Event-based eye-tracking with template matching using the deformation of the iris

Yvonne Vullers
y.a.g.vullers@student.tue.nl
Supervisors: Arren Glover (IIT), Luna Gava (IIT), Chiara Bartolozzi (IIT), Federico Corradi (TU/e)

Abstract—**Being able to accurately track the eye's gaze is an important property in applications such as virtual reality and medical applications. Accurate eye-tracking requires a fast algorithm, however, frame-based methods have a trade-off between accuracy and frequency. Instead, we propose an algorithm that uses event-data only, which allows a fast and low-power algorithm. By taking advantage of the high temporal resolution of event-data, the approach is more lightweight than event-based methods using CNNs. The iris center and gaze direction can be estimated from a model representing the shape of the iris. Negative influences in the accuracy due to occlusions of the iris by the eyelids were avoided by considering cuts in the model of the iris. The algorithm resulted in accurately tracking the eye in real-time, however only in a constrained environment without movements of the head or blinks. Nevertheless, our model improved the mean error compared to an event-data-only eye-tracking method tested on the same datasets. Additional steps, e.g. implementing a detector, need to be taken in order to realize a real-life implementation.**

I. Introduction

Eye-tracking is the process of estimating the direction of interest of a person, i.e. the gaze, using camera frames. Gaze plays a role in understanding the emotional state [1] and in estimating the attentional state of humans [2]. This allows many applications in augmented and virtual reality (AR/VR), e.g. foveated rendering [3] and having interaction with a program using the eyes [4], to benefit from eye-tracking technologies. Additionally, eye-tracking enables medical applications such as eye laser-surgery [5] and psychological research [6].

Visual methods for tracking gaze have been developed but traditional camera technology has drawbacks for the gaze tracking task. These cameras have a fundamental trade-off between accuracy, latency, and power consumption: For accurate eye-tracking, a high frame-rate and high resolution are required. This results in a high volume of data that should be processed, therefore the process can be slow and consume a lot of power. This is not favorable in real-time applications where low-latency is required or in the case of AR/VR where wearable devices only have limited power available [7].

Instead, event-based cameras can be used for acquiring the eye-data. These cameras respond only to local changes in brightness, i.e. for each pixel, and output a stream of events. The pixels only respond in case of a movement in the scene, therefore the camera is asynchronous. Event-cameras offer many benefits: first, they offer a high temporal resolution, as monitoring only brightness changes is fast. This also reduces motion blur in the data. Next, low-latency is achieved, because each pixel can respond independently from the other pixels in the frame. Lastly, the volume of the data is low since there is only a response in the case of a movement in the scene, therefore the event-cameras are also low-power [8].

However, it is not possible to directly use existing methods for eye-tracking with event-data. Additionally, these methods will not take advantage of the high temporal resolution of the event-data: Because the time between two events is very small, the spatial displacement of the iris between two events will also be small. Therefore, the algorithm only needs to check for small changes, allowing it to be fast and lightweight.

Only a few event-based methods have been developed for eye-tracking. Some of these methods are successful in finding the iris center using a convolutional neural network (CNN), however, these methods are not able to find the gaze direction [9–12]. [7] is able to find the gaze direction of the user from a model of the eye-ball. Though they try to benefit from the high temporal resolution of the event-data, frames are still used, therefore not fully taking advantage.

Instead, in this paper we propose a template matching-based eye-tracking method using event-data. This algorithm:

- Has the potential for low power consumption and a high frequency compared to frame-based or hybrid methods.
- Estimates the gaze direction as well as the iris center.
- Takes advantage of the high-temporal resolution of the event-data, allowing a more lightweight approach than event-based methods using CNNs.

II. Related Works

The field of eye-tracking is widely researched. There are different parts of the eye that research focuses on, for example blinks [9, 13, 14], the glint [15–17], or the pupil or iris [7, 9–12, 18–23]. Using these features, the pupil or iris position or the gaze of the eye can be tracked. In this work, we will focus on both the iris center position and the gaze direction by tracking the iris.

Work in this field can generally be divided into two approaches: appearance-based methods and model-based methods. In appearance-based methods [18–20], the pupil/iris position or the gaze direction is determined directly from the features of the eye, typically by using convolutional neural networks or other data-driven approaches. Model-based methods attempt to fit features of the eye to an eye-model and find the iris position and the gaze direction from the state of the model. In the model, the deformation of the iris shape is considered for different rotations of the eye. The state of the model is then found by comparing it to the eye-data by means of template matching [21, 22] or by fitting landmarks to the model using optimization [23].

Methods in both approaches are able to achieve high accuracy, however, all these are subject to a fundamental trade-off between accuracy, latency, and power-consumption as all methods use a frame-based camera [7]. Instead, event-based cameras can be used to reduce the latency and power-consumption [8].

Event-based eye-tracking is still a fairly new topic, but in the few methods that have been proposed it is again possible to make a distinction between appearance-based and model based-methods. In [9–12] appearance based methods are proposed, using a CNN to find the iris or pupil position. Models such as YOLO [9], a convolutional Long Short-Term Memory network [10], or a convolutional encoder-decoder structure (U-Net) [11] are used for achieving this. [12] implements a simpler approach with a convolutional sliding window using a single filter to find the position.

However, these methods are not able to find the gaze direction which is an important property in applications such as AR/VR [21]. A model-based method, such as [7], is able to estimate the gaze of the eye, while also trying to benefit from the asynchronous properties of the event-data. However, for finding the shape of the pupil, frames are also used, therefore not taking full advantage of the sparsity and asynchronous properties of the event-data.

In this work, we propose a method that solely uses event-data to estimate the center position of the iris and the gaze of the eye, therefore benefiting from the low-latency and low-power of the event-camera. By using the eye-ball model proposed in [22], the center position of the iris can be accurately estimated and additionally, the gaze direction can now be estimated. The high temporal resolution of the event-data allows template matching to be used for finding the state of the model with only a limited amount of templates, therefore the algorithm is faster and more lightweight than CNN methods using event-data [9–11].

III. Methods

The eye-tracking algorithm aims to find the gaze direction of the eye at each time step. The gaze direction is determined from the current state of the eyeball. This state can be represented by

$$X = [u, v, \theta, \phi, R] \tag{1}$$

where u and v denote the position of the center of the eyeball in the frame in pixels, θ and ϕ the yaw and pitch orientation of the eye in radians, and R the radius of the eyeball.

In the eye-tracking algorithm we can consider three main components:

- The processing of the input data from the event-camera. Given the event-data a constant and persistent observation is produced.
- The model of the eye. This model defines the iris shape as projected on the 2D image plane given any possible eye state defined in 1.
- The tracking of the eye. Here the state at $t + 1$ will be estimated given the state X and the observation at t.

In Fig. 1, the full pipeline is depicted. We first receive the raw events from the event-camera, after which it is processed and represented on the Exponential Reduced Ordinal Surface (EROS) [24]. This surface is matched to templates created from the current state model. For the first iteration, an initial state X_0 is considered. The best fitting template is used to update parameters θ and ϕ of the current state.

A. Data

The event camera produces so-called events, which are spikes from individual pixels if the brightness changes in that pixel. This event is represented as a tuple

$$event : \{t, [v_x, v_y], p\} \tag{2}$$

containing the timestamp at which the event takes place, the $[v_x, v_y]$ position of the pixel where the event is observed, and the polarity p of the event, i.e. the direction of the brightness change in the pixel.

Instead of using a fixed temporal window, which shows all events that occurred within a certain time interval, EROS [24] will be used to represent the event data. This representation can be considered a pseudo-frame, like an image, but it is updated asynchronously, i.e. for every new event. Using EROS allows the representation of events to be persistent and constant, which is not possible with just using a fixed temporal window: A too-small window shows no edges when there is no movement, which can cause the tracking to fail when random noise is present. Conversely a window too large will show a very wide, unclear edge, making it impossible to track. Therefore a velocity-independent representation, such as EROS, is required for successful tracking.

The representation, described by Algorithm 1, is defined

Algorithm 1 EROS update algorithm

```
for $x = v_x - k_{EROS} : v_x + k_{EROS}$ do
    for $y = v_y - k_{EROS} : v_y + k_{EROS}$ do
        $EROS_{xy} \leftarrow EROS_{xy} \times d_{EROS}^{1/k_{EROS}}$
    $EROS_{v_x v_y} \leftarrow 1.0$
```

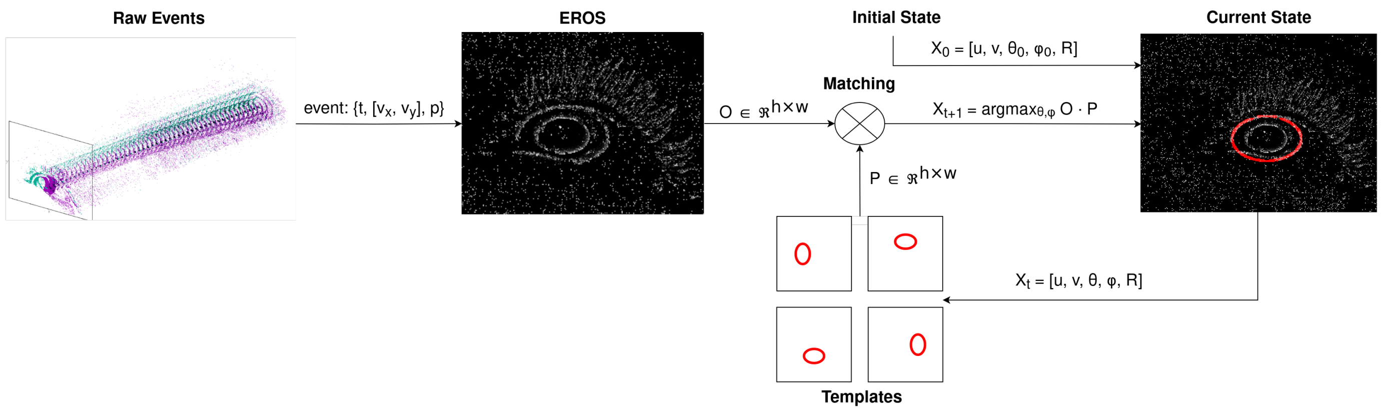


Fig. 1: The full pipeline for eye-tracking with events using template matching.

on a 2D array with the dimensions equal to the camera resolution $h \times w$. When a new event is received, the pixel on the EROS frame at the same location where the event was received is updated to the maximum value. The pixel values in the neighborhood around this pixel, defined by a kernel-size k_{eros}, are then multiplied by an exponential decay factor $d_{eros}^{1/k_{eros}}$, resulting in a clear edge of the object at the event location. Afterward, a median filter is applied to the surface to eliminate some of the noise present. The EROS represents the observation $O \in \Re^{h \times w}$ of the eye.

B. Eye-model

When rotating the eyeball in the world space, the iris appears to be deforming from a circle to an ellipse on the image plane. Therefore, when attempting to find the current state X of the eyeball, the shape of the iris should be accurately represented. To achieve this, a representation of a circle (the iris) on a rotating sphere (the eyeball) should be defined. To achieve this, the derivation of the eyeball model in [22] will be followed, however instead of yaw and roll, yaw and pitch rotations of the eyeball will be considered.

In case of no rotations, the equation describing the circular iris of an eye with unit eyeball radius and eyeball center $[0, 0]$ is equal to

$$(x, y) = (\tau \cdot cos(\alpha), \tau \cdot sin(\alpha)), \alpha \in [-\pi, \pi], \tag{3}$$

where α is the central angle of the circle and τ is a parameter that denotes the ratio between the eyeball radius and iris radius. This value is 0.5, based on the anthropometric information of the eye structure [25].

Mathematical morphology can be used to describe the projection on the 2D-image plane of a circle on a 3D rotated sphere. First, the rotation should be defined. The eye either rotates by an angle θ around the x-axis (pitch) or ϕ around the y-axis (yaw). This rotation is described by

$$R_x(\theta)R_y(\phi) = \begin{bmatrix} cos(\phi) & 0 & sin(\phi) \\ sin(\theta)sin(\phi) & cos(\theta) & -sin(\theta)cos(\phi) \\ -cos(\theta)sin(\phi) & sin(\theta) & cos(\theta)sin(\phi) \end{bmatrix} \tag{4}$$

using the rotation matrices $R_x(\theta)$ and $R_y(\phi)$ of the 3D rotation group SO(3).

Then, to define the rotation of the eyeball, the spherical shape should be included in the rotation. A sphere is described by the equation $x^2 + y^2 + z^2 = 1$, therefore z can be expressed in terms of x and y as

$$z = \sqrt{1 - x^2 - y^2}, \tag{5}$$

which can be simplified for a point on a circle on the sphere to

$$z = \sqrt{1 - \tau^2} \tag{6}$$

using equation 3. Thus when rotating the sphere with an angle θ and ϕ, the point (x, y, z) on a circle will have the new position

$$\begin{aligned} \begin{bmatrix} x_r \\ y_r \\ z_r \end{bmatrix} &= R_x(\theta)R_y(\phi) \cdot \begin{bmatrix} x \\ y \\ z \end{bmatrix} \\ &= \begin{bmatrix} cos(\phi) & 0 & sin(\phi) \\ sin(\theta)sin(\phi) & cos(\theta) & -sin(\theta)cos(\phi) \\ -cos(\theta)sin(\phi) & sin(\theta) & cos(\theta)sin(\phi) \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ \sqrt{1-\tau^2} \end{bmatrix} \\ &= \begin{bmatrix} x \cdot cos(\phi) + \sqrt{1-\tau^2} \cdot sin(\phi) \\ x \cdot sin(\theta)sin(\phi) + y \cdot cos(\theta) - \sqrt{1-\tau^2} \cdot sin(\theta)cos(\phi) \\ -x \cdot cos(\theta)sin(\phi) + y \cdot sin(\theta) + \sqrt{1-\tau^2} \cdot cos(\theta)sin(\phi) \end{bmatrix} \end{aligned} \tag{7}$$

From this, we find the x and y position in the image plane of all the points on a circle on a rotated sphere to be

$$\begin{bmatrix} x_{plane} \\ y_{plane} \end{bmatrix} = R \cdot \begin{bmatrix} x \cdot cos(\phi) + \sqrt{1-\tau^2} \cdot sin(\phi) \\ (x \cdot sin(\phi) - \sqrt{1-\tau^2} \cdot cos(\phi)) \cdot sin(\theta) \\ +y \cdot cos(\theta) \end{bmatrix} + \begin{bmatrix} u \\ v \end{bmatrix} \tag{8}$$

with x and y defined as in equation 3 to represent the circular iris, scaled with the radius of the eyeball R and taking into account the actual center position of the eyeball $[u, v]$ in the image plane.

C. Tracking

Due to the small spatial displacement between the two events, only potential states close to the current state of the iris need to be checked. In order to compare the projections of potential states to the observation O, template matching is used. By finding the template that is closest to the data, the state of the scene can be determined.

In the case of the iris and the eyeball, the potential states that are considered are where the eyeball is slightly rotated compared to the previous state, i.e. $\theta \pm \Delta\theta$ or $\phi \pm \Delta\phi$. The projection on the image plane of these potential states is the expectations $P \in \Re^{h \times w}$. Additionally, the expectations are smoothed using a Mexican-hat filter to have more tolerance to noise or inaccuracies in the model. In every iteration, these templates are compared to the current EROS surface, the observation O, by means of a similarity score

$$score = O \cdot P. \tag{9}$$

The θ and ϕ of the template with the highest score will be considered the new state for the next iteration. If the templates of possible future states show no improvement to the current state, the state will not be updated.

However, one change in the rotation will cause a different pixel shift of the iris center at different orientations of the eyeball: At $\theta = 0, \phi = 0$, the displacement of the center of the iris in pixels is much bigger for a change in rotation than when θ and ϕ are bigger. Therefore using the same change in rotation for all possible states can cause problems: the template will not shift enough at larger θ and ϕ or too much at a smaller value.

To solve this, it should be determined how much the eyeball should rotate to achieve a certain pixel-shift Δp for every possible state. First, the center of the iris on the image plane $[x_c, y_c]$ should be determined. This can be found by setting the values x and y in equation 8 to zero. This results in

$$\begin{bmatrix} x_c \\ y_c \end{bmatrix} = R \cdot \begin{bmatrix} \sqrt{1-\tau^2} \cdot sin(\phi) \\ -\sqrt{1-\tau^2} \cdot cos(\phi) \cdot sin(\theta) \end{bmatrix} + \begin{bmatrix} u \\ v \end{bmatrix}. \tag{10}$$

We can then find the amount of change in rotation given a desired pixel shift by solving

$$\begin{bmatrix} x_c + \Delta_p \\ y_c + \Delta_p \end{bmatrix} = R \cdot \begin{bmatrix} \sqrt{1-\tau^2} \cdot sin(\phi + \Delta\phi) \\ -\sqrt{1-\tau^2} \cdot cos(\phi) \cdot sin(\theta + \Delta\theta) \end{bmatrix} + \begin{bmatrix} u \\ v \end{bmatrix}. \tag{11}$$

For movement along the x-axis, the change in rotation will be equal to

$$\Delta\phi = sin^{-1}\left(\frac{x_c + \Delta_p - u}{R \cdot \sqrt{1-\tau^2}}\right) - \phi \tag{12}$$

and for movement along the y-axis

$$\Delta\theta = sin^{-1}\left(-\frac{y_c + \Delta_p - v}{R \cdot \sqrt{1-\tau^2} \cdot cos(\phi)}\right) - \theta. \tag{13}$$

D. Influence surrounding features

As we are only interested in the movement of the eye, the influence of surrounding features, e.g. the eyelids or eyelashes, should be limited. Especially for large rotations θ and ϕ where parts of the iris are occluded by the eyelid, the model should not accidentally match the eyelids. In both the data representation and the model, changes can be made to realize this.

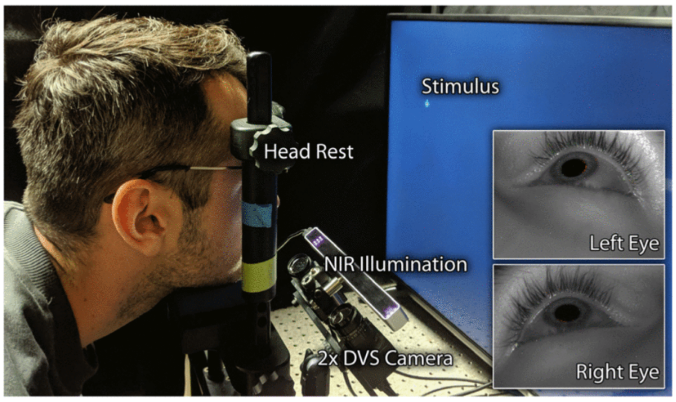


Fig. 2: The setup for the data collection of the 27 users dataset [7].

The surrounding features can be excluded from the data by only considering values at pixels that lie between the eyelids. To achieve this, the eyelids are approximated by polynomials before the start of tracking. EROS is updated only for v_x and v_y lying between these polynomials. The EROS with cuts is then considered the observation $O \in \Re^{h \times w}$ of the eye.

In order to limit the influence of surrounding features of the iris in the model, cuts are made in the ellipse. To achieve this, we consider $\alpha \in [-\pi, -3\pi/4] \wedge \alpha \in [-\pi/4, \pi/4] \wedge \alpha \in [3\pi/4, \pi]$. This way, the cuts are located in the areas where the eyelids will be. The resulting ellipse segments are then the expectation $P \in \Re^{h \times w}$.

IV. Experiments

Experiments are performed to analyze which algorithm features are the most influential for the performance. Both the data representation and the cuts in the EROS and model are considered here. Additionally, the algorithm is compared to an available implementation in accuracy. This algorithm of [12] was chosen because this is the only fully event-based method that uses this dataset and the same ground truth. Next, the gaze direction is evaluated to show this can be found from the eyeball model. Lastly, the algorithm is evaluated in terms of speed.

A. Dataset

The dataset that is used for the experiments is the 27-users dataset [7]. In this dataset, the movements of the left and the right eye of 27 users are recorded using the DAVIS346b sensor, resulting in 54 recordings of events. The types of movements of the eye are random saccades and smooth pursuit, collected in two experiments where the users were looking at stimuli on a screen. The setup is depicted in Fig. 2. The users do not move their heads during the recordings,

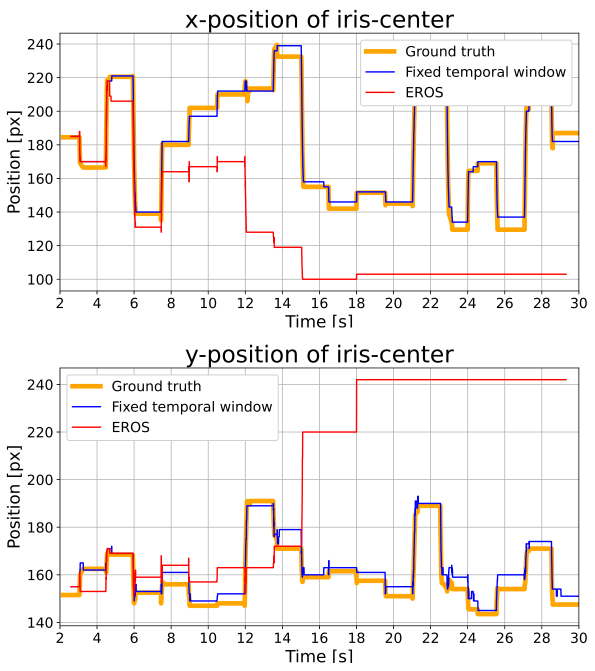


Fig. 3: The x and y position for the left eye of user 5 determined by the tracker using a fixed temporal window and a cut model compared to the ground truth.

	Mean error [px]
EROS	4.129 $\pm$ 3.963
Fixed temporal-window	20.619 $\pm$ 15.820

TABLE I: The mean pixel error and standard deviation over all the users using EROS compared to the mean pixel error using a fixed temporal window.

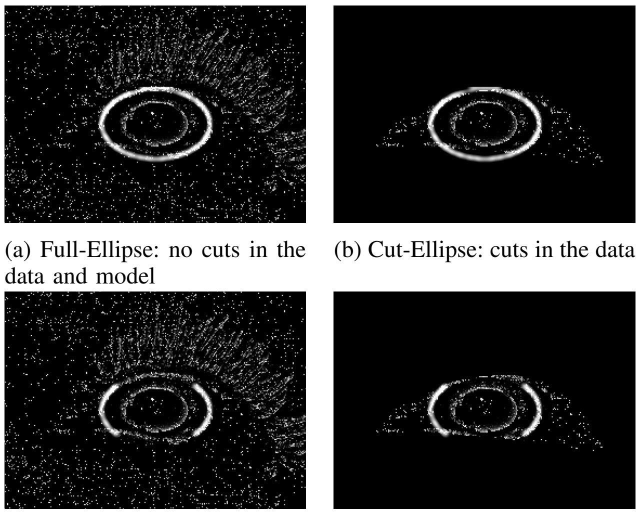
therefore the position of the center of the eyeball $[u, v]$ and the iris radius R of a user are constant. These values and the initial orientation of the eye θ and ϕ are found by hand and define the initial state X_0.

Since the algorithm is only able to perform tracking and recovering tracking after a blink detection is needed [12], only users that have no blinks in the first 30 seconds are considered to assess the performance of the tracking. The eyes that are considered are the left and the right eye of users 5, 13, 18, and 19. For all users the same algorithm parameters are used, namely $k_{eros} = 15$ and $d_{eros}^{1/k_{eros}} = 0.6$. For the median filter on the EROS, a kernel of 3 is used and the Mexican-hat filter has a 2D kernel of 15×15

The ground truth of the dataset consists of the center of the pupils at each time step and is hand-labeled by [12]. This ground truth is only available for the first 30 seconds of data for each user, therefore the experiments are only performed for these 30 seconds.

B. Data representation study

The performance of the algorithm on the EROS surface is compared to using a fixed temporal window of $4ms$. The error in the center position of the iris for the fixed temporal-window, the error is much higher than when using the EROS, as denoted in Table I. The plot of the x and y position of the iris center determined by the tracker using a fixed temporal window is depicted in Fig. 3. This shows that after only 8 seconds the eye is not tracked anymore, therefore the velocity-independent property of EROS is of benefit for the successful tracking of the eye.



(a) Full-Ellipse: no cuts in the data and model (b) Cut-Ellipse: cuts in the data (c) Full-Segments: cuts in the model (d) Cut-Segments: cuts in the data and model

Fig. 4: The data and the model for the four experiments in the ablation study

C. Ablation study

In an attempt to reduce the possible influence of surrounding features of the iris, there are cuts in the EROS and the model of the iris. In order to investigate the actual influence of these cuts on the performance of the tracking, an ablation study is performed. The EROS and model for the four experiments that are performed are depicted in Fig. 4. Fig. 8 depicts the tracked x and y position of the iris-center for all experiments and the ground truth for a single user. The figure shows that the algorithms using the full EROS and full ellipse and the cut EROS and full ellipse loses tracking at some points, e.g. at $13s$, $27s$ and $29s$, whereas the other two algorithms are able to track the position throughout the whole experiment.

The mean pixel error over all users for these experiments is denoted in Table . We can see from the table that just making cuts in the EROS will not improve the accuracy. However, the mean error does decrease significantly when only using segments of the ellipse on the full surface. The error is small compared to the diameter of the iris, which is on average 105 pixels. Lastly, it can be noted that making cuts in both the data and the model shows improvements in the pixel error compared to the experiments where the full ellipse was used. However, it does not show improvements compared to the ellipse segments as the model and using the full EROS.

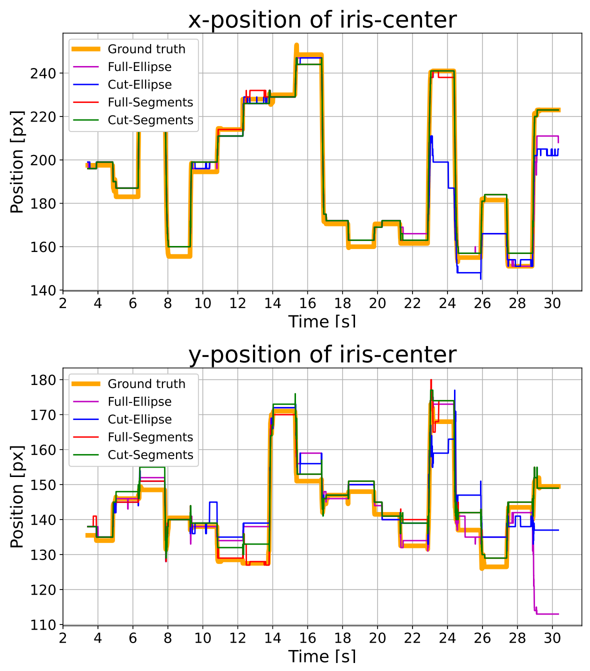


Fig. 5: The x and y position for the left eye of user 19 determined by the tracker for the four experiments compared to the ground truth

	Mean error [px]
Full-Ellipse	11.489 $\pm$ 10.520
Cut-Ellipse	11.613 $\pm$ 10.674
Full-Segments	4.129 $\pm$ 3.963
Cut-Segments	8.235 $\pm$ 7.958

TABLE II: The mean pixel error and standard deviation over all the users for the full EROS and full model, with cuts in the EROS, with cuts in the model, and with cuts in both.

D. Comparison study

Next, the best performance of the tracking algorithm is compared against the algorithm of [12]. The results of the comparison over the users 5, 13, 18, and 19 for both algorithms are depicted in Table III . Only the right eye of user 18 is excluded, because this data of [12] was not available. From this we can conclude that our algorithm has a smaller mean pixel error.

In Fig. 6 the difference in error is clearly visible. In general, the error of [12] is slightly higher, with exceptions at $6, 18, 23$ and $25s$ where the error is much higher. After this high error, the error is quickly reduced to a much smaller value. This is where the detector in the algorithm of [12] needs to correct the algorithm due to mistakes in the tracking. It should be noted that our algorithm only employs tracking and does not use a detector.

E. Final performance

The experiments were performed on an Intel Core i7-9750H CPU @ 2.60GHz. On this platform, the frequency of the algorithm was $424Hz$. This frequency was achieved without any optimization of the algorithm. The used pixel shift for

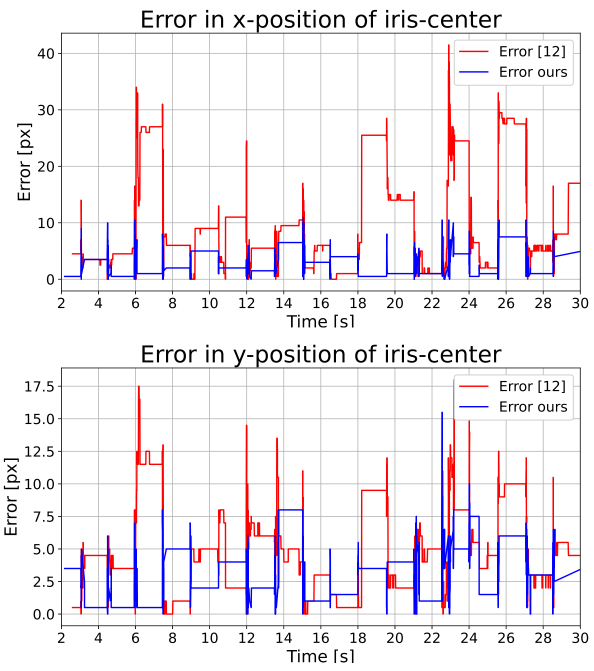


Fig. 6: The error in the x and y position for the left eye of user 5 determined by the tracker using the full EROS and a cut model compared to the tracking error of [12].

	Mean error [px]
Ours	4.191 $\pm$ 3.963
[12]	7.349 $\pm$ 10.284

TABLE III: The mean pixel error and standard deviation over the users compared to the mean pixel error of [12].

potential states is $\Delta_p = 3$, therefore the algorithm is able to follow a shift of maximum $424 \cdot 3 = 1272$ pixels per second of the iris. If the maximum velocity of the dataset is smaller or equal to this value, the algorithm is able to track the eye in real-time. It was qualitatively observed that the algorithm can run with the data at the normal speed, therefore we can assume it is fast enough for the eye-tracking task.

From Table II, we see that the best performance is obtained by using the model with the ellipse segments. In Fig. 8, the x and y position of the center of the iris are compared to the ground truth for a single user. The figure shows that the model responds at the right time by moving in the correct direction for both the x and y coordinates. However, there are small mistakes in the exact position of the iris-center, especially in the y coordinate. In some cases, there is an error present that seems not to be caused by an incorrect state of the model, e.g. at $t = 8s$. The model appears to fit perfectly on top of the EROS, as depicted in Fig. 9, however, the estimated center position is not equal to the ground truth.

In addition to finding the iris center, our algorithm is able to estimate the gaze direction, which was not possible with the algorithm of [12]. In Fig. 7 this property is shown with an arrow in the direction of the gaze. This direction in the 3D world-space is directly determined by θ and ϕ. We

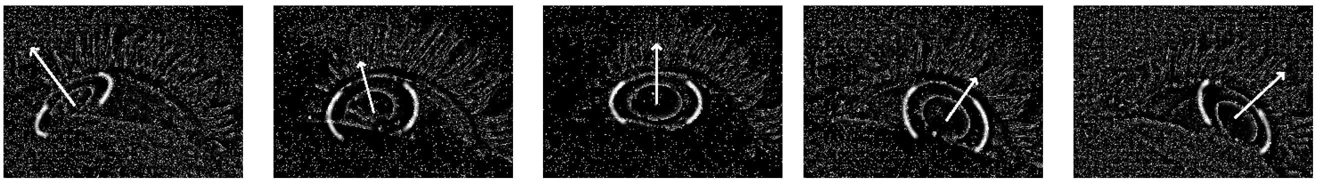


Fig. 7: An arrow denoting the gaze direction for when the eye is looking up to the far left (a), left (b), middle (c), right (d), and far-right (e).

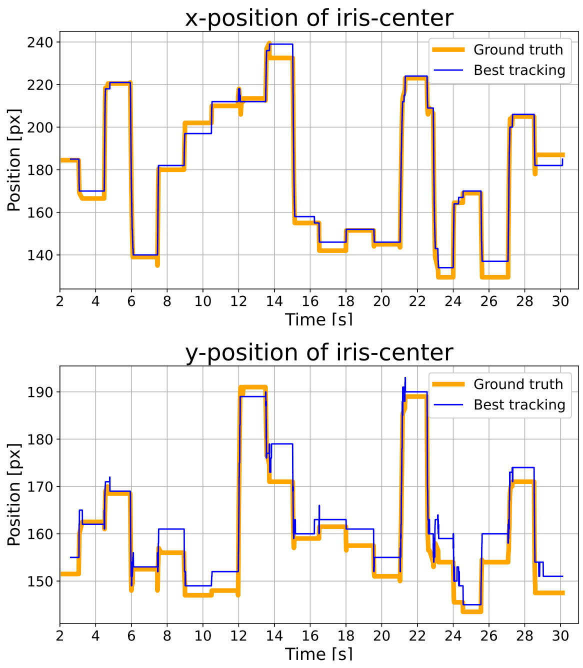


Fig. 8: The x and y position for the left eye of user 5 determined by the tracker using the full EROS and a cut model compared to the ground truth

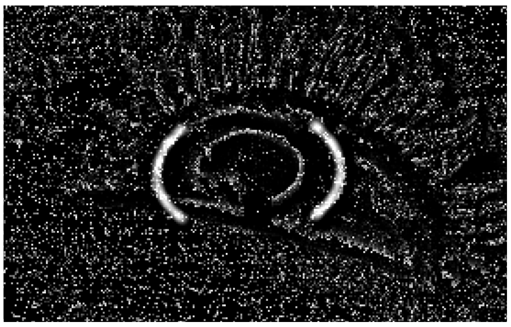


Fig. 9: The model on top of the EROS at $t = 8s$

see that the estimated direction of the gaze by the current state corresponds to where the eye is looking.

V. Discussion

The algorithm was able to track the iris real-time with a small error, of only 4% of the iris diameter. In order to avoid any negative effects from the occlusion of the eye due to eyelids, cuts were made in both the surface and the model. For the cuts in the surface, the eyelids were approximated by polynomials. We assumed the position of the eyelids to be constant while the eye is constant, however, this assumption proved not to be true, therefore there was no advantage in updating the surface only between the static eyelids. Instead making cuts in the model proved to work better. Additionally, these cuts are much simpler to create than finding a polynomial approximation for the eyelids of each user.

Compared to the work using a convolutional sliding window [12], the algorithm showed better results for finding the center of the iris. Instead of using a single circular kernel, we considered the actual shape of the iris, therefore the position could be estimated more accurately.

However, there are still some limitations to the approach. The limitations can be divided into two parts: Limitations of the algorithm itself and limitations in how the performance of the algorithm was evaluated.

A. *Limitations of the Algorithm*

The first thing that should be noted is the constraints of the algorithm. The algorithm considers the initial position of the center of the eyeball $[u, v]$ to be constant over the whole period of tracking for a user. In the used dataset this can be assumed because there are no movements of the head or the camera. However, in real-life scenarios, these movements are likely to happen. In order to also track a changing eye position, the $[u, v]$ position of the state could be also estimated. Because the initial state is determined by hand before tracking, unlocking $[u, v]$ could also solve any errors in the tracked position and gaze that might be caused by an incorrectly estimated initial state. Further experiments are needed to evaluate if tracking still converges with the higher dimensional state.

Additionally, finding the initial state by hand is time-consuming. Therefore applying the algorithm to a new user is not a fast operation. In order to have a fast and accurate initialization of the initial state, a detector should be implemented that considers all possible orientations and positions of the eyeball. Also, because there is no detector it is not possible to continue tracking after a blink occurs. During a blink, it is possible the iris moves and therefore the last tracked state is not accurate anymore.

By being able to detect the current orientation of the eye, tracking will be possible for longer periods of time. Because the algorithm allows all users to be tracked using the same parameters, it should be possible to implement such a detector.

Next, for determining the gaze direction, we assume θ and ϕ are the actual gaze direction of the user, however, this assumption can only be made when the camera is directly in front of the user. θ and ϕ are the gaze direction relative to the vector between the eyeball center and the camera, therefore the position of the camera relative to the eyeball should be taken into account for finding the actual gaze of the user.

Another assumption that is made is about the movement of the eye. Because the displacement of the iris between two events is very small, at only a few pixels, we assume the movement in the x and y direction to be independent and therefore only update by $\theta \pm \Delta\theta \vee \phi \pm \Delta\phi$. If this cannot be assumed at this small scale, diagonal updates $\theta \pm \Delta\theta \wedge \phi \pm \Delta\phi$ should be included in tracking.

Lastly, because we track the iris there is occlusion of the shape when the eye moves away from the center. This is where there is the most error in tracking the position of the iris. With the cuts in the model, there is already an attempt to omit accidentally tracking surrounding features when the iris is not visible, however, this is done in a naive way where the cuts are the same, even when the iris is at an opposite orientation. Instead, it should be taken into account what part of the iris is likely to be occluded at a certain rotation to make tracking easier.

B. Limitations in the Evaluation

The first limitation is because of the constraint that there can be no blinks during the tracking, only a limited amount of users can be evaluated. Additionally, implementing a new user was a time-expensive operation, because for every user the initial state has to be determined by hand. Because of this, additional users who did not have any blinks in the first 30 seconds were not able to be tested. Having more users allows a stronger conclusion to be drawn from the experiments.

Next, the limitations of the available ground truth should be considered. As the ground truth only offers the true center position of the iris, it is not possible to verify whether the gaze direction that is found is actually correct. From the images in Fig. 7 we can see that it seems to be in the right direction, but whether this is actually correct is unclear.

Additionally, we see in Fig. 8 that when the eye is not moving there is a small error in the position for both the x and y coordinate, but it is constant over a small period of time, therefore there is not a better fitting state that can be found. Also comparing the model to the shape of the iris in Fig. 9 at the time of the error, the fit seems to be perfect. There could be a mismatch in how the center of the iris is determined, as the ground truth is determined from a bounding box around the pupil, while the model finds the center directly by fitting the iris. Further inspection is needed to see if this difference in format causes the error in the center position.

VI. Conclusion

In this paper, we presented an algorithm that is able to track the position of the iris and the gaze of the eye using solely event-data. This was achieved by creating templates for possible orientations of the eye using an eye-model describing the deformation of the iris under rotation of the eye. These templates were matched to the event-data by using the EROS as a data representation.

The best results for the eye-tracking are found when there are cuts in the model. Therefore these model cuts help to avoid the influence of surrounding features of the eye, e.g. eyelid occlusions, which occur often in eye-tracking.

The tracking algorithm is able to achieve small pixel errors when tracking the users' eyes on a subset of the 27 users dataset compared to a current algorithm, with an error of 4.129 pixels. Next to this, it is also possible to track the gaze of the eye from the events. This algorithm is able to run real-time, however, there are some limitations, both in the algorithm and the evaluation. Because there is no detector for the iris, the initial state has to be manually found and the tracking only works when the user does not blink. Additionally, the user cannot move their head, as only rotations of the eyeball are considered in the model. In the evaluation, the ground truth should be carefully inspected as an inexplicable error is always present. Furthermore, a ground truth for the gaze direction is not available. The limitations should be considered and improvements should be made before the tracker can be implemented in real-life scenarios.

References

[1] J. Reginald B. Adams and R. E. Kleck, "Perceived gaze direction and the processing of facial displays of emotion," *Psychological Science*, vol. 14, no. 6, pp. 644–647, 2003, pMID: 14629700. [Online]. Available: https://doi.org/10.1046/j.0956-7976.2003.psci˙1479.x

[2] R. Stiefelhagen, J. Yang, and A. Waibel, "Estimating focus of attention based on gaze and sound," in *Proceedings of the 2001 Workshop on Perceptive User Interfaces*, ser. PUI '01. New York, NY, USA: Association for Computing Machinery, 2001, p. 1–9. [Online]. Available: https://doi.org/10.1145/971478.971505

[3] B. Guenter, M. Finch, S. Drucker, D. Tan, and J. Snyder, "Foveated 3d graphics," *ACM Trans. Graph.*, vol. 31, no. 6, nov 2012. [Online]. Available: https://doi.org/10.1145/2366145.2366183

[4] M. Kytö, B. Ens, T. Piumsomboon, G. A. Lee, and M. Billinghurst, "Pinpointing: Precise head- and eye-based target selection for augmented reality," in *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, ser. CHI '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 1–14. [Online]. Available: https://doi.org/10.1145/3173574.3173655

[5] I. M. Neuhann, B. A. Lege, M. Bauer, J. M. Hassel, A. Hilger, and T. F. Neuhann, "Static and dynamic rotational eye tracking during lasik treatment of myopic astigmatism with the zyoptix laser platform and advanced control eye tracker," *Journal of Refractive Surgery*, vol. 26, no. 1, pp. 17–27, 2010. [Online]. Available: https://journals.healio.com/doi/abs/10.3928/1081597X-20101215-03

[6] M. L. Mele and S. Federici, "Gaze and eye-tracking solutions for psychological research," *Cognitive Processing*, vol. 13, no. S1, pp. 261–265, 7 2012. [Online]. Available: https://doi.org/10.1007/s10339-012-0499-z

[7] A. N. Angelopoulos, J. N. Martel, A. P. Kohli, J. Conradt, and G. Wetzstein, "Event-based near-eye gaze tracking beyond 10,000 hz," *IEEE Transactions on Visualization and Computer Graphics*, vol. 27, no. 5, pp. 2577–2586, 2021.

[8] G. Gallego, T. Delbrück, G. Orchard, C. Bartolozzi, B. Taba, A. Censi, S. Leutenegger, A. J. Davison, J. Conradt, K. Daniilidis, and D. Scaramuzza, "Event-based vision: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 1, pp. 154–180, 2022.

[9] C. Ryan, B. O. Sullivan, A. Elrasad, J. Lemley, P. Kielty, C. Posch, and E. Perot, "Real-time face eye tracking and blink detection using event cameras," 2020.

[10] Q. Chen, Z. Wang, S.-C. Liu, and C. Gao, "3et: Efficient event-based eye tracking using a change-based convlstm network," 2023.

[11] N. Li, A. Bhat, and A. Raychowdhury, "E-track: Eye tracking with event camera for extended reality (xr) applications," in *2023 IEEE 5th International Conference on Artificial Intelligence Circuits and Systems (AICAS)*, 2023, pp. 1–5.

[12] R. Alessi, "Eye Tracking Algorithm Based on An Event-Driven Camera," 11 2023. [Online]. Available: https://unire.unige.it/handle/123456789/6818

[13] K. Grauman, M. Betke, J. Gips, and G. Bradski, "Communication via eye blinks - detection and duration analysis in real time," in *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, vol. 1, 2001, pp. I–I.

[14] T. Morris, P. Blenkhorn, and F. Zaidi, "Blink detection for real-time eye tracking," *Journal of Network and Computer Applications*, vol. 25, no. 2, pp. 129–143, 2002. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S108480450290130X

[15] T. Stoffregen, H. Daraei, C. Robinson, and A. Fix, "Event-based kilohertz eye tracking using coded differential lighting," in *2022 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2022, pp. 3937–3945.

[16] J. Sigut and S.-A. Sidha, "Iris center corneal reflection method for gaze tracking using visible light," *IEEE Transactions on Biomedical Engineering*, vol. 58, no. 2, pp. 411–419, 2011.

[17] F. Zhao, H. Wang, and S. Yan, "Eye glint detection and location algorithm in eye tracking," in *2016 IEEE/CIC International Conference on Communications in China (ICCC)*, 2016, pp. 1–5.

[18] E. Demjén, V. Abosi, and Z. Tomori, "Eye tracking using artificial neural networks for human computer interaction," *Physiological research*, vol. 60, no. 5, p. 841, 2011.

[19] F. Lu, Y. Sugano, T. Okabe, and Y. Sato, "Inferring human gaze from appearance via adaptive linear regression," in *2011 International Conference on Computer Vision*. IEEE, 2011, pp. 153–160.

[20] K. Alberto Funes Mora and J.-M. Odobez, "Geometric generative gaze estimation (g3e) for remote rgb-d cameras," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 1773–1780.

[21] K. TAMURA, K. HASHIMOTO, and Y. AOKI, "Head pose-invariant eyelid and iris tracking method," *Electronics and Communications in Japan*, vol. 99, no. 2, pp. 19–27, 2016. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/ecj.11776

[22] S.-J. Baek, K.-A. Choi, C. Ma, Y.-H. Kim, and S.-J. Ko, "Eyeball model-based iris center localization for visible image-based eye-gaze tracking systems," *IEEE Transactions on Consumer Electronics*, vol. 59, no. 2, pp. 415–421, 2013.

[23] M. Vester-Christensen, D. Leimberg, B. K. Ersbøll, and L. K. Hansen, "Deformable models for eye tracking," in *Den 14. Danske Konference i Mønstergenkendelse og Billedanalyse*, aug 2005. [Online]. Available: http://www2.compute.dtu.dk/pubdb/pubs/3900-full.html

[24] L. Gava, M. Monforte, M. Iacono, C. Bartolozzi, and A. Glover, "Puck: Parallel surface and convolution-kernel tracking for event-based cameras," 2022.

[25] J.-G. Wang, E. Sung, and R. Venkateswarlu, "Estimating the eye gaze from one eye," *Computer Vision and Image Understanding*, vol. 98, no. 1, pp. 83–103, 2005, special Issue on Eye Detection and Tracking. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1077314204001134