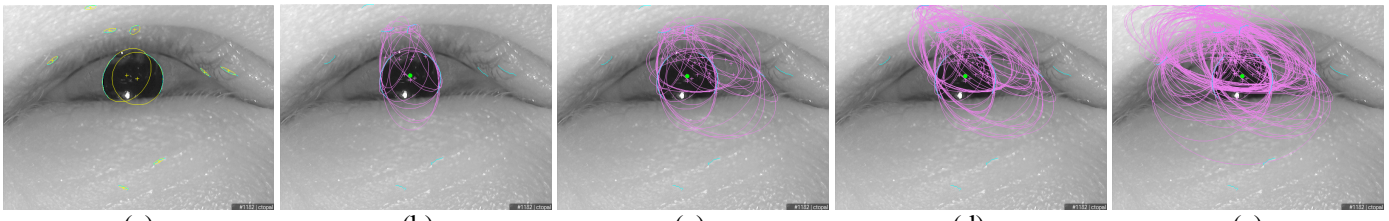


(a) (b) (c) (d)

Figure 5: Arc extraction process. First two rows include examples where the pupil is clearly visible; whereas, in the last two rows the pupil is severely occluded. (a) input images, (b) detected edge segments displayed in different colors, (c) short segments, low gradient entropy segments and segments from which arcs will be extracted are colored in blue, green and red, respectively. The near-circular segment displayed in orange, if it exists, (d) corners (orange boxes) and extracted arcs (cyan segments). Note that if the near-circular segment is found, corner and arc detection is only applied to that segment.

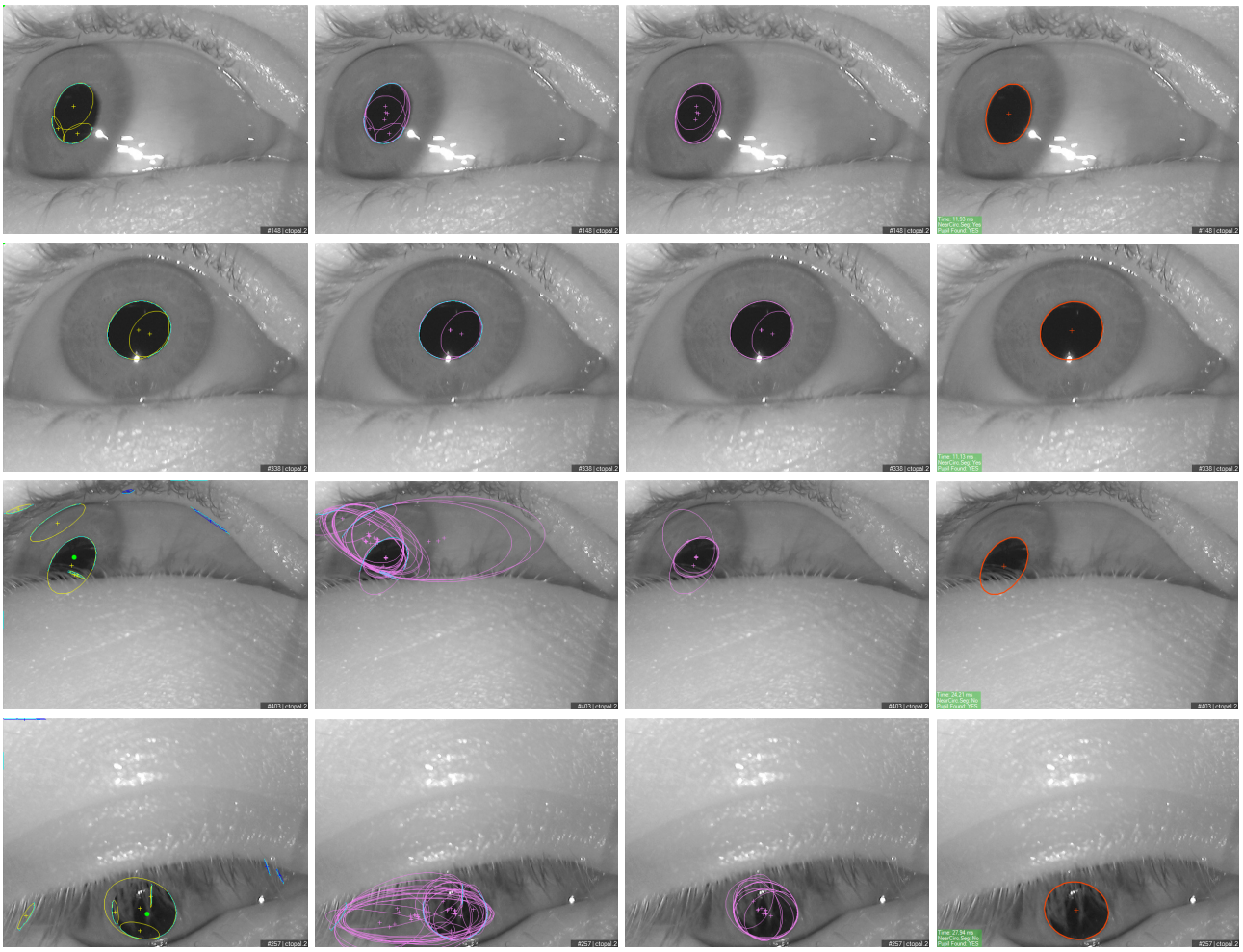
Similar to many pupil center estimation methods, we simply binarize the image and find the centroid of the largest blob as a very rough estimation of pupil center. Then, we select a predefined number of closest arcs to that center.

In the experiments, we see that the pupil counter is detected as a couple of separated arcs even in tough occlusions, hence we select 5 arcs to guarantee reconstruction of the pupil boundary. If the number of extracted arcs is below 5, we do not perform arc selection which is the common case if there is no occlusion. In the last two rows of Fig. 7.a, selected and eliminated arcs are indicated with yellow and blue ellipses, respectively. The arcs displayed with yellow ellipses are selected with respect to the distance between their center and rough pupil center estimation (indicated with green circle). Please note that any arc elimina-



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

Figure 6: Generated pupil candidates for different number of arc selections. (a) Eye image with 10 elliptical arcs detected. (b)-(e) Generated 15, 31, 63, 127 pupil candidates for 4, 5, 6, 7 selected arcs, respectively. Remember that we generate $2^n - 1$ ellipses for each pupil candidate for n selected arcs.



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

Figure 7: Pupil candidate generation and pupil detection steps. First two rows include examples where the pupil has clear sight; whereas, last two rows present occlusive cases. (a) Detected pupil blob center is indicated with green spot. Ellipses show fitting results of arcs and displayed in yellow and blue for selected and eliminated ones, respectively. If the number of arcs exceeds a specific value (5 in the example), arcs selected with respect to the distance between the ellipse center and the pupil blob center. (b) $2^n - 1$ pupil candidates are generated by joining n arcs in different combinations, (c) remaining pupil candidates after elimination due to high fitting error, (d) the selected ellipse representing the pupil contour using the argument in Eq. 4.

tion is not applied to the first two rows due to the small number of (i.e., ≤ 5) extracted arcs.

Once we obtain arcs, we analyze all $2^n - 1$ arc subsets by fitting an ellipse to all arc pixels in each subset. Fig. 7.b shows all generated pupil candidates for the input images in Fig. 5.a. Since the pupil candidate generation process considers all subsets of selected arcs, groups of unrelated arcs which do not form a valid conic geometry are also subjected to ellipse fit operation. Therefore, the next step is to eliminate those candidates having a high fitting error, which clearly cannot be the pupil boundary segment. Fig. 7.c shows the remaining ellipse candidates after this elimination. The real pupil boundary segment will be chosen from among these remaining candidates.

3.5. Detection of the Pupil

In the previous step, we get a number of pupil candidates each of which consists of one subset of elliptical arcs. Accordingly, we still need to select one candidate ellipse to be the final pupil contour. To make the decision, we define a cost function which considers the following properties of a candidate ellipse:

i. the ellipse fitting error (ε),

ii. the eccentricity (τ),

iii. the ratio of the arc pixels to the perimeter of resulting ellipse (ϕ).

Each of the pupil candidates is formed by one or more arcs. If the pupil boundary is detected with multiple arcs, the fitting error should be reasonable because we expect the arcs to be parts of the same contour. Thus we need to minimize the fitting error ε.

The eccentricity (τ) indicates the compactness of an ellipse and is the ratio of the semi-major and semi-minor axes. For a circle, it is obviously 1. We obtain diverse pupil candidates whose eccentricities can vary, i.e. $0.1 \sim 10$. However, pupil's projection onto the image plane is usually closer to a circle

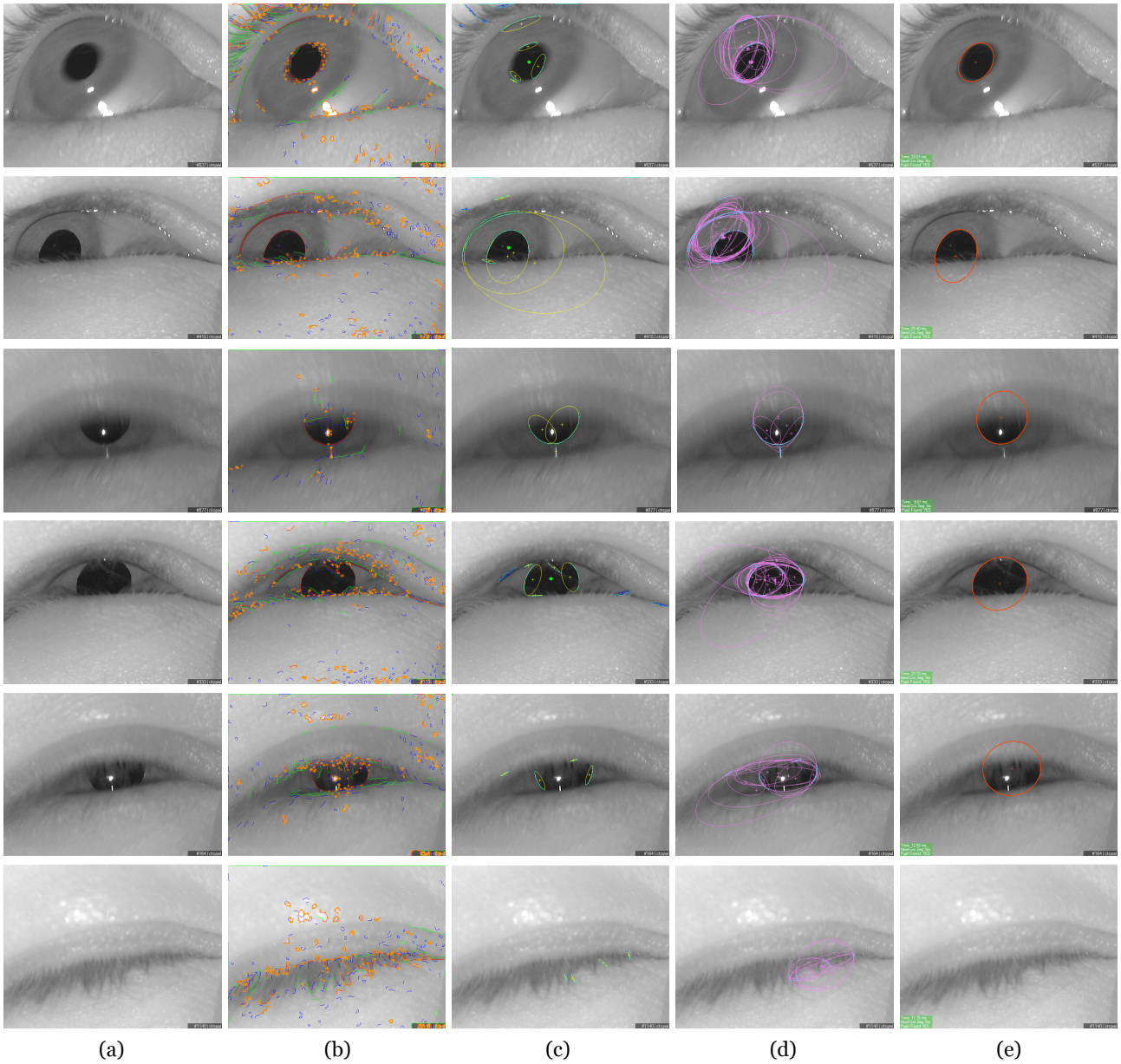


Figure 8: A set of experimental results where the algorithm succeeds. (a) input image, (b) edge segments and corners, (c) extracted arcs pixels (cyan), selected arcs (yellow), eliminated arcs (blue) and pupil blob center (green), (d) generated pupil candidates, (e) the selected pupil candidate. Frame IDs and other related information are printed at the bottom of the images.

rather than a skewed ellipse in related applications. Therefore, we tend to select a candidate having an eccentricity close to 1.

The parameter ϕ is the ratio of the number of pixels involved in ellipse fitting to the perimeter of the resulting ellipse. In some circumstances, one single and short arc may result in a large pupil candidate ellipse that may lead to inconsistency. Therefore, we look for the pupil candidates which are formed by consensus of more arc pixels and have larger ϕ.

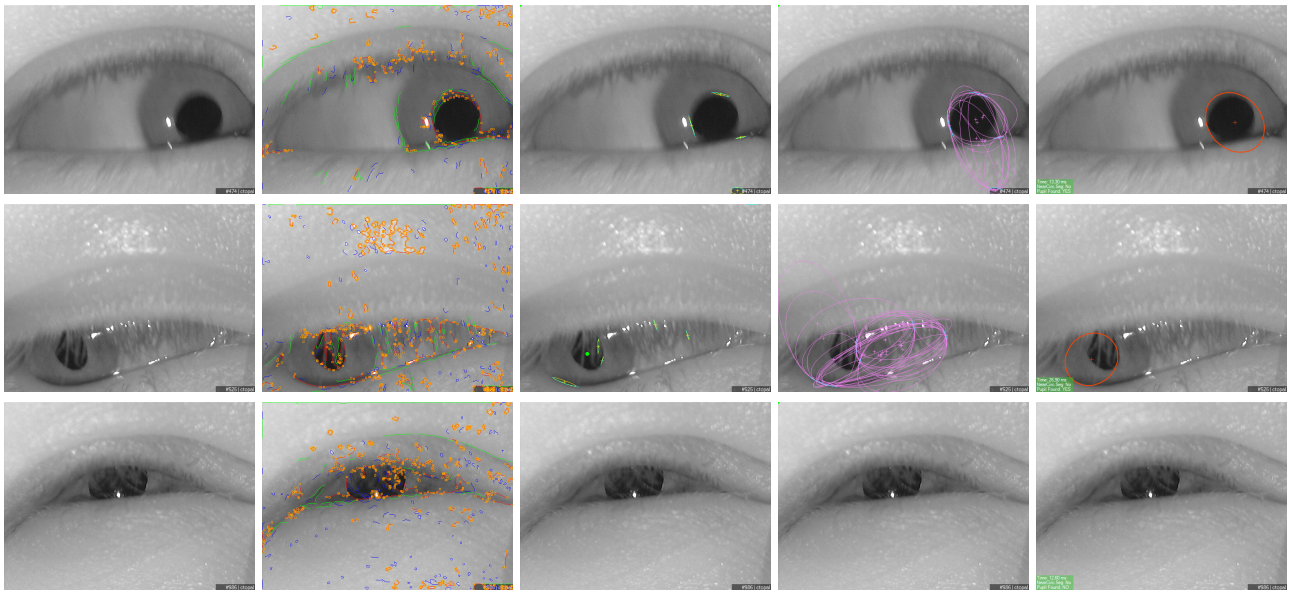
During our experiments, we observed that the effect of the eccentricity (τ) is less than the effect of ε and ϕ due to the possibility of true pupil not being the most compact ellipse among the candidates. Accordingly, we take the squares of ε and ϕ to increase their effect on the cost function. Finally, we need to minimize ε and τ and maximize ϕ in our formulation, and select the candidate that minimizes the following argument:

$$\arg\min_{(\varepsilon,\tau,\phi)} \left| \frac{\varepsilon^2 \times e^{|1-\tau|}}{\phi^2} \right| \tag{4}$$

Fig. 7.d. shows the pupil detection results. Among the pupil candidates shown in Fig. 7.c, the one that minimizes cost in Eq. 4 is selected as the pupil.

3.6. Detection of True Negatives

In many applications, having the information that there is no pupil in the image is important as much as detecting the pupil. This information can provide very useful extensions to the eye



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

Figure 9: A set of experimental results where the algorithm fails. Please refer to section 4.1 for the explanation of the causes of failures.

tracking applications such as blink detection. It is possible to obtain arcs and pupil candidates although there is no pupil with the algorithm (see last row of Fig. 8). We observe that the cost function overshoots in these circumstances due to large ε and small ϕ values. Therefore, we can easily find the lack of the pupil by examining the functions output. In Fig. 10 we present a plot of the cost function versus a number of frames sampled from an eye blink operation. It is seen that the function's output increases as the visible part of the pupil is getting smaller. Notice that the plot is in logarithmic scale.

4. Experimental Results

In this section, we present results for a comprehensive set of experiments. First, we give pupil detection examples with challenging occlusive cases followed by cases where the algorithm fails. Then we analyse the running time of the algorithm by giving details for each step in both easy and tough cases. Finally, we test the accuracy of the algorithm with a simple gaze estimation application. Besides the results given in this section, we present videos and more results on our website [43].

4.1. Detection Performance

In the experiments we used a mobile eye tracker built with two off-the-shelf webcams shown in the video at `<http://goo.gl/wh1NBL>`. Fig. 8 shows the results for a number of cases where the algorithm succeeds in detection of the pupil even when the pupil suffers from motion blur (1^{st} row), is occluded from above, below and from both sides. In the last row the eye is closed, and the algorithm correctly concludes that there is no pupil in the image.

Fig. 9 shows the results for cases where the algorithm fails. The failures can occur as false detections or misses. The common reasons for the failures can be listed as follows:

- ◇ failure on detecting edge segments from pupil contour due to the motion blur or high level occlusion,
- ◇ failure on arc detection due to inaccurate corner detection,
- ◇ failure of the cost function in selecting the correct pupil candidate.

We also present a video consists of 1600+ frames including blinks and occlusive cases at `<http://goo.gl/z2VugA>`.

4.2. Running Time Analysis

Table 3 gives a dissection of the running time of the algorithm for the images given in Fig. 8 and Fig. 9. The running time of the algorithm strictly depends on the detection of the near-circular segment in the first place. If the near-circular segment is found, the algorithm adapts itself and runs much faster. The most time-consuming routines of the algorithm are ellipse

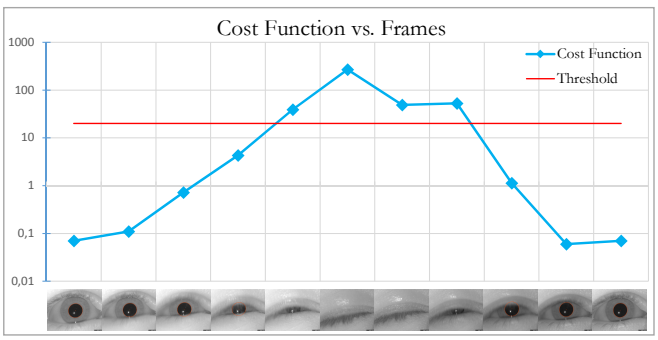


Figure 10: Output of our cost function for a set of example frames. It is seen that the result of cost function increases proportional to the pupil occlusion. If the occlusion is dramatic or there is no pupil in the image, the value of the function overshoots.

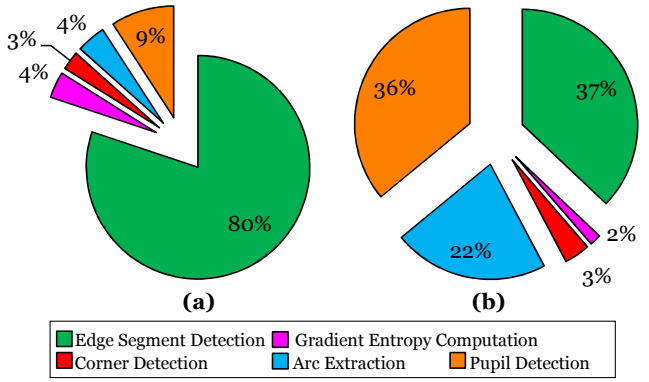


Figure 11: Distribution of execution times among the steps of the algorithm: (a) Easy (non-occlusive) cases, (b) Tough (occlusive) cases.

fitting and error computation methods and they are being invoked intensively during the execution. Especially, fitting error computation method [38] estimates the distance between each point and the ellipse by an iterative approach and its computational cost strictly depends on the total length of extracted arcs in terms of pixels. On the other hand, edge segment detection and gradient entropy computation steps take the same amount of time regardless of the input image as seen in Table 3.

In Fig. 11.a and Fig. 11.b we illustrate the distribution of average execution times for clear and occlusive cases, respectively. As seen in Fig. 11.a, when the pupil is in clear sight, most of the time is spent on edge segment detection and the rest takes very little time. When the pupil is severely occluded and the near-circular segment cannot be found, then arc extraction and joining take more time than edge segment detection (see Fig. 11.b). We can say that the average running times for the images in Fig. 11.a and Fig. 11.b are roughly 10 ms and 30 ms on an 2.8 GHz Intel CPU.

We also prepared a demo video to clearly show how the algorithm adapts itself and runs faster when a near-circular segment is found. In the demonstration, we used a synthetic object and indicated the gradient entropy inside the detected boundary. When near-circular segment cannot be found due to the view angle or occlusion, arc extraction is applied on many segments and running time increases dramatically. More detail is provided on the video explanation at <http://goo.gl/tky8Zr>.

Table 3: Running times (in ms) for examples in Fig.s. on an Intel 2.8 GHz CPU

Input Image	Edge Segment Detection	Gradient Entropy Comp.	Corner Detection	Arc Extraction	Pupil Detection	**TOTAL** (ms)
Fig.7 - 1st row	9.65	0.92	0.07	0.34	0.95	**11.93**
Fig.7 - 2nd row	9.22	0.85	0.05	0.35	0.66	**11.13**
Fig.7 - 3rd row	9.17	0.43	1.21	5.66	7.74	**24.21**
Fig.7 - 4th row	11.48	0.57	1.38	5.44	9.08	**27.94**
Fig.8 - 1st row	10.69	0.48	0.97	7.74	8.12	**28.01**
Fig.8 - 2nd row	9.50	0.29	0.73	5.40	13.49	**29.40**
Fig.8 - 3rd row	7.98	0.12	0.19	0.43	1.15	**9.87**
Fig.8 - 4th row	11.39	0.37	0.78	3.24	7.35	**23.13**
Fig.8 - 5th row	9.64	0.26	0.55	0.76	1.72	**12.93**
Fig.8 - 6th row	9.05	0.29	0.65	0.61	1.20	**11.79**

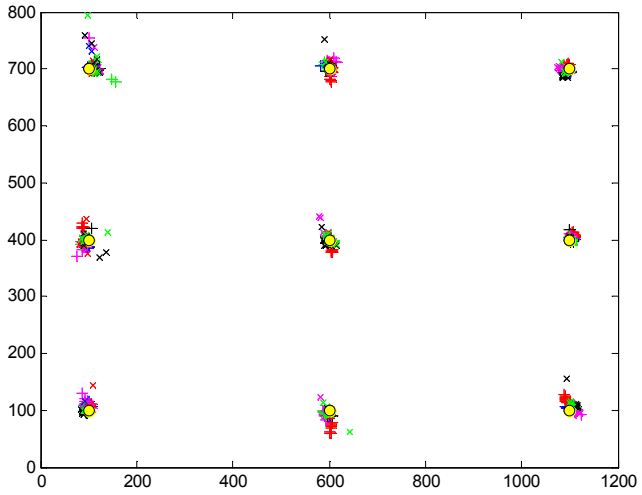


Figure 12: Point-of-gaze estimation errors for 10 different calibrations. Screen calibration points are displayed as yellow circles and each calibration result is indicated with different marking and color.

4.3. Accuracy

After visually inspecting the algorithm and analyzing the timing results, we incorporated the algorithm into a point-of-gaze (PoG) estimation application to obtain quantitative accuracy results. We performed a calibration procedure for 9 screen points and employed a second-order non-linear mapping.

In the experiments, we obtained an average gaze error about 0.3°. Since our system does not compensate the head movements yet, we used a chin-rest during the PoG estimation experiments. Although keeping the head still on a chin rest eases the problem, obtained accuracy value is still a neat achievement. In Fig. 12 PoG error results for 10 different calibrations is presented. Additionally, we present a video including both calibration and tracking parts of our gaze estimation experiment at <http://goo.gl/RVt2sz>. In another video, the user is entering text with our virtual keyboard design [44] at <http://goo.gl/QMOmuo>.

5. Conclusions

Pupil detection is an indispensable step in many eye tracking applications and have to be performed precisely. In most studies, pupil detection is handled with straightforward methods which lack accuracy and fail in occlusive cases. In this study we focused on developing a state of the art algorithm for pupil boundary detection and center estimation. We basically find the arc segments in the image and try to obtain a final ellipse encircling the pupil with the consensus of all obtained features.

Because the edge segment detection method we used has optimum localization, extracted arcs represents the pupil boundary, hence its center, very precisely. This ability of the algorithm can be useful especially in medical applications. Another important property of the algorithm is the proposed heuristic

based on the distribution of gradients. In this way, the algorithm can adapt itself and runs a couple of times faster if there is no or very less occlusion.

As the experimental results show, the proposed algorithm can detect the pupil even in tough occlusive cases without compromising the real-time applicability constraints. According to results, the algorithm is able to detect the pupil boundary even when 25% of its contour is visible by the camera. With the results achieved in experiments, the algorithm offers a promising performance for real life vision applications.

References

[1] Y. Ebisawa, "Improved Video-Based Eye-Gaze Detection Method," IEEE Trans. Instrumentation and Measurement, vol.47, no.4, Aug 1998.

[2] C.H. Morimoto, D. Koons, A. Amir and M. Flickner, "Detection and tracking using multiple light sources," Image and Vision Computing, vol.18, no.4, pp. 331-335, Mar 2000.

[3] J.B. Hiley, A.H. Redekopp, R. Fazel-Rezai, "A Low Cost Human Computer Interface based on Eye Tracking," In Proc. of IEEE EBMS, pg. 3226, 2006.

[4] Y. Ebisawa, "Robust pupil detection by image difference with positional compensation," In Proc. of IEEE VECIMS, pp.143-148, 2009.

[5] J.G. Wang, E. Sung, "Study on eye gaze estimation," IEEE Trans. SMC - Part B: Cybernetics, vol.32, no.3, pp.332-350, Jun 2002.

[6] S. Gõni, J. Echeto, A. Villanueva, R. Cabeza, "Robust algorithm for pupil-glint vector detection in a video-occulography eye tracking system," Proc. of Int'l Conf. on Pattern Recognition (ICPR), 2004.

[7] A. Keil, M. Andreas, K. Berger, M.A. Magnor, "Real-Time Gaze Tracking with a Consumer-Grade Video Camera," In Proc. of WSCG, 2010.

[8] J.S. Agustin, H. Skovsgaard, E. Mollenbach, M. Barret, M. Tall, D.W. Hansen, and J.P. Hansen, "Evaluation of a low-cost open-source gaze tracker," In Proc. of ETRA, ACM, 2010.

[9] X. Long, O.K. Tonguz, A. Kiderman, "A High Speed Eye Tracking System with Robust Pupil Center Estimation Algorithm," Proc. of IEEE EMBS, pp.3331-3334, 2007.

[10] L. Lin, L. Pan, L. Wei, and L. Yu, "A Robust and Accurate Detection of Pupil Images," In Proc. of IEEE Biomedical Engineering and Informatics (BMEI), 2010.

[11] L. Ma, T. Tan, Y. Wang, and D. Zhang, "Efficient iris recognition by characterizing key local variations," IEEE Trans. Image Processing, vol.13. no.6, pp.739-750, June 2004.

[12] D. Ballard, "Generalized Hough transform to detect arbitrary patterns," IEEE Trans. Pattern Anal. Machine Intelligence, vol.13, pp. 111-122, 1981.

[13] T. Mäenpää, "An Iterative Algorithm for Fast Iris Detection," Advances in Biometric Person Authentication, LNCS 2005, vol.3781, pp.127-134, 2005.

[14] S. Dey, D. Samanta, "An Efficient Approach for Pupil Detection in Iris Images," Intl Conf. on Advanced Computing and Communications, pp.382-389, 2007.

[15] D. Zhu, S.T. Moorea, and T. Raphan, "Robust pupil center detection using a curvature algorithm," Computer Methods and Programs in Biomedicine, 59(3), pp.145-157, Elsevier, 1999.

[16] D. Li, D. Winfield, D.J. Parkhurst, "Starburst: A hybrid algorithm for video-based eye tracking combining feature-based and model-based approaches," Proc. of IEEE CVPR - Workshops, 2005.

[17] M. Fischler and R. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," Communications of the ACM, vol. 24, no. 6, pp. 381-395, 1981.

[18] W.J. Ryan, D.L. Woodard, A.T. Duchowski, and S.T. Birchfield, "Adapting Starburst for Elliptical Iris Segmentation," Proc. of IEEE BTAS, 2008.

[19] N. Kumar, S. Kohlbecher, and E. Schneider, "A Novel approach To Video-Based Pupil Tracking," Proc. of IEEE SMC, 2009.

[20] J. Daugman, "How Iris Recognition Works," Proc. of IEEE ICIP, vol.1, pp. 33-36, 2002.

[21] E.M. Arvacheh, H.R. Tizhoosh, "IRIS Segmentation: Detecting Pupil, Limbus and Eyelids," Proc. of IEEE ICIP, pp.2453-2456, 2006.

[22] D.W. Hansen, Q. Ji, "In the Eye of the Beholder: A Survey of Models for Eyes and Gaze," IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 32, no. 3, pp.478-500, Mar. 2010.

[23] C.H. Morimoto, M.R.M. Mimica, "Eye gaze tracking techniques for interactive applications," Computer Vision and Image Understanding, vol.98, pp. 4-24, 2005.

[24] C. Topal and C. Akinlar, "Edge Drawing: A Combined Real-Time Edge and Segment Detector," Journal of Visual Communication and Image Representation, 23(6), 862-872, 2012.

[25] C. Topal, C. Akinlar, Y. Genc, Edge Drawing: A Heuristic Approach To Robust Real-time Edge Detection, The Twentieth Intl Conf. Pattern Recognition (ICPR), pp. 2424-2427, 2010.

[26] C. Topal, O. Ozsen, C. Akinlar, Real-time edge segment detection with edge drawing algorithm, Int'l Symp. on Image and Signal Processing and Analysis (ISPA), Dubrovnik, Croatia, September 2011.

[27] Edge Drawing website and online demo page. http://ceng.anadolu.edu.tr/CV/EdgeDrawing. Accessed on 27.Agu.2013.

[28] C. Akinlar, C. Topal, EDPF: A Real-time Parameter-free Edge Segment Detector with a False Detection Control, International Journal of Pattern Recognition and Artificial Intelligence, 26 (1), 2012.

[29] EDPF website and online demo page. http://ceng.anadolu.edu.tr/CV/EDPF. Accessed on 27.Agu.2013.

[30] A. Desolneux, L. Moisan, J.M. Morel, Edge detection by Helmholtz principle, Journal of Mathematical Imaging and Vision, 14(3), 271-284, 2001.

[31] A. Desolneux, L. Moisan, J.M. Morel, Meaningful Alignments, International Journal of Computer Vision, 40(1), pp.7-23, 2007.

[32] L. Jia and L. Kitchen, "Object-Based Image Similarity Computation Using Inductive Learning of Contour-Segment Relations," Trans. Image Processing, 9(1), pp.80-87, Jan. 2000.

[33] M. Shia, Y. Fujisawab, T. Wakabayashia, F. Kimura, "Handwritten numeral recognition using gradient and curvature of gray scale image," Pattern Recognition 35(10), pp.2051-2059, Oct. 2002.

[34] A.C. Jalba, M.H. F. Wilkinson, J.B.T.M. Roerdink, "Shape Representation and Recognition Through Morphological Curvature Scale Spaces," Trans. Image Processing, 15(2), pp.331-341, Feb. 2006.

[35] V. Ferrari, L. Fevrier, F. Jurie, and C. Schmid, "Groups of Adjacent Contour Segments for Object Detection," IEEE Trans. PAMI, 30(1), Jan. 2008.

[36] C. Martinez-Ortiz, J. Žunić, "Curvature weighted gradient based shape orientation ," Pattern Recognition, 43(9), pp.3035-3041, Sep. 2010.

[37] A. Fitzgibbon, M. Pilu, and R.B. Fisher, "Direct Least Square Fitting of Ellipses," IEEE Trans. Pattern Anal. Machine Intelligence, vol. 21, no. 5, May 1999.

[38] R. Nürnberg, Distance from a Point to an Ellipse, 2006. Online available at http://www2.imperial.ac.uk/ rn/distance2ellipse.pdf. Accessed on 27.Agu.2013.

[39] S. Ramanujan, "Collected Papers of Srinivasa Ramanujan," Chelsea Publishing, New York, 1962.

[40] C. Akinlar, C. Topal, "EDCircles: A real-time circle detector with a false detection control," Pattern Recognition, 46(3), 725-740, March 2013.

[41] EDCircles website and online demo page. http://ceng.anadolu.edu.tr/CV/EDCircles. Accessed on 27.Agu.2013.

[42] C. Topal, K. Özkan, B. Benligiray, C. Akinlar, "A Robust CSS Corner Detector based on the Turning Angle Curvature of Image Gradients," IEEE Int'l Conf. Acoustics, Speech and Signal Processing (ICASSP), Vancouver, CANADA, 2013.

[43] Anadolu CVLab Eye tracking research website. http://ceng.anadolu.edu.tr/CV/EyeTracking. Accessed on 27.Agu.2013.

[44] C. Topal, B. Benligiray, C. Akinlar, "On The Efficiency Issues of Virtual Keyboard Design," In Proc. of IEEE Int'l Conf. VECIMS, Tianjin, CHINA, 2012.