

Event-Based Near-Eye Gaze Tracking Beyond 10,000 Hz

Anastasios N. Angelopoulos*, Julien N.P. Martel*, Amit P. Kohli, Jörg Conradt, Gordon Wetzstein

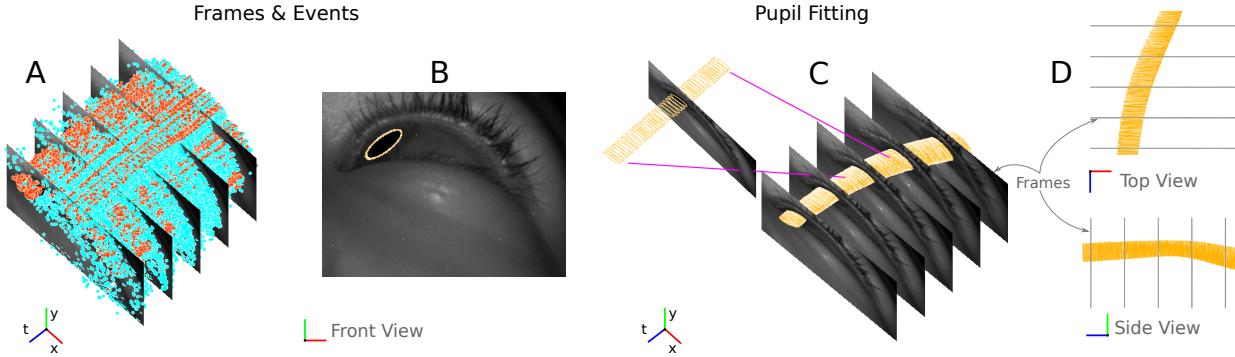


Fig. 1. Input and output of our system. The inputs, shown in plot A, are frames recorded at a fixed sampling rate and events, asynchronously sampling the eye motion at high speed. Frames and events are captured by the same sensor, and event polarity is color coded in blue (+) or red (-). We output a gaze point, computed from our estimate of the pupil, shown in yellow as seen from several perspectives in plots B, C, and D (x and y are the columns and rows of the sensor, and t is time). Events continuously trigger between frames, allowing pupil estimation much faster than the frame rate. Every pupil estimate yields a yellow circle. These estimates are so frequent that they form an almost continuous tubular structure outlining the pupil's movement through time in plots C and D.

Abstract—The cameras in modern gaze-tracking systems suffer from fundamental bandwidth and power limitations, constraining data acquisition speed to 300 Hz realistically. This obstructs the use of mobile eye trackers to perform, e.g., low latency predictive rendering, or to study quick and subtle eye motions like microsaccades using head-mounted devices in the wild. Here, we propose a hybrid frame-event-based near-eye gaze tracking system offering update rates beyond 10,000 Hz with an accuracy that matches that of high-end desktop-mounted commercial trackers when evaluated in the same conditions. Our system, previewed in Figure 1, builds on emerging event cameras that simultaneously acquire regularly sampled frames and adaptively sampled events. We develop an online 2D pupil fitting method that updates a parametric model every one or few events. Moreover, we propose a polynomial regressor for estimating the point of gaze from the parametric pupil model in real time. Using the first event-based gaze dataset, we demonstrate that our system achieves accuracies of 0.45°–1.75° for fields of view from 45° to 98°. With this technology, we hope to enable a new generation of ultra-low-latency gaze-contingent rendering and display techniques for virtual and augmented reality.

Index Terms—Event-based camera, Eye tracking, Augmented and virtual reality.

1 INTRODUCTION

Gaze tracking is the process of estimating where a person is looking, usually with a camera. It enables dozens of applications in augmented and virtual reality (AR/VR), such as foveated or physiologically accurate rendering [22, 29], interactive programs that respond to eye movement by allowing the user to select a target with their eyes [32], and so forth. Such applications, and others like laser eye surgery [48], benefit from fast and accurate tracking of an eye that essentially fills the sensor field of view ('near-eye' tracking). Ideally, to meet the battery and processing constraints of mobile headsets, eye trackers should also conserve power and compute.

However, cameras force a tradeoff between resolution, framerate, and power, since every pixel costs energy and bandwidth to acquire, communicate, and process. Thus, the accuracy and latency of an eye tracking system are often in tension. High-end eye tracking systems resolve this using high-speed cameras, bespoke protocols, and customized

readout interfaces to maximize bandwidth. Consequently, they are large and power hungry. These complexities are unavoidable because of the sheer volume of data generated by high-speed, high-resolution cameras. But in near-eye gaze tracking, most of this data is redundant. Only the pupil moves, while most of the image does not change.

Dynamic vision sensors (DVS) overcome these limitations by adaptively sampling when the eye moves. DVS pixels separately sample when the instantaneous change in their incident irradiance exceeds a threshold. The result is a stream of pixel-by-pixel timestamped packets signaling changes, called *events*. Events contain the pixel location, the time of sampling, as well as the sign of the change. In near-eye tracking, motion is sparse in time *and* space. Events thus use bandwidth more efficiently than frames, because only relevant information is sampled and processed. This improves speed and power consumption.

Here, we report the first real-time event-based eye tracking system, described in Figure 2. The sensor is placed close to the user's eye such that the eye nearly covers its field of view. For this reason, we refer to this system as being "near-eye", although our data is collected in a desktop setting. We discuss a miniature prototype in Section 6. At the core of our system is the update of a parametric representation of the pupil at the event rate. This is fed to a polynomial regressor, also evaluated on an event-by-event basis, that maps this internal parametric representation to a *gaze vector* (the 3D direction of gaze). Along with events, we also use frames captured at low rates (15–20Hz) by the same sensor. The frames anchor our pupil tracking system with traditional pupil-detection algorithms, while the events allow it to update the

* Anastasios N. Angelopoulos and Amit P. Kohli are with the University of California Berkeley. E-mail: {angelopoulos,apkohli}@berkeley.edu,
• Julien N.P. Martel and Gordon Wetzstein are with Stanford University, E-mail: {jnmartel,gordonwz}@stanford.edu,
• Jörg Conradt is with the KTH Royal Institute of Technology. E-mail: jconradt@kth.se.
• * denotes equal contribution
Project website: <http://angelopoulos.ai/blog/posts/ebv-eye>

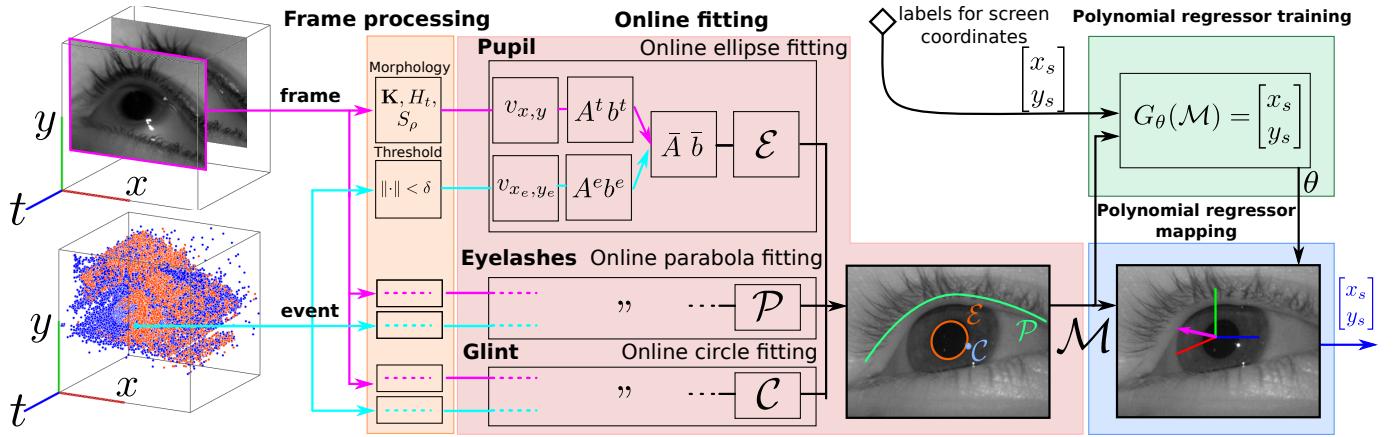


Fig. 2. A flow diagram of our system. Both frames and events are inputs to our system. A first stage preprocesses them separately before both streams are combined to update the eye model $\mathcal{M} = \{\mathcal{E}, \mathcal{P}, \mathcal{C}\}$. Here, \mathcal{E} and \mathcal{P} are tuples of the parameters of quadratic functions defining the limbus and eyelid, and \mathcal{C} is a tuple of the two center coordinates (see Section 3 for further details). The model fitting is performed online, in real-time at the arrival rate of frames (low, 25 Hz) and events (high, up to 100 kHz). To output a gaze vector, \mathcal{M} is fed to a 5th-order polynomial regressor that can be evaluated at event rate. This regressor is trained via a calibration procedure similar to EyeLink’s.

pupil’s location at high-speed. In our system, gaze vectors can be queried at an estimated rate equivalent to 10,000 Hz or more. This is an improvement of $>10\times$ over high-end, desktop-mounted devices (see Table 1) using a small, low power sensor¹. The following list summarizes our contributions.

- We introduce the first hybrid event-based eye tracking system and demonstrate a binocular prototype.
- We develop a model-based eye tracking algorithm, functioning at the event rate and implement it in a real-time system.
- We capture a binocular dataset of events and frames from 24 subjects performing saccadic motions and smooth pursuits.

1.1 Is 10kHz gaze tracking useful?

The extreme speed of the eye tracking technology we introduce raises a natural question: why should we care about an extremely fast eye-tracking system? The answer has several layers. Firstly, the human eye moves quickly during certain motions, at speeds sometimes exceeding $300^\circ/s$ [58]. This fact justifies the use of very fast sampling, in the kHz range, in order to capture small, fast eye movements as they initiate and track their trajectories. Furthermore, the eye’s acceleration regularly reaches astronomical values—e.g. $24,000^\circ/s^2$ —meaning the eye muscles are capable of making forceful and high frequency movements [4]. A simple Nyquist argument then motivates sampling in the 10 kHz range to fully capture the eye’s motion. In particular, a sensor with a high peak sampling rate could detect when the eye is moving more quickly, characterize that motion with higher fidelity, and perhaps even enable predictive approaches to eye tracking. For applications like e-sports, where latency has a pernicious negative impact [26, 27], fast event-based predictive rendering could be a *game-changer*.

The adaptive sampling inherent to event-based cameras also helps speak to the utility of speed. Our system does not always run at 10 kHz, although it is capable of sustaining that rate. It only does so when the underlying motion of the eye generates many events. Rather than deciding a-priori to sample at 10 kHz, this rate emerged from the motion of the eye, a natural justification that this speed may be useful. This sampling approach allows the sensor to be much more power efficient than a bespoke camera system like the EyeLink 1000, which may not be suitable for an AR/VR headset due to power constraints.

¹A Pregius line 40FPS SONY sensor consumes 300mW–2W [54] vs. 50mW–0.9W for a DAVIS-346 [25].

system	update rate (Hz)	accuracy ($^\circ$)
Pupil Labs [1]	200	~ 1
Tobii [3]	120	0.5–1.1
EyeLink [2, 16]	1,000	~ 0.5
Ours	> 10,000	0.45–1.75

Table 1. Overview of existing eye tracking systems. Our results can be directly compared with EyeLink because we use the same protocol [17]. Our system’s accuracy is 0.45° within the same field of view as EyeLink, and speed is $10\times$ faster.

2 RELATED WORK

Event cameras date back to the neuromorphic Silicon Retina introduced by the seminal works of Mahowald and Mead [41, 43], and have since advanced to a mature technology [12, 13, 38, 50]. In the intervening time, the utility of these high-speed sensors has been successfully exploited in a variety of application scenarios, including object and action tracking [44], combined frame and event based object tracking [39], visual odometry [52], 6-DOF pose tracking [46], 3D reconstruction [42], SLAM [11, 61], and hand tracking [33]. See the review by Gallego et al. for a dedicated overview of the diverse work in the event-based vision community [20]. **Hybrid frame-event approaches** The fusion of events and frames has been explored in different works such as [21, 40]. In those approaches, direct, absolute photometric correlates acquired at low rate (the frames) initialize or enhance the estimation of a quantity tracked at high update rates (by the events). Similarly to visual-inertial odometry in which a high-rate, potentially drifting, sensor such as an inertial measurement unit (IMU) is corrected by a lower rate sensor that provides absolute anchor points (such as visual features), these fusion approaches are potentially more robust [35].

Early approaches to eye tracking include electro-oculography, search coils, and other invasive methods [64]. Camera-based eye tracking has evolved from tracking Purkinje reflections [8, 9] to model-based approaches that extract parameterizations of the eye from frames [37, 56, 60]. The model extraction, or *pupil fitting*, can be decoupled from regressing the point of gaze. Algorithms for pupil fitting and gaze estimation are reviewed by Morimoto et al. [45] and Duchowski et al. [15]. However, unlike in event-based vision, these methods are rate-limited by camera frames. A number of eye tracking strategies do not involve cameras, including photodiode base limbus trackers [57], display embedded trackers for near-eye displays [59], and LED based trackers [5, 36]. We do not focus our attention on these strategies, although they are promising.

Appearance-based trackers directly estimate points of gaze from camera frames, often using neural networks [6, 31, 51, 66]. A number of datasets have emerged over the past five years, many of which



Fig. 3. Events are generated by the pupil as it moves. The events on the right are generated by the eye milliseconds after the frame shown on the left, during a downward saccadic eye movement. The bottom generates negative (red) events and the top generates positive (blue) ones, since the pupil induces a negative contrast change in its direction of motion. Each event in the right panel is received sequentially as a separate packet.

leverage synthetic data, including: GazeCapture [31], UT Multi-view [55], SynthesEyes [63], UnityEyes [62], MPIIGaze [65], and, most recently, NVGaze [28]. These datasets comprise millions of synthesized and real eye images. However, there is no dataset for event-based eye tracking.

More recent gaze-tracking literature generally focuses on one of two problems: gaze-tracking *in the wild*, i.e. on full-face images of users without infrared (IR) illumination, such as from a laptop webcam [24]; or near-eye IR-illuminated gaze-tracking for use in controlled environments like an AR/VR headset. The review by Koulieris et al. details these modern topics [30]. Note that we do not focus on eye-tracking in the wild, although a deep learning model like NVGaze [28] used in the frame-based portion of our system might enable us to do so. Hence, our work falls in the second category, and we focus on eye tracking in controlled, near-eye settings. We evaluate against commercial eye-tracking systems in Table 1 using similar conditions and protocols. Our eye tracker, like all other infrared near-eye trackers, is not meant to be used where there will be large variations in head pose, reflections, or background. It is geared towards AR/VR or biomedical settings where high framerates are desirable (hence why the EyeLink 1000 samples at 1-2 kHz). Consequently, the accuracy and update rate of our near-eye tracker with respect to controlled data is the proper evaluation metric. No portable eye tracker achieves a framerate over roughly 200 Hz; but using event-cameras, we achieve >10,000 Hz with comparable accuracy to the gold standard, desktop mounted EyeLink device. Compared to the EyeLink, our system is similarly accurate and an order of magnitude faster. Compared to mobile systems, our system is similarly accurate and two orders of magnitude faster.

The work closest to ours is the course project by Gisler [10], who suggests the idea of using a DVS for eye-tracking. However, the accuracy of their system is limited to “*a third of the size of a 1024 × 768 screen*” and they did not demonstrate real-time performance. To the best of our knowledge, our work is the first to implement such a system, to develop a practical algorithmic framework that includes an online pupil fitting procedure, and to capture an event-based eye tracking dataset, which will be released to the public.

3 SYSTEM AND METHODS

A DVS pixel triggers independently from its neighbors when it sees a contrast change. This is called an event. Events are like pixels in a difference image, except they are received separately at the time they were generated. They carry no notion of a frame; events arrive as a sparse stream of data, as illustrated in Figure 4. This stream consists only of the timestamps, signs (± 1 for positive or negative contrast changes), and locations of the triggered pixels. Sparsity comes with many benefits, like speed and processing efficiency. However, previously developed frame-based eye tracking methods like binarization and dark/bright pupil tracking cannot be used on such data. Our goal in this section is to develop a technique which can efficiently process this new data stream while still leveraging the large body of existing work in frame-based tracking.

To be precise, we built a hybrid event-frame-based gaze tracker combining the low latency of events with the proven robustness of frame-based methods. Our system consists of two distinct, concurrent processing stages. First, it fits a *2D eye model* from the event and frame data. Second, it maps the estimated eye model parameters to a *3D gaze vector* representing the direction the user is looking, or alternatively, a pixel on a screen at fixed distance (both representations have two free parameters). An overview of our algorithmic framework is illustrated in Figure 2.

3.1 2D Model Fitting

We start by defining a parametric *eye model*, with parameters \mathcal{M} . Whenever a frame or event is received from the sensor, these parameters are updated. Since frames are produced at a constant rate, they update the model independent of scene dynamics. But events only occur during eye motion, and update it at high frequency. Our method fuses these synchronous and asynchronous streams.

Eye model Motivated by Tian et al. [56], our eye model consists of an *ellipse representing the pupil* with parameters $\mathcal{E} = (a, h, b, g, f)^\top \in \mathbb{R}^5$, a *parabola representing the eyelid* with parameters $\mathcal{P} = (u, v, w)^\top \in \mathbb{R}^3$, and a *circle representing the glint* (the reflection of the IR light source off of the user's eyeball) with parameters $\mathcal{C} = (r, c)^\top \in \mathbb{R}^2$. The eye is thus fully parameterized by the 11 coefficients in $\mathcal{M} = \{\mathcal{E}, \mathcal{P}, \mathcal{C}\}$.

The parameters $\mathcal{E}, \mathcal{P}, \mathcal{C}$ are fit separately. Ellipses, parabolas, and circles are expressed canonically as quadrics, so they can be asynchronously estimated using the same method. In the following, we detail the updates for the fitting of \mathcal{E} (see supplement for \mathcal{P} and \mathcal{C}). The task ahead is to estimate \mathcal{E} from a set of “candidate ellipse points” \mathcal{D} in the image plane that we believe lie on the edge of the pupil.

Parameterizing the pupil with an ellipse The locations of points $\vec{p} = (x, y)^\top$ on the ellipse representing the pupil in the image plane satisfy the quadric equation:

$$E_{\mathcal{E}}(\vec{p}) = E_{\mathcal{E}}(x, y) = 0 \quad (1)$$

$$\text{with } E_{\mathcal{E}}(x, y) = ax^2 + hxy + by^2 + gx + fy + d$$

We set $d = -1$ for convenience as it is an arbitrary scaling factor corresponding to the offset of the plane intersecting the conic defined by $\mathcal{E} = (a, h, b, g, f)$.

For each frame, we classify pixels near the edge of the pupil in an image as candidate points \mathcal{D}_{img} . Events near the edge of the pupil are considered candidate points \mathcal{D}_{evt} . Thus, the model of the ellipse is ultimately updated by the set of points $\mathcal{D} = \mathcal{D}_{\text{evt}} \cup \mathcal{D}_{\text{img}}$.

Receiving a frame Under off-axis IR illumination, the pupil appears as a dark blob in the frame (see Fig. 5). By binarizing the greyscale frame I using a constant threshold θ , removing noise on the resulting image using morphological opening, and applying an edge detector, we identified the candidate points:

$$\mathcal{D}_{\text{img}} = \{(x, y) \mid \mathbf{K}(H_\theta(I) \circ S_\sigma)(x, y) = 1\}, \quad (2)$$

where H_θ is the unit step function shifted by θ used for thresholding; \circ denotes morphological opening; S_σ is its structuring element, a discretized circle parameterized by its radius σ ; and \mathbf{K} is a binary edge detection function. We found that recovering candidate ellipse points with these simple operations worked sufficiently well. However, one could use any state-of-the-art frame-based pupil tracking algorithm outputting a set of candidate points to replace this stage of our system, such as PuReST [53], ExCuSe [18], and Else [19].

Receiving an event Events only contribute to the fitting of \mathcal{E} when they are δ -close to the border of the last estimated ellipse,

$$\mathcal{D}_{\text{evt}} = \{(x, y) \mid \vec{p} = (x, y)^\top, \|P_{\mathcal{E}}(\vec{p}) - \vec{p}\|_2 < \delta\}, \quad (3)$$

where $P_{\mathcal{E}}(\vec{p})$ is the projection of a point \vec{p} on the ellipse — this is step 7 of Algorithm 1.

The projection operator $P_{\mathcal{E}}(\vec{p})$ amounts to solving a system of two equations (one linear and one quadratic) in the specific case of our ellipse parameterization.

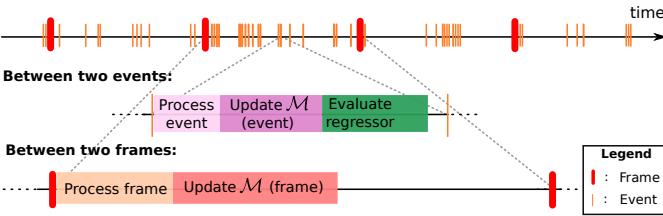


Fig. 4. The processing flow and time taken by the different stages in our system. Our system allows events and frames to be processed concurrently and update the same underlying model. Events and frames are shown in time (top row). We also illustrate the operations happening sequentially and concurrently on events (middle) and frames (bottom).

Our method can perform updates of \mathcal{M} on an event-by-event basis, in which case \mathcal{D}_{evt} is a singleton containing a single event. As we shall see in our experiments, the robustness of our method benefits from considering more than one event per update, in which case \mathcal{D}_{evt} contains more than one event.

Fitting the ellipse from images and events We fit our ellipse model (and similarly our parabola and circle models) using least squares. The data points \mathcal{D}_{img} coming from the same frame can be thought of as having been generated synchronously, allowing us to fit the model to the data as a batch:

$$\mathcal{E}^* = \arg \min_{\mathcal{E}} \sum_{(x,y) \in \mathcal{D}_{\text{img}}} E_{\mathcal{E}}(x,y)^2 \quad (4)$$

whose solution is simply $\mathcal{E}^* = A^{-1} b$ with

$$A = \sum_{(x,y) \in \mathcal{D}_{\text{img}}} v_{x,y} v_{x,y}^T, \quad b = \sum_{(x,y) \in \mathcal{D}_{\text{img}}} v_{x,y} \quad (5)$$

and

$$v_{x,y} = (x^2, xy, y^2, x, y)^T \quad (6)$$

Generally, we start with an initial estimate of the ellipse's parameters and then wish to update it with new sensor information (such as a received frame or event). We can do so because in Equation (11), A and b are sums and can thus be updated “online.” Practically, we store a matrix \bar{A} and a vector \bar{b} , that are both updated upon the reception of new candidate points by “summing them in.” More formally, when a set of candidate points \mathcal{D}^t arrives at time t , it is used to produce a matrix A^t and a vector b^t according to Equation (11). A^t and b^t can then be blended with a matrix \bar{A}^t and a vector \bar{b}^t representing and storing the “current” state of the fit.

$$\begin{aligned} \bar{A}^{t+1} &= \gamma \bar{A}^t + (1 - \gamma) A^t, \\ \bar{b}^{t+1} &= \gamma \bar{b}^t + (1 - \gamma) b^t, \text{ with } \gamma \in [0, 1] \end{aligned} \quad (7)$$

This method has the advantage of storing a single small 5×5 matrix and a 5-dimensional vector, and blends information in time in a principled way. Our method is reminiscent of reweighted least-squares (RLS) with the loss of each candidate-point geometrically decayed by γ . However, it is not equivalent due to the thresholding operation. To prove this fact, in the setting of Equations 4–7, number $v_{x,y}^{(1)}, \dots, v_{x,y}^{(t+1)}$. RLS has a solution of the form $R_t^{-1} r_t$ where: $R_t = \sum_{i=1}^t a_i v_{x,y}^{(i)} v_{x,y}^{(i)T}$, $r_t = \sum_{i=1}^t a_i v^{(i)}$ and $a^{(i)}$ are constant discount factors. Given \bar{A}^t and \bar{b}^t , we want \bar{A}^{t+1} and \bar{b}^{t+1} to have the RWLS form. But $A^{t+1} = \mathbf{1}(\text{proj}_{\mathcal{E}_t}((x_{t+1}, y_{t+1})) \leq t)(1 - \gamma)(v_{x,y}^{(t+1)} v_{x,y}^{(t+1)T} - \bar{A}^t) + \bar{A}^t$, since $v_{x,y}^{(t+1)}$ is only included if it is close enough to the last ellipse. The definition of \bar{b}^{t+1} is analogous. Therefore Algorithm 1 is RLS if and only if $\mathbf{1}(\text{proj}_{\mathcal{E}_t}((x_{t+1}, y_{t+1})) \leq t)$ is constant. One can interpret δ and γ as spatio-temporal regularization of our method; for example setting $\gamma = 0$ will throw out all old candidate points (leading to degenerate ellipses because A becomes rank-1) while setting $\gamma = 1$ will stop all updates.

Algorithm 1 Online fitting of \mathcal{E} from events and images

```

1:  $\bar{A} = \text{Id}_{5 \times 5}, \bar{b} = \vec{0}_5$                                 ▷ Init.  $\bar{A}$  and  $\bar{b}$ 
2:  $\bar{A}_{\text{inv.}} = \bar{A}^{-1} = \text{Id}_{5 \times 5}$                       ▷ If using SMW init.  $\bar{A}_{\text{inv.}}$ 
3: while we are receiving data  $d$  do
4:   if  $d$  is a frame  $I^t$  then
5:      $\mathcal{D}_{\text{img.}} = \{(x, y) \mid \mathbf{K}(H_\theta(I^t) \circ S_p)(x, y) = 1$ 
6:   else if  $d$  is an event  $E = (\vec{p} = (x, y)^T, p, t)$  then
7:      $\mathcal{D}_{\text{evt.}} \leftarrow \mathcal{D}_{\text{evt.}} \cup \vec{p}$  if  $\|\mathcal{P}_{\mathcal{E}}(\vec{p}) - \vec{p}\|_2 < \delta$ 
8:   for all  $(x, y) \in \mathcal{D} = \mathcal{D}_{\text{evt.}} \cup \mathcal{D}_{\text{img.}}$  do
9:      $v_{x,y}^T = (x^2, xy, y^2, x, y)^T$                                 ▷ Eq. (13)
10:     $A = \sum_{(x,y) \in \mathcal{D}} v_{x,y} v_{x,y}^T$                                 ▷ Eq. (11)
11:     $b = \sum_{(x,y) \in \mathcal{D}} v_{x,y}$ 
12:    if  $|\mathcal{D}| > 1$  then                                         ▷ Batch update (frame/acc. evts.)
13:       $\bar{A} \leftarrow \gamma \bar{A} + (1 - \gamma) A$                                 ▷ Eq. (7)
14:       $\bar{b} \leftarrow \gamma \bar{b} + (1 - \gamma) b$ 
15:       $\bar{A}_{\text{inv.}} \leftarrow \bar{A}^{-1}$                                          ▷ Full-rank inversion
16:       $\mathcal{E} \leftarrow \bar{A}_{\text{inv.}} \bar{b}$ 
17:    else                                                 ▷ (Event-based update)
18:       $\bar{A}_{\text{inv.}} \leftarrow \frac{1}{\gamma} \bar{A}_{\text{inv.}} - \frac{1 - \gamma}{\gamma} \frac{\bar{A}_{\text{inv.}} A \bar{A}_{\text{inv.}}}{\gamma + (1 - \gamma) v_{x,y}^T \bar{A}_{\text{inv.}} v_{x,y}}$  ▷ SMW
19:       $\bar{b} \leftarrow \gamma' \bar{b} + (1 - \gamma') b$ 
20:       $\mathcal{E} \leftarrow \bar{A}_{\text{inv.}} \bar{b}$ 

```

In the case that \mathcal{D}^t comes from a frame, A^t and b^t can be directly calculated from (11) since A^t is usually full rank. In contrast, events arrive one at a time and asynchronously. Since our goal is to take advantage of the low-latency and high-time resolution of the event generation process, we should update \bar{A} and \bar{b} from $\mathcal{D}_{\text{evt.}}$, as often as every event. An event generates a single candidate point, but $v_{x,y}$ can nonetheless be computed using Equation (13). The corresponding A^t and b^t for that event are computed using Equation (11). Note that because A^t is rank-1, it is not invertible. This is not surprising, since \mathcal{E} has 5 parameters, therefore one needs 5 independent points to fit them. In case one aims at performing an update every N events, $|\mathcal{D}_{\text{evt.}}| = N$, and we can update \mathcal{E} batch-wise, similarly to a frame.

Again, applying Equation (7), we can update \bar{A} as $\bar{A}^{t+1} = \gamma' \bar{A}^t + (1 - \gamma') \bar{A}^t$, with $\gamma' \in [0, 1]$. After the reception of the first 5 events in a non-degenerate configuration, \bar{A}^t is rank-5 and can thus be inverted (given we keep blending in new information, it is generally invertible for the rest of time). Since $v_{x,y}$ and the blending of A and b are both easy to compute, these updates can be performed at the event rate in practice. But, updating \mathcal{E}^* eventwise (typically up to 200 times per millisecond during a saccade) also entails computing $(\bar{A}^t)^{-1}$ eventwise, which might be computationally infeasible to perform in real time. However, because every event generates an A^t that is rank-1, one can store $(\bar{A}^t)^{-1}$ and use the Sherman-Morrison-Woodbury (SMW) identity [23] to update it directly, online, after applying a small decay term to downweight old data in time. The fitting of the ellipse is summarized in Algorithm 1. Again, the fitting of \mathcal{P} and \mathcal{C} is analogous. This formulation and implementation of least squares is well suited for the fusion of both the event and frame streams: it is a natural online method that is agnostic to the synchronicity of the data.

3.2 Mapping the eye-model to a point of gaze

The output of our gaze tracker is the 2D screen coordinate the user is looking at, which we call the point of gaze. In the first stage of our system (Sec. 3.1), we fit the parameters $\mathcal{M} = \{\mathcal{E}, \mathcal{P}, \mathcal{C}\}$ of an eye model given incoming events and frames. We now discuss how we associate a point of gaze to those parameters.

The 2D screen coordinate position a user is looking at is denoted $(x_s, y_s)^T$. The problem is to find a mapping from \mathcal{M} to $(x_s, y_s)^T$. We could explicitly model and fit the relative poses of the camera, user eye, and screen along with the projection of the camera and transformation between screen and world coordinates. However, we adopt another approach, common in the gaze mapping literature [45] that consists of phenomenologically regressing the output $(x_s, y_s)^T$ from the pupil

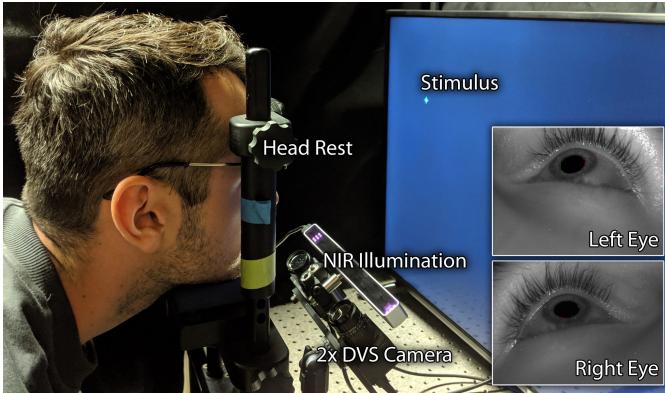


Fig. 5. The binocular eye tracking system setup used for evaluation.
We collected a dataset with binocular saccadic and smooth-pursuit data on 24 subjects looking at an 11×11 grid of fixation points over a $64 \times 96^\circ$ FoV. Altogether, we collected ~ 30 million events per subject per eye. Two DAVIS sensors and a near-infrared illumination source are mounted close to the user's head. The head is fixed by a head rest and the user observes a stimulus on the screen.

center (x_c, y_c) using two 5th order polynomial functions $G_{\theta^1}|_x$ and $G_{\theta^2}|_y$ (one for each coordinate):

$$G_\theta(x_c, y_c) = \begin{pmatrix} x_s \\ y_s \end{pmatrix} = \begin{pmatrix} G_{\theta^1}|_x(x_c, y_c) \\ G_{\theta^2}|_y(x_c, y_c) \end{pmatrix} \quad (8)$$

The simplicity of our model and polynomial regressor is a strength of our system. An advantage of using such a polynomial G_θ to map an eye model to screen coordinates is that it requires virtually no computation to evaluate: a few additions and multiplications. Hence, it is particularly well suited for event updates, even at high rates. Regressing the parameters θ of G_θ requires $((x_c, y_c), (x_s, y_s))$ input-output training pairs (the exact number depending on the exact degree of the polynomial) which are generally obtained during calibration by extracting the pupil center as in Section 3.1 and regressing against the known gaze point. Other simple regression strategies that we did not attempt, like Gaussian processes and random forests, may provide further benefits like uncertainty quantification. Although a more complex model such as NVGaze [28] may improve robustness and generalization, it requires orders of magnitude more computation and power and does not provide an accuracy advantage (see discussion).

4 DATASET

Data Acquisition Our setup is shown in Figure 5. It consists of two DAVIS346b (iniVation) sensors, imaging the right and left eye of a user placed at about 25 cm from each camera center. The user's head is fixed on an ophthalmic head rest and strapped during the experiment to prevent excessive slippage. The sensors are mounted with two 25 mm f/1.4 VIS-NIR C-mount lenses (EO-#67-715) equipped with two UV/VIS cut-off filters (EO-#89-834). The eyes are illuminated using the NIR illuminators of the Eye Tribe tracker. Both sensors are synchronized (their timestamps are aligned), and the 8-bit 346×260 px greyscale frames and events of both DAVIS346b sensors are recorded simultaneously. The exposure is set so as to maximize contrast in the frame, resulting in a frame rate of about 25 FPS. The stimuli are displayed on a 40 in diagonal, 1920×1080 px monitor (Sceptre 1080p X415BV_FSR), placed 40 cm away (standard reading distance) from the user [14, 34]. It is horizontally centered and vertically aligned so that a user looking straight roughly gazes at a point placed at a third from the top.

Dataset Characteristics Our dataset is, to the best of our knowledge, the first collected for gaze-tracking using event-based vision sensors. It was recorded on 24 subjects and consists of two experiments corresponding to two different types of eye-motion: random saccades and smooth pursuit. The stimulus is a 40×40 px green cross centered

on a 20 px diameter disk presented against a black background. In our setup the monitor spans a field of view (FoV) of $64 \times 96^\circ$.

In the first experiment, users were asked to fixate on the stimulus randomly displayed at one of 121 different locations (corresponding to an 11×11 grid on the monitor) for 1.5 s each. All locations are presented once, and the random sequence was the same for all users. The grid is visualized in the bottom row of Figure 6. In the second experiment, users were asked to fixate on the stimulus, which moved smoothly along a predictable square-wave trajectory starting at the top of the screen and moving towards the bottom while spanning the whole screen horizontally with a vertical period of 150 px. This trajectory is the black dotted line in the top row of Figure 6. Although we only explicitly induced saccadic and smooth pursuit motions, our dataset contains rich eye motions including microsaccades and tremor. As one example, carefully parse the top row of Figure 7, and notice that after every saccade (large spike in events) there is a corrective microsaccade (small spike in events). Careful inspection of our public dataset will reveal detailed information about such subtle eye motions (e.g., the time between a saccade and the subsequent microsaccade).

5 RESULTS

Calibration (i.e. defining G_θ and estimating θ) is addressed in Section 3 and with more details in Supplement S2 and S4. Before using our eye tracker, a user looks at a set of “calibration points” whose coordinates in screen space are known. The pupil position in camera space is then extracted for each point, and a second-order polynomial is regressed mapping the pupil center to the screen coordinates: this is the gaze point. Anytime we report accuracy or precision results for a particular FoV, we only calibrate on half the points in that FoV (e.g. odd indexes). Then, we report results on the full set of points. For a similar FoV to EyeLink’s, the center $20^\circ \times 40^\circ$ FoV, this means we are testing on 16 points and calibrating on 8. EyeLink uses a 12 point calibration procedure [17] which supposedly enhances accuracy. Our results can thus be readily compared to the commercial gold standard.

Assessing Update Rates Our system can operate in real-time and update the pupil fit on an event-per-event basis; a conservative estimate of the peak update rate of our system is 10,000 Hz. We can achieve such a rate because saccadic motions induce hundreds of thousands of events due to the high-contrast edge between the pupil and the iris. These events can thus be used to “track” the pupil between two frames (see Figs. 1,7). In contrast, when the eye is still, very few events are produced.

We calculate the update rate as follows: we first estimate, in a conservative way, the amount of events per second for a saccade ρ . Figure 7 shows a saccade typically induces more than $\rho = 200 \text{ evts.ms}^{-1}$. Second, we calculate the optimal number of events per fit N to produce a robust (smooth) pupil position estimate: this is $N^* = 20$ according to Figure 8 in which we have performed an experiment varying the number of events per fit. This yields the conservative update rate of our system, $R = \frac{\rho}{N^*}$ which is $R = \frac{200 \cdot 10^{-3}}{20} = 10 \text{ kHz}$. Our system can sustain an update rate of 10,000 Hz or above indefinitely, but this is not desirable because when the eye is still, no updating is required. Our event-driven update rate therefore only samples quickly when the motion of the eye requires it. In other words, there would be no speed advantage to using a frame-based system running at = 10 kHz.

The number of events used to perform a fit is the number of events accumulated in \mathcal{D} before solving for \mathcal{E} (in Algorithm 1). Figure 7 illustrates the use of different values of N : we plot the fitted pupil center coordinates in image space for a random subject performing the random saccade experiment. As expected, when every event is used to update \mathcal{E} , the update rate is very high ($N = 1, \rho = 200 \text{ evts.s}^{-1}$ thus $R = 200 \text{ kHz}$) but the algorithm is not robust to series of noisy events which cause the fit to change drastically and reach an unrecoverable state until the next frame corrects it. Moreover, fitting for every event causes us to make wasteful updates to \mathcal{E} even when the eye is not really moving and events are just noise. In the opposite case, where we consider N to be hundreds, our data is desirably sparse and very robust (even to large perturbations such as blinks), but it does not smoothly

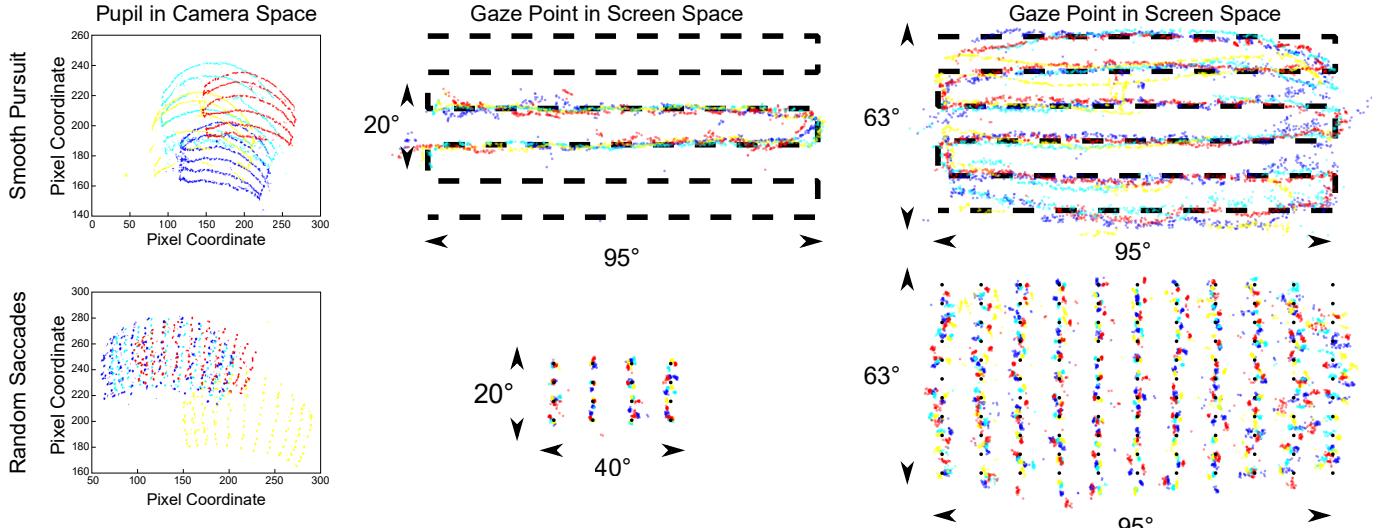


Fig. 6. Fitted pupil locations and gaze point estimates for smooth pursuit motion and random saccadic motion are shown for four different users in different colors. The figure is organized into grids; the first row plots smooth pursuit data and the second row plots random saccadic data. The first column shows the extracted pupil center in camera image space, and the second/third columns shows the gaze point in screen space for a small/large field of view, with the ground truth locations indicated in black. The average visual angle accuracy in a 47° FoV for smooth pursuit data is 1° (top center) and it is 3.9° for the entire 113° FoV (top right).

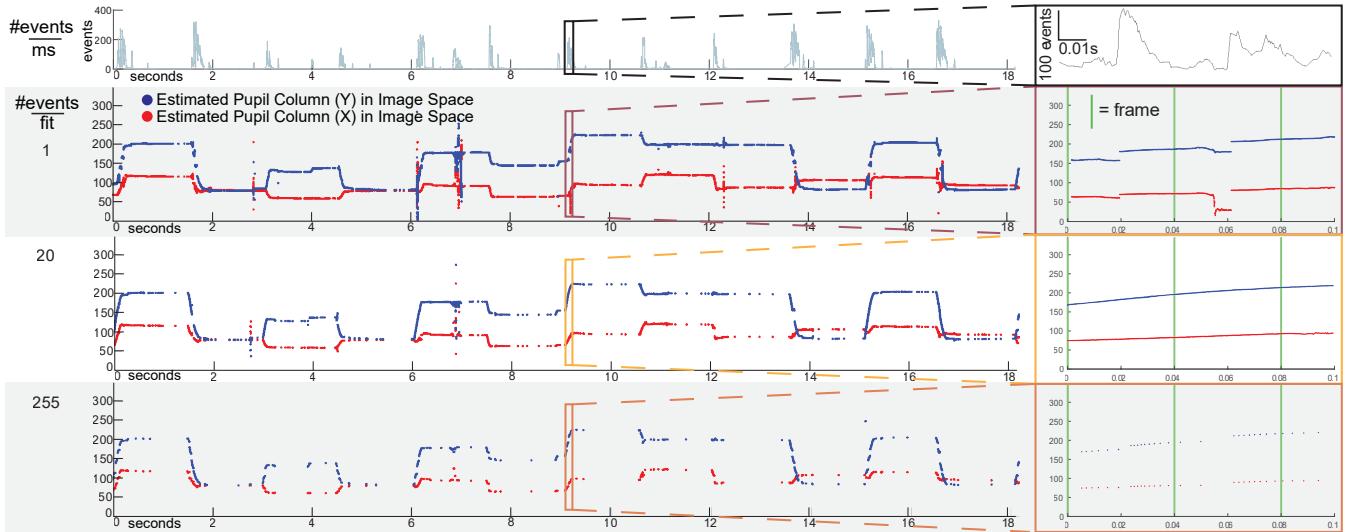


Fig. 7. The X-Y coordinates of the fitted pupil over time. Our system can update the pupil's location as fast as every event (row 2), but by considering more events, we can improve robustness and sparsity (row 3). Updating too slowly, however, harms robustness again (row 4). In the top row, a plot of the number of events per millisecond is shown for one user. Then, from top to bottom, different plots of the X (in red) and Y (blue) coordinates of the pupil position tracks are shown with an increasing number of events used for each fit. One pupil position is obtained per fit. Hence, fewer fits are produced in the lower plots (as more events are used per fit) than in the upper plots. The number of fits also varies with the number of events produced per millisecond (top plot). We find 20 events per fit to give a good trade-off between the speed of our system (the maximal number of fits per second we can obtain) and the robustness of the fit. The 10 kHz peak update rate we report is calculated for 20 events and based on the typical 200 events we observe per millisecond during a saccade. The "glitches" at 3 and 7 seconds are blinks, from which the system recovers at the next frame.

follow a saccade between frames. This exposes an inherent tradeoff in our system between the smoothness, sparsity, and update rate.

In order to find an optimal value for this tradeoff, Figure 8 plots a measure of smoothness against the number of events per fit for a given subject. The quantitative measure of smoothness we use is the inverse norm of the concatenated 1-forward differences of the X and Y coordinates. It shows that a clear optimum is obtained for this measure of smoothness using $N = 20$ events per fit. Indeed, this is also confirmed visually in the middle row of Figure 7: this parameter has a good balance between sparsity, robustness, and speed.

Pupil Tracking Evaluations To assess the effectiveness of event-based pupil tracking, we plot the intersection over union (IOU) and error in center estimation as histograms in Figure 9 over our entire dataset. For two binary matrices x and y , IOU can be expressed as

$$IOU(x,y) = \frac{\sum x \vee y}{\sum x \wedge y},$$

where the logical operations are performed elementwise. In our case, $x_{i,j} = 1$ if the pixel (i, j) lies within the event-based pupil estimate. The matrix $y_{i,j}$ corresponds similarly to the ground truth pupil extracted

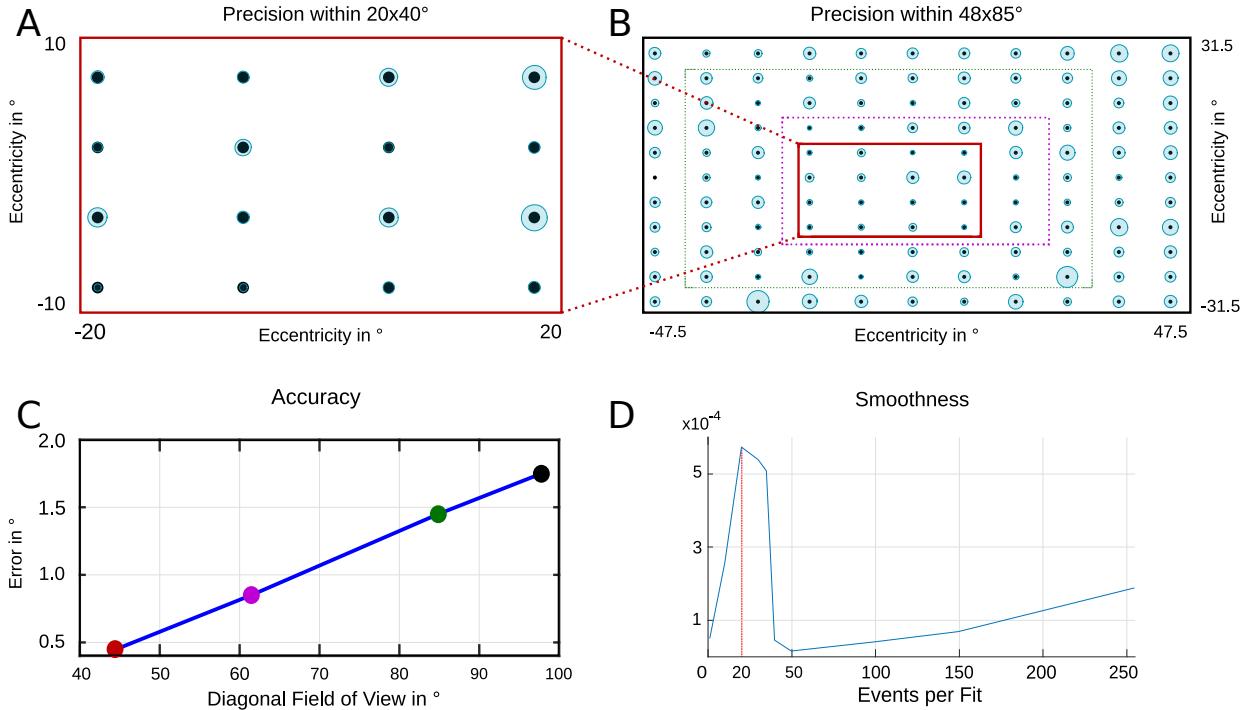


Fig. 8. **Precision and accuracy of our system** depend on the target field of view. The circle diameters in plots A and B represent the precision, defined as the trimmed standard deviation of estimated gaze points centered at their ground truth label, averaged across all subjects. Plot A is a restricted field of view from plot B. The average precision is 1.6° in the smaller field of view and 3° in the larger. Calibration is performed using only half the points within the evaluated field of view. The black circles in the center represent our stimulus size, 1.3° . Plot C shows the best accuracy of a single subject, which ranges from 0.45 – 1.75° for diagonal fields of view between 45 – 98° . Plot D illustrates the empirical smoothness (see the Appendix for rigorous definition), against the number of events per fit for our method. The plot shows a clear optimum at around 20 events per fit, indicating that the robustness is highest for this setting

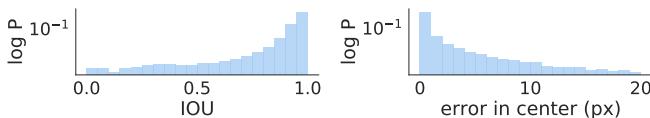


Fig. 9. **The error in center estimation and intersection over union (IOU)** of our method over the full field of view and all subjects are plotted as normalized histograms. Notice the log scale on the y axis.

from a frame. IOU measures the quality of the pupil estimate x an IOU of 1 implies perfect overlap of x and the ground truth y . We compared the event-based pupil estimate obtained immediately before every frame in our dataset and compared it to the the pupil estimate from the frame. The IOU almost never falls below 0.8 and the pupil center almost never deviates by more than 3 pixels. These results inspire confidence that our method tracks the true underlying motion of the eye, since the events agree with an independently estimated pupil from the future frame.

Gaze Mapping Accuracy Accuracy represents the closeness of the pupil estimates to ground truth. Specifically, we calculate it as the average absolute deviation of the pupil estimates from the labelled gaze points in the random saccade experiment. Our system achieves an accuracy comparable with commercial frame-based eye tracking systems, $< 0.5^\circ$ [16], within a standard field of view, which degrades to 2° on a larger field (see bottom row of Figure 8). A purely frame-based eye-tracking system provides a lower-bound on our accuracy since events add information during fast motions of the eye that frames cannot sample. However, there is no way to evaluate the accuracy of event-wise updates of the pupil's position with a traditional camera and monitor. Hence, we performed our own experiments, in which we compared our frame-based algorithm's estimated gaze-locations with the ground truth point a user was looking at on a screen during saccadic motions.

Although we cannot directly evaluate the accuracy in-between frames, using the assumption that eye motion is continuous in camera-space at a very small timescale (this is why we optimized for smoothness of the trace in Figure 7), we can indirectly assess accuracy at event rate. If our event-based updates did not match the motion of the pupil, then, when a frame was received, the X and Y pupil center traces in Figure 7 would have a “glitch” corresponding to the correction of the bad estimate (as in the case of the blink at roughly the 7 seconds mark). This does not occur, so our system must at least match the accuracy of our frame-based method.

Gaze Mapping Precision We calculate precision as the empirical standard deviation, in visual angle, of the estimated pupil centers and plot it in Figure 8A and B across all subjects. The blue circles in the plot are thus a measure of the statistical spread, while the accuracy is reported as a line in the Figure 8C. We assess precision by fitting a polynomial on a subset of points and evaluating on a held-out set. Specifically we train on all the “even” positions on the grid and evaluate on the “odd” ones, do the opposite and average the numbers. We obtain 1.6° of visual angle precision on the smaller field of view. This precision decreases on the larger field to 3° . Precision can also be visually assessed in Figure 6. In Figure 8, only the last half second (out of a 1.5 s stimulus presentation) is used for each saccade, as we assume the user's gaze might have changed in the first second (due to the large FoV, some subjects had to search for the stimulus). Blinks are removed using an automated blink detector which is part of our pipeline (see supplement). The top and bottom center plots are obtained by fitting a second-degree polynomial regressor on a smaller FoV, while the right plots are obtained by fitting a second-degree polynomial on a larger FoV. Notice that at the edges of the top right and bottom left displays of Figure 8, we can clearly observe that the accuracy and precision both worsen in the edges of the field of view, both because of occlusion and also because a small change in pupil center can have a large effect on gaze location when viewed at that angle.

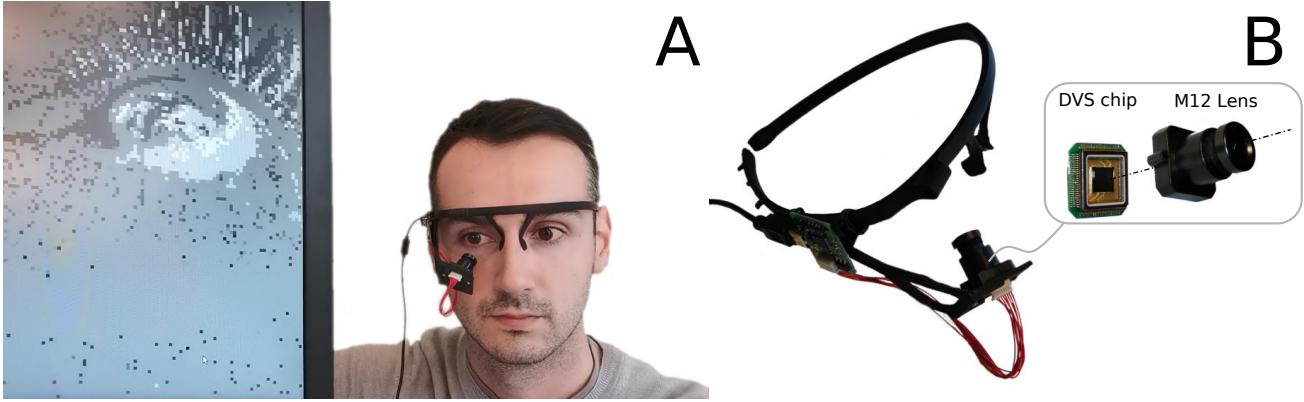


Fig. 10. **Our miniature prototype** streams events in real time (A) and mounts on a pair of glasses (B), shown here with a M12 lens. Data is streamed out of the prototype using less than 12Mbit/s of bandwidth.

We quantitatively assess the precision of our system in Figure 8. A second-degree polynomial is fit on the best-subject for both a small $20 \times 40^\circ$ (top left) and larger $63 \times 95^\circ$ (top right) field of view. The total FoV $64 \times 96^\circ$ spanned by the screen in our experiments is comparatively much larger (and harder to regress) than previously reported in the literature ($26 \times 40^\circ$ etc.) [16], explaining why we report results for two regressors fitted on two FoV. Precision in visual angle averaged for all the points in the small FoV is 1.6° .

6 DISCUSSION

We presented a system and a method for near-eye gaze tracking using a vision sensor that can produce both conventional images and event data. Our system inherits the capabilities of frame-based sensors: we demonstrated state-of-the-art precision of 0.5° of visual angle error in a 121 fixation point task, in addition to the advantages of event-based data. Specifically, we obtain a conservative peak rate of 200 events per millisecond in a 2 px radius around the pupil and showed our method can achieve a robust fit performed every 20 events. Hence we claim a conservative peak update rate of 10 kHz. Our method can sustain the real-time processing of those 200 evts.ms^{-1} . A method that could reliably estimate the pupil position every single event, or would be able to generate/consider an even higher event rate, would theoretically yield even higher update rates. The update rate we demonstrate is about 10× higher than the fastest commercial systems we surveyed, and such cannot be envisioned, even with a modest resolution conventional camera, due to the bandwidth required to output frames at those frequencies.

Limitations of our work, as well as most other existing eye trackers, includes susceptibility to slippage, and unavailability of ground truth. Specifically, G_θ is not robust to slippage of the cameras with respect to the face. This is an open challenge that, for now, requires periodic recalibration. Ground truth is unavailable for all eye-trackers; the standard method of evaluation is to assess both against one-another with the same methodology [17]. A final limitation is that we do not report the time between the true movement of the human eye and the time we register an update, otherwise known as latency. Latency is heavily dependent on the hardware details of the processor and the communication channel with the camera. Therefore, reporting latency is outside the scope of our work, since we did not attempt to build a custom embedded system. However, due to our extremely fast update rate, our software would surely not limit the speed of a commercial system, and optimizing the hardware would ensure fast operation. We estimate each event based update to require, in worst case, about 300 FLOPS. This means, our algorithm lends itself to dedicated implementations that are likely to be very efficient on most modern low-power embedded-processors. Additionally, although it was not the focus of our work, there is the possibility of using a deep convolutional neural network to output a pupil center estimate in our gaze tracking pipeline (see Figure 2, top). This may improve the robustness of our system and allow generalization to different subjects, like NVGaze [28]. It

is worth noting, however, that the best-case accuracy NVGaze reports when trained and evaluated on a single subject in a near-eye scenario is 0.5° , the same as ours. Additionally, using such models in a compute-constrained setting may bottleneck tracking speed or consume too much power. Finally, our choices of the tuning parameters γ , δ , and # of events per fit (see Figure 8) depend on assumptions about the true Lipschitz parameter of the eye’s motion. Partially because the true Lipschitz parameter is unknown, changing, and perhaps unknowable, the user of our system will need to hand tune these parameters to strike a balance between the sensitivity of the method to jerky eye motions and its susceptibility to noise. Because our dataset consists mostly of smooth motions like fixation, saccades, and smooth pursuit, we just chose the parameters that maximized the smoothness. However, a vision scientist may find it interesting to experiment with other parameter choices when studying subtler eye motions. Indeed, one can imagine a future system that dynamically adjusts the tuning parameters over time based on the motion of the eye as it is being tracked.

6.1 Towards a mobile, event-based tracker for AR/VR

We built a head-mounted prototype of our eye tracking system to demonstrate its potential for miniaturization. We designed a custom mount that sits on a user’s head and holds a 22×22 mm DVS and lens connected to a battery-powered Raspberry Pi. Figure 10 shows our setup. It is slightly different from the one we used in our main experiments, since the sensor is lower resolution (128×128 pixels) and does not output frames. However, the DAVIS346 chip used in our desktop mounted experiments is already small enough (25×25 mm) to fit on the same mount; this would require a custom PCB, which we will soon be able to demonstrate, along with recorded data from our setup and our custom 3D mount².

Particularly given the relative ease of building a miniature event-based near-eye gaze tracker, we believe the technology could benefit the AR/VR community. We discussed the speed advantage in Section 1.1, but there are further benefits. For example, AR/VR systems must conserve power to extend battery life, and event-based eye tracking consumes less power. To see why, notice that during fixation and small eye motions, the sensor induces fewer events and incurs less processing, whereas each pixel in a frame must always be acquired and processed in a traditional system. Furthermore, AR systems should perform well in a variety of indoor and outdoor lighting conditions; event-based cameras have a dynamic range of $\approx 130\text{dB}$, much more robust than the traditional camera. The future of eye-tracking undoubtedly involves mobile, untethered, always-on AR/VR glasses, so the efficiency of event-based sensors makes them a natural choice. As a starting direction for interested future researchers, we note that the high update rate of our system would likely enable prediction of saccadic landing zones and durations, with AR/VR applications in efficient predictive foveated

²See project webpage: <http://angelopoulos.ai/blog/posts/ebv-eye/>

rendering [22,29] and autofocus AR [7,49]. The most notable remaining challenge is slippage: head-mounted eye tracking systems often require 3D modeling of the user’s head and face pose or recalibration to account for relative movement between the eye tracker and the head during use. Our evaluations in Section 5 do not account for slippage, which is often the primary driver of error in head-mounted eye tracking products. However, given the longstanding work on event-driven 3D algorithms like SLAM [47], we are optimistic that future researchers will use our platform as a starting point to build event-based slip compensation algorithms too. As a parting conjecture, since the polynomial mapping between the pupil center and gaze point is quite simple, an automatic calibration scheme may be possible by selecting an appropriate nearest-neighbor from a bank of users; the neighbor might be chosen based on the glint and ellipse parameters, for example.

ACKNOWLEDGMENTS

A.N.A. was supported by a National Science Foundation (NSF) Fellowship and a Berkeley Fellowship. J.N.P.M. was supported by a Swiss National Foundation (SNF) Fellowship (P2EZP2 181817), G.W. was supported by an NSF CAREER Award (IIS 1553333), a Sloan Fellowship, by the KAUST Office of Sponsored Research through the Visual Computing Center CCF grant, and a PECASE by the ARL. Thanks to Stephen Boyd and Mert Pilanci for helpful conversations.

REFERENCES

- [1] Pupil labs website. <https://pupil-labs.com/products/vr-ar/tech-specs>. Accessed: 2019-08-14.
- [2] Sr research website. <https://www.sr-research.com/products/eyelink-1000-plus/>. Accessed: 2019-11-12.
- [3] Tobii website. <https://vr.tobii.com/products/htc-vive-pro-eye/>. Accessed: 2019-11-12.
- [4] R. A. Abrams, D. E. Meyer, and S. Kornblum. Speed and accuracy of saccadic eye movements: characteristics of impulse variability in the oculomotor system. *Journal of Experimental Psychology: Human Perception and Performance*, 15(3):529, 1989.
- [5] K. Akşit, J. Kautz, and D. Luebke. Gaze-sensing leds for head mounted displays. *arXiv preprint arXiv:2003.08499*, 2020.
- [6] S. Baluja and D. Pomerleau. Non-intrusive gaze tracking using artificial neural networks. In *Advances in Neural Information Processing Systems*, pp. 753–760, 1994.
- [7] P. Chakravarthula, D. Dunn, K. Akşit, and H. Fuchs. Focusar: Auto-focus augmented reality eyeglasses for both real world and virtual imagery. *IEEE transactions on visualization and computer graphics*, 24(11):2906–2916, 2018.
- [8] T. N. Cornsweet and H. D. Crane. Accurate two-dimensional eye tracker using first and fourth purkinje images. *JOSA*, 63(8):921–928, 1973.
- [9] H. D. Crane and C. M. Steele. Generation-v dual-purkinje-image eyetracker. *Applied Optics*, 24(4):527–537, 1985.
- [10] G. Damian. Eye tracking using event-based silicon retina, 2007.
- [11] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse. Monoslam: Real-time single camera slam. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (6):1052–1067, 2007.
- [12] T. Delbrück. Silicon retina with correlation-based, velocity-tuned pixels. *IEEE Transactions on Neural Networks*, 4(3):529–541, 1993.
- [13] T. Delbrück, B. Linares-Barranco, E. Culurciello, and C. Posch. Activity-driven, event-based vision sensors. In *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*, pp. 2426–2429. IEEE, 2010.
- [14] A. K. Dexl, H. Schlägel, M. Wolfbauer, and G. Grabner. Device for improving quantification of reading acuity and reading speed. *Journal of Refractive Surgery*, 26(9):682–688, 2010.
- [15] A. T. Duchowski. Eye tracking methodology. *Theory and practice*, 328(614):2–3, 2007.
- [16] B. V. Ehinger, K. Gross, I. Ibs, and P. Koenig. A new comprehensive eye-tracking test battery concurrently evaluating the pupil labs glasses and the eyelink 1000. *BioRxiv*, p. 536243, 2019.
- [17] Eyelink-Support-Staff. Private Communication, 2019. To understand how EyeLink arrived at their accuracy and speed claims we corresponded with them. The correspondence is anonymized and attached in the supplement.
- [18] W. Fuhl, T. Kübler, K. Sippel, W. Rosenstiel, and E. Kasneci. Excuse: Robust pupil detection in real-world scenarios. In *International Conference on Computer Analysis of Images and Patterns*, pp. 39–51. Springer, 2015.
- [19] W. Fuhl, T. C. Santini, T. Kübler, and E. Kasneci. Else: Ellipse selection for robust pupil detection in real-world environments. In *Proceedings of the Ninth Biennial ACM Symposium on Eye Tracking Research & Applications*, pp. 123–130. ACM, 2016.
- [20] G. Gallego, T. Delbrück, G. Orchard, C. Bartolozzi, B. Taba, A. Censi, S. Leutenegger, A. Davison, J. Conradt, K. Daniilidis, et al. Event-based vision: A survey. *arXiv preprint arXiv:1904.08405*, 2019.
- [21] D. Gehrig, H. Rebecq, G. Gallego, and D. Scaramuzza. Asynchronous, photometric feature tracking using events and frames. *CoRR*, abs/1807.09713, 2018.
- [22] B. Guenter, M. Finch, S. Drucker, D. Tan, and J. Snyder. Foveated 3d graphics. *ACM Transactions on Graphics (TOG)*, 31(6):164, 2012.
- [23] W. W. Hager. Updating the inverse of a matrix. *SIAM review*, 31(2):221–239, 1989.
- [24] D. W. Hansen and A. E. Pece. Eye tracking in the wild. *Computer Vision and Image Understanding*, 98(1):155–181, 2005.
- [25] iniVation. Datasheet, 2020. DAVIS346 Sensor, maximal value reported in the datasheet.
- [26] J. Kim, P. Knowles, J. Spjut, B. Boudaoud, and M. McGuire. Post-render warp with late input sampling improves aiming under high latency conditions. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 3(2):1–18, 2020.
- [27] J. Kim, J. Spjut, M. McGuire, A. Majercik, B. Boudaoud, R. Albert, and D. Luebke. Esports arms race: Latency and refresh rate for competitive gaming tasks. *Journal of Vision*, 19(10):218c, 2019.
- [28] J. Kim, M. Stengel, A. Majercik, S. De Mello, D. Dunn, S. Laine, M. McGuire, and D. Luebke. Nvgaze: An anatomically-informed dataset for low-latency, near-eye gaze estimation. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, p. 550. ACM, 2019.
- [29] R. Konrad, A. Angelopoulos, and G. Wetzstein. Gaze-contingent ocular parallax rendering for virtual reality. *ACM Trans. Graph.*, 39, 2020.
- [30] G. A. Koulieris, K. Akşit, M. Stengel, R. K. Mantuk, K. Mania, and C. Richardt. Near-eye display and tracking technologies for virtual and augmented reality. In *Computer Graphics Forum*, vol. 38, pp. 493–519. Wiley Online Library, 2019.
- [31] K. Kafka, A. Khosla, P. Kellnhofer, H. Kannan, S. Bhandarkar, W. Matasik, and A. Torralba. Eye tracking for everyone. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2176–2184, 2016.
- [32] M. Kyto, 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*, p. 81. ACM, 2018.
- [33] J. H. Lee, P. K. Park, C.-W. Shin, H. Ryu, B. C. Kang, and T. Delbrück. Touchless hand gesture ui with instantaneous responses. In *2012 19th IEEE International Conference on Image Processing*, pp. 1957–1960. IEEE, 2012.
- [34] G. E. Legge, C. M. Madison, and J. S. Mansfield. Measuring braille reading speed with the mnread test. *Visual Impairment Research*, 1(3):131–145, 1999.
- [35] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale. Keyframe-based visual-inertial odometry using nonlinear optimization. *The International Journal of Robotics Research*, 34(3):314–334, 2015.
- [36] R. Li, E. Whitmire, M. Stengel, B. Boudaoud, J. Kautz, D. Luebke, S. Patel, and K. Akşit. Optical gaze tracking with spatially-sparse single-pixel detectors. *arXiv preprint arXiv:2009.06875*, 2020.
- [37] Y. Li, S. Wang, and X. Ding. Eye/eyes tracking based on a unified deformable template and particle filtering. *Pattern Recognition Letters*, 31(11):1377–1387, 2010.
- [38] P. Lichtsteiner, C. Posch, and T. Delbrück. A 128x128 120 db 15us latency asynchronous temporal contrast vision sensor. *IEEE journal of solid-state circuits*, 43(2):566–576, 2008.
- [39] H. Liu, D. P. Moeys, G. Das, D. Neil, S.-C. Liu, and T. Delbrück. Combined frame-and event-based detection and tracking. In *2016 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 2511–2514. IEEE, 2016.
- [40] H. Liu, D. P. Moeys, G. Das, D. Neil, S.-C. Liu, and T. Delbrück. Combined frame-and event-based detection and tracking. In *2016 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 2511–2514. IEEE, 2016.

- [41] M. Mahowald. The silicon retina. In *An Analog VLSI System for Stereoscopic Vision*, pp. 4–65. Springer, 1994.
- [42] J. N. Martel, J. Müller, J. Conradt, and Y. Sandamirskaya. An active approach to solving the stereo matching problem using event-based sensors. In *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1–5. IEEE, 2018.
- [43] C. A. Mead and M. A. Mahowald. A silicon model of early visual processing. *Neural networks*, 1(1):91–97, 1988.
- [44] A. Mitrokhin, C. Fermüller, C. Parameshwara, and Y. Aloimonos. Event-based moving object detection and tracking. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1–9. IEEE, 2018.
- [45] C. H. Morimoto and M. R. Mimica. Eye gaze tracking techniques for interactive applications. *Computer vision and image understanding*, 98(1):4–24, 2005.
- [46] E. Mueggler, H. Rebecq, G. Gallego, T. Delbruck, and D. Scaramuzza. The event-camera dataset and simulator: Event-based data for pose estimation, visual odometry, and slam. *The International Journal of Robotics Research*, 36(2):142–149, 2017.
- [47] E. Mueggler, H. Rebecq, G. Gallego, T. Delbruck, and D. Scaramuzza. The event-camera dataset and simulator: Event-based data for pose estimation, visual odometry, and slam. *The International Journal of Robotics Research*, 36(2):142–149, 2017.
- [48] I. M. Neumann, B. A. Lege, M. Bauer, J. M. Hassel, A. Hilger, and T. F. Neumann. Static and dynamic rotational eye tracking during lasik treatment of myopicastigmatism with the zyoptix laser platform and advanced control eye tracker. *Journal of Refractive Surgery*, 26(1):17–27, 2010.
- [49] N. Padmanaban, R. Konrad, and G. Wetzstein. Autofocals: Evaluating gaze-contingent eyeglasses for presbyopes. *Science advances*, 5(6):eaav6187, 2019.
- [50] C. Posch, T. Serrano-Gotarredona, B. Linares-Barranco, and T. Delbruck. Retinomorphic event-based vision sensors: bioinspired cameras with spiking output. *Proceedings of the IEEE*, 102(10):1470–1484, 2014.
- [51] R. Ranjan, S. De Mello, and J. Kautz. Light-weight head pose invariant gaze tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 2156–2164, 2018.
- [52] H. Rebecq, T. Horstschaefner, and D. Scaramuzza. Real-time visual-inertial odometry for event cameras using keyframe-based nonlinear optimization. In *BMVC*, 2017.
- [53] T. Santini, W. Fuhl, and E. Kasneci. Purest: robust pupil tracking for real-time pervasive eye tracking. In *Proceedings of the 2018 ACM Symposium on Eye Tracking Research & Applications*, p. 61. ACM, 2018.
- [54] SONY. Datasheet, 2019. IMX387 Pregius sensor, SLVS – EC 8 Lane 12 bit 40.4 FPS, operating in typical conditions.
- [55] Y. Sugano, Y. Matsushita, and Y. Sato. Learning-by-synthesis for appearance-based 3d gaze estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1821–1828, 2014.
- [56] Y.-l. Tian, T. Kanade, and J. F. Cohn. Dual-state parametric eye tracking. In *Proceedings Fourth IEEE International Conference on Automatic Face and Gesture Recognition (Cat. No. PR00580)*, pp. 110–115. IEEE, 2000.
- [57] C. Topal, Ö. N. Gerek, and A. Doğan. A head-mounted sensor-based eye tracking device: eye touch system. In *Proceedings of the 2008 symposium on Eye tracking research & applications*, pp. 87–90, 2008.
- [58] P. Verghese and B. R. Beutter. Motion processing. In V. Ramachandran, ed., *Encyclopedia of the Human Brain*, pp. 117 – 135. Academic Press, New York, 2002. doi: 10.1016/B0-12-227210-2/00215-6
- [59] U. Vogel, D. Kreye, B. Richter, G. Bunk, S. Reckziegel, R. Herold, M. Scholles, M. Törker, C. Grillberger, J. Amelung, et al. Bi-directional oled microdisplay for interactive see-through hmds: Study toward integration of eye-tracking and informational facilities. *Journal of the Society for Information Display*, 17(3):175–184, 2009.
- [60] K. Wang and Q. Ji. Real time eye gaze tracking with 3d deformable eye-face model. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1003–1011, 2017.
- [61] D. Weikersdorfer, D. B. Adrian, D. Cremers, and J. Conradt. Event-based 3d slam with a depth-augmented dynamic vision sensor. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 359–364. IEEE, 2014.
- [62] E. Wood, T. Baltrušaitis, L.-P. Morency, P. Robinson, and A. Bulling. Learning an appearance-based gaze estimator from one million synthesised images. In *Proceedings of the Ninth Biennial ACM Symposium on Eye Tracking Research & Applications*, pp. 131–138. ACM, 2016.
- [63] E. Wood, T. Baltrušaitis, X. Zhang, Y. Sugano, P. Robinson, and A. Bulling. Rendering of eyes for eye-shape registration and gaze estimation. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 3756–3764, 2015.
- [64] L. R. Young and D. Sheena. Survey of eye movement recording methods. *Behavior research methods & instrumentation*, 7(5):397–429, 1975.
- [65] X. Zhang, Y. Sugano, M. Fritz, and A. Bulling. Mpiigaze: Real-world dataset and deep appearance-based gaze estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(1):162–175, Jan 2019. doi: 10.1109/TPAMI.2017.2778103
- [66] W. Zhu and H. Deng. Monocular free-head 3d gaze tracking with deep learning and geometry constraints. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 3143–3152, 2017.

A FORMULATION OF THE PROBLEM FOR \mathcal{P} AND \mathcal{C}

A.1 Updating and fitting the parabola \mathcal{P}

The location of points $\vec{p} = (x, y)$ on the parabola representing the eyelash in the image plane satisfy the quadric equation:

$$E_{\mathcal{P}}(\vec{p}) = E_{\mathcal{P}}(x, y) = 0 \quad (9)$$

with $E_{\mathcal{P}}(x, y) = a' y^2 + g' y + d' - x$

Here, we assume the parabola can be written as a function $x = \text{fct}^\circ(y)$ in which the y axis is along the columns and the x axis along the rows. This assumption is valid as long as the eye is “upright”, that is, it is not excessively rotated in the image-space. The parabola is thus parameterized as $\mathcal{P} = (a', g', d')^\top$ and it can be fitted similarly to the ellipses representing the pupils by solving:

$$\mathcal{P}^* = \arg \min_{\mathcal{P}} \sum_{(x, y) \in \mathcal{D}'} E_{\mathcal{P}}(x, y)^2 \quad (10)$$

In which \mathcal{D}' is the set of points belonging to the parabola, which is analogous to the set \mathcal{D} for the ellipse representing the pupil. Again, the solution is simply $\mathcal{P}^* = A'^{-1} b'$ with

$$A' = \sum_{(x, y) \in \mathcal{D}_{\text{img}}} v_{x, y}^{1\top} v^{1x, y\top}, \quad b' = \sum_{(x, y) \in \mathcal{D}_{\text{img}}} v_{x, y}^{2\top} \quad (11)$$

in which $v_{x, y}^{1\top}$ is now:

$$v_{x, y}^{1\top} = (y^2, y, 1)^\top \quad (12)$$

and $v_{x, y}^{2\top}$ is different (due to the asymmetry in x and y):

$$v_{x, y}^{2\top} = x v^{1x, y\top} = (xy^2, xy, x) \quad (13)$$

Lastly, we need to describe how the points are selected in \mathcal{D}' . Similarly to $\mathcal{D} = \mathcal{D}_{\text{evt}} \cup \mathcal{D}_{\text{img}}$, we define $\mathcal{D}' = \mathcal{D}'_{\text{evt}} \cup \mathcal{D}'_{\text{img}}$. We detect the points belonging to the eyelash in an image by running a Harris corner detector on an image in which all grey values outside of $[t_1, t_2]$ have been clipped. Then all the candidate Harris corners further than a certain radius ρ' from the currently estimated pupil center are discarded, and all the points in the lower half of the image are also discarded. This is:

$$\begin{aligned} \mathcal{D}'_{\text{img.}} = & \{ \vec{p} = (x, y)^\top \mid \vec{p} \in \text{HarrisCorner} \circ \text{clip}(I, t_1, t_2), \\ & \text{and } \|\vec{p} - (x_e, y_e)^\top\|^2 < \rho', \\ & \text{and } y < \frac{\text{rows}}{2} \} \end{aligned} \quad (14)$$

The events considered in the fit for the parabola are also those falling with a radius δ of the currently estimated parabola.

$$\mathcal{D}'_{\text{evt.}} = \{ \vec{p} \mid |E_{\mathcal{P}}(\vec{p})| < \delta \} \quad (15)$$

A.2 Updating and fitting the circle for the glint \mathcal{C}

Regressing the glint is a subcase of regressing the ellipse in which one can choose the scaling $a = b = 1$, and has $h = 0$. The location of points $\vec{p} = (x, y)^\top$ on the circle representing the glint in the image plane satisfy the quadric equation:

$$E_{\mathcal{C}}(\vec{p}) = E_{\mathcal{C}}(x, y) = 0 \quad (16)$$

with $E_{\mathcal{C}}(x, y) = x^2 - 2x c_x - 2y c_y + (c_x^2 + c_y^2 - r^2)$

Hence we have to fit the parameters $\mathcal{C} = (c_x, c_y, r)^\top$. The points in an image to be selected to update the fit of the glint are those that exceed a certain threshold t_3 and that are less than a certain distance ρ'' from the current pupil center estimate.

$$\begin{aligned} \mathcal{D}''_{\text{img.}} = & \{ \vec{p} = (x, y)^\top \mid \vec{p} \in H_{t_3}(I(x, y)) \\ & \text{and } \|\vec{p} - (x_e, y_e)^\top\|^2 < \rho' \} \end{aligned} \quad (17)$$

Events are those falling less than δ away from the currently estimated glint:

$$\mathcal{D}''_{\text{evt.}} = \{ \vec{p} \mid |E_{\mathcal{C}}(x, y)|^2 < \delta^2 \} \quad (18)$$

B ADDITIONAL DETAILS ABOUT THE FITTING OF θ FOR THE REGRESSOR

The regressor maps parameters of \mathcal{M} to screen coordinates x_s, y_s . In its simplest form, also yielding the lowest computational load, our regressor is a second order polynomial that takes as input, the pupil center $(x_e, y_e)^\top$ extracted from the parameters \mathcal{E} , and outputs the point $(x_s, y_s)^\top$ in screen coordinates that the user is supposed to look at. Thus, our regressor is a multivariate vector-valued function:

$$G_\theta(\mathcal{E}) = \begin{pmatrix} x_s \\ y_s \end{pmatrix} = \begin{pmatrix} G_{\theta^1}|_x(\mathcal{E}) \\ G_{\theta^2}|_y(\mathcal{E}) \end{pmatrix} \quad (19)$$

In which the coordinate functions $G_{\theta^1}|_x(\mathcal{E})$ (on x) and $G_{\theta^2}|_y(\mathcal{E})$ (on y) are second order polynomials with parameters θ^1 and θ^2 :

$$G_{\theta^i}|_{x/y}(x_e, y_e) = \alpha_i x_e^2 + \gamma_i x_e y_e + \beta_i y_e^2 + \varepsilon_i x_e + \zeta_i y_e + \eta_i \quad (20)$$

The parameters θ^1 and θ^2 are fitted solving the following linear least squares (G's are linear in their coefficients):

$$\arg \min_{\theta^1} \|G_{\theta^1}|_x(x_e, y_e) - x_s\|^2 \quad (21)$$

$$\arg \min_{\theta^2} \|G_{\theta^2}|_y(x_e, y_e) - y_s\|^2 \quad (22)$$

The regression is supervised by pairs $\{((x_e, y_e)^\top, (x_s, y_s)^\top)\}$ produced during the calibration procedure in which (x_s, y_s) points are presented to the user, and $(x_e, y_e)^\top$ are obtained from the ellipse fit in image space:

$$x_e = \frac{2bg - hf}{h^2 - 4ab}, \quad y_e = \frac{2af - hg}{h^2 - 4ab} \quad (23)$$

From the parameters $\mathcal{E} = (a, h, b, g, f, d)^\top$

C DEFINITIONS USED FOR ACCURACY AND PRECISION IN OUR EXPERIMENT

In the main text, our “accuracy” results are calculated using the ISO 5725 definition of “trueness”: the closeness of agreement between the arithmetic mean of a large number of test results and the true or accepted reference value, this is:

$$\text{Accuracy} = \frac{1}{n} \sum_{i=1}^n l_i \quad (24)$$

where n is the total number of estimates, and l_i is the L_2 norm of the difference between our estimated gaze direction $\hat{d}_i = (\hat{\phi}_i, \hat{\theta}_i)$ and the true gaze angle $d_i = (\phi_i, \theta_i)$:

$$l_i = \|\hat{d}_i - d_i\|_2 \quad (25)$$

Precision is defined by ISO 5725 as: “the closeness of agreement between test results”. We quantify this by computing the empirical standard deviation of the \hat{d}_i 's:

$$\text{Precision} = \sqrt{\frac{1}{n-1} \sum_{i=1}^n (\hat{d}_i - \bar{\hat{d}})^2} \quad (26)$$

With $\bar{\hat{d}}$ being the empirical mean of all \hat{d}_i .

D CALCULATING θ AND ϕ FROM x_s AND y_s

The relationship between the horizontal angle θ , the vertical angle ϕ , and the screen coordinates which a user looks at, (x_s, y_s) , is trigonometric because the screen is at a fixed distance. Although the gaze vector seemingly gives information about 3D space, it is in fact only a two-dimensional quantity, because it has no (or unit) magnitude. With (c_x, c_y) being the center of the screen at a fixed distance D , The conversion is calculated simply as:

$$\theta = \frac{180}{\pi} \tan^{-1}(|x_s - c_x|/D)$$

$$\phi = \frac{180}{\pi} \tan^{-1}(|y_s - c_y|/D)$$

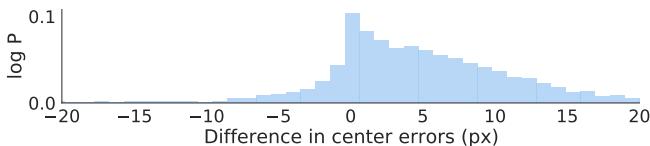


Fig. 11. The difference in pupil center error of the frame-only ablation is plotted as a histogram. See Appendix G for details.

E BLINK DETECTOR

We use a blink detector to classify which frames are likely to be blinks and remove them from our accuracy and precision calculations. The blink detector operates on a simple principle: during a blink, our fitted ellipse to the pupil will deform drastically in shape, often becoming long and thin to fit the dark line of the eyelashes. This blink detector uses changes in *eccentricity*, or the ratio between the major and minor radius of the fitted ellipse, to identify blinks. However, raw changes in eccentricity can vary widely when the eye is looking at different areas on the screen; so, an adaptive threshold must be computed to identify relatively large changes in eccentricity. Specifically, the eccentricities $r_{1:n}$ of the last n fitted ellipses are stored in a vector, R_n . The sample mean μ_n and sample standard deviation σ_n of R_n are calculated. Then, when a new frame comes in at time $n+1$, the eccentricity of the fitted ellipse from that frame, r_{n+1} , is compared to $\mu_n + \lambda\sigma_n$, with $\lambda > 0$ being a tunable parameter. If $r_{n+1} > \mu_n + \lambda\sigma_n$, then that frame will be classified as a blink, and is not considered in accuracy and precision calculations. In addition, a small number k of following frames are also classified as blinks, since a blink takes on average 3–4 frames to complete. Then, $r_{2:n+1}$ is assigned to R_{n+1} to preserve a constant buffer length, and the blink detector considers the next frame (r_{n+2}). Finally, we trim the data at the 2.5% level to remove outliers. This blink detection method is advantageous in that it adaptively computes how large of an eccentricity change constitutes a blink based on statistics of our ellipse data; it is also conservative, in that if eccentricity changes in a certain region of gaze directions are high on average, they will not be classified as blinks.

F EMPIRICAL SMOOTHNESS

Consider a sequence of gaze estimates $\{x_s^{(t)}, y_s^{(t)}\}_{t=1}^T$ at evenly spaced times $t = 1, \dots, T$. We quantify the empirical smoothness of the eye as

$$\text{smoothness} = \frac{1}{T-1} \sum_{t=2}^T \frac{1}{\| [x_s^{(t)}, y_s^{(t)}] - [x_s^{(t-1)}, y_s^{(t-1)}] \|_2}.$$

When the eye center does not change much between increments, the smoothness value is high. The smoothness value is dependent on the time scale; we would expect that at our high update rate, the eye's motion should be smooth because the increments of time are small. However, we would not expect smooth motion at the 30Hz rate of frames. Nonetheless, because we cannot know the true physiological smoothness of the eye's motion, optimizing parameters based on the empirical smoothness is an assumption. We discuss the limitations of this assumption in Section 6 and suggest how future work may lift it.

G FRAME ONLY ABLATION

We consider the relative error our system would incur if events were not used. In particular, we define a simple *frame-only ablation*: rather than track the pupil between frames, we assume it does not change position between frames. This frame-only ablation is equivalent to traditional (non-predictive) eye tracking. For the majority of frames in our dataset, the eye is fixated, and in that case, there is not much difference between our system and the frame-only ablation. Quantitatively, we confirm this: the mean IOUs of our system and the frame-only ablation are identical on such frames. It is more informative to discuss the relative performance of the frame-only ablation on saccadic motions. For every frame taken during a saccade, we calculate d_{frame} as the error of the

pupil center estimate of the frame-only ablation, and d_{event} as the error of the pupil center estimate of the full system. Then, in Figure 11, we plot a histogram of $d_{frame} - d_{event}$. Most of the histogram's mass is on positive numbers, illustrating the degradation in performance caused by the frame-only ablation.

Although we do not implement it here, one might also consider an event-only baseline. An event-only ablation of our Algorithm 1 would need to be re-initialized after every blink, since the event-based estimate cannot recover from bad prior estimates of the pupil. However, one can imagine an event-only baseline where events are stored and, for example, a Hough transform is performed to identify the pupil from the pooled event data only. Various tricks could be performed to improve the performance and efficiency of such an algorithm; instead of storing a ring buffer of events, for example, they could be used in an online Hough-transform much like Algorithm 1. Furthermore, the fact that the pupil and iris are concentric could increase the method's robustness (since there would be two nearby peaks in Hough space). We hope these ideas might be useful to a future researcher working with an event-only sensor.