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

Anastasios N. Angelopoulos[1*], Julien N.P. Martel[2*], Amit P.S. Kohli[2], Jörg Conradt[3],
and Gordon Wetzstein[2]

[1]University of California, Berkeley, [2]Stanford University, [3]KTH Royal Institute of Technology
gordon.wetzstein@stanford.edu

Abstract. Fast and accurate eye tracking is crucial for many applications. Current camera-based eye tracking systems, however, are fundamentally limited by their bandwidth, forcing a tradeoff between image resolution and framerate, i.e. between latency and update rate. 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 eye trackers when evaluated in the same conditions. Our system 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 gaze vector from the parametric pupil model in real time. Using the first hybrid frame-event gaze dataset, which will be made public, we demonstrate that our system achieves accuracies of $0.45°$–$1.75°$ for fields of view ranging from $45°$ to $98°$.

1 Introduction

Eye tracking has emerged as an enabling technology in a variety of applications, including LASIK eye surgery[46], AR/VR (e.g.,[4,50,20,48,29]), target selection [32], user interaction [58,16], attention monitoring [59,25,63], psychological studies [42], medical pathology [31], and sports analysis [56]. Fast and accurate eye tracking is critical for most of these applications, and many of them also require portable systems.

State-of-the-art eye trackers use cameras to monitor the user's pupils. Unfortunately, conventional camera sensors are significantly limited in the bandwidths they provide, forcing a tradeoff between camera resolution and frame rate, and they also require relatively high power to operate. For eye tracking applications, the choice between resolution and framerate results in a tradeoff between the accuracy and latency of the system. High-end eye tracking systems use specialized protocols and readout interfaces to maximize bandwidth, but they often sacrifice device form factor, making portable operation challenging. The bandwidth limitation of camera-based eye tracking systems is primarily due to their *frame-based* principle of operation. While this is a great general-purpose strategy, in the specific application of near-eye gaze tracking most pixels capture redundant information: only the pupil moves while most of the image does not change.

Dynamic vision sensors (DVS) have recently emerged as a new paradigm of data capture, overcoming many of the limitations of conventional frame-based sensing. In a DVS, individual pixels are asynchronously sampled and solely report information whenever they incur a local contrast change. A DVS can therefore be interpreted as adapting its sampling rate to the underlying scene motion [10]. Specifically, a DVS pixel reports a time-stamped data packet whenever the sign of the instantaneous temporal
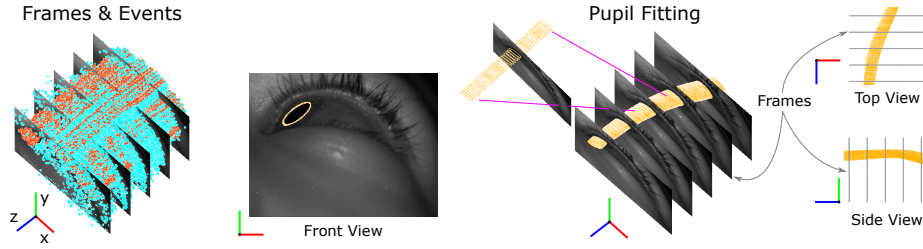
Fig. 1: **Input and output of our system.** The inputs are frames recorded at a fixed sampling rate and events, asynchronously sampling the eye motion at high speed and time resolution (left). Frames and events are captured by the same sensor, and event polarity is color coded in blue (+) or red (-). The output is a fitted pupil, shown in yellow as seen from several perspectives. Events are continuously triggered between the frames, allowing us to update the pupil location at a significantly higher rate than the frame rate. Every estimate of the pupil yields a yellow circle. These estimates are so frequent that they form an almost continuous tubular structure outlining the pupil's movement through time.

derivative of the incident irradiance exceeds a threshold. This packet is referred to as an *event* and it contains the location of the pixel incurring the change, the time at which it happened, and usually the sign of the change. A stream of events is sent off-chip for further processing. For applications where the temporal derivative of a video stream is sparse, event-based capture uses the bandwidth of the system much more efficiently than a frame-based methodology, because only relevant information is sampled and processed. This directly leads to significant improvements in update rates and power consumption while providing portable device form factors.

Here, we report the first real-time event-based eye tracking system. At the core of our system is the update of a parametric representation of the pupil at 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. Our hybrid frame-event fusion approach utilizes pupil estimates obtained at a low update rate by an Active Pixel Sensor pixel sharing the same photodiode as the DVS pixel. The APS sensor thereby optimizes the robustness of the tracker using frame-based pupil-detection algorithms, while the DVS provides updates at the speed of the event stream. 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 using a sensor that could be integrated in a system with smaller form factor and operates at lower power[1].

Our specific contributions are as follows:

 – 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 labeled, binocular dataset of synchronized events and frames with n=24 subjects performing randomized saccadic motion and smooth pursuit.

---

[1] A Pregius line 40FPS SONY sensor consumes 300mW–2W [60] vs. 50mW–0.9W for a DAVIS-346 [24].

2   Related Work

system	update rate (Hz)	accuracy (°)	portable
Pupil Labs [1]	200	∼ 1	✓
Tobii [3]	120	0.5–1.1	✓
EyeLink [2,14]	1,000	∼ 0.5	
Ours	> 10,000	0.45–1.75	✓

Our results can be directly compared with EyeLink because we use the same protocol [15]. Our system's accuracy is $0.45°$ within the same field of view as EyeLink, and speed is $10×$ faster.

Table 1: **Overview of existing eye tracking systems**

Event cameras date back to the neuromorphic Silicon Retina introduced by the seminal work of Mahowald and Mead [41,40]. Activity-driven, event-based vision sensors were advanced over the next thirty years [11,37,10,51]. 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 [55,47,23,43], combined frame and event based object tracking [38], visual odometry [53], 6-DOF pose tracking [45,26,27], SLAM [9,65], and hand tracking [33].

Hybrid frame-event approaches The fusion of events and frame has been explored in different works such as [39,19]. 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 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 [68]. Camera-based eye tracking has evolved from tracking Purkinje reflections [6,7] to model-based approaches that extract parameterizations of the eye from frames [62,36,64]. The model extraction, or *pupil fitting*, can be decoupled from regressing the gaze vector. Algorithms for pupil fitting and gaze estimation are reviewed by Morimoto et al. [44] and Duchowski et al. [13]. However, unlike in event-based vision, these methods are rate-limited by camera frames.

Appearance-based trackers directly estimate gaze vectors from camera frames, often using neural networks [5,30,52,70]. A number of datasets have emerged over the past five years, many of which leverage synthetic data, including: GazeCapture [30], UT Multi-view [61], SynthesEyes [67], UnityEyes [66], MPIIGaze [69], 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 [22]; or near-eye IR-illuminated gaze-tracking for use in controlled environments like an AR/VR headset. 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 potentially portable, 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 [8], 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

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. [62], our eye model consists of an *ellipse representing the pupil* with parameters $\mathcal{E} = (a, h, b, g, f)^\mathsf{T} \in \mathbb{R}^5$, a *parabola representing the eyelashes* with parameters $\mathcal{P} = (u, v, w)^\mathsf{T} \in \mathbb{R}^3$, and a *circle representing the glint* (the reflection of the IR light source off of the user's eyeball) with parameters $\mathcal{C} = (r, c)^\mathsf{T} \in \mathbb{R}^2$. 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.
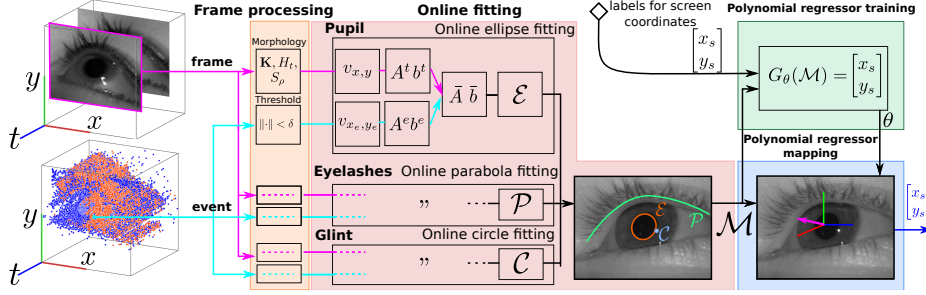
Fig. 2: **A flow diagram of our system.** Both frames and events are inputs to our system. A first stage preprocesses them separately before both streams are combined to update the eye model $\mathcal{M} = \{\mathcal{E}, \mathcal{P}, \mathcal{C}\}$. The model fitting is performed online, in real-time at the arrival rate of frames (low, $25\,\mathrm{Hz}$) and events (high, up to $100\,\mathrm{kHz}$). To output a gaze vector, $\mathcal{M}$ is fed to a polynomial regressor that can be evaluated at event rate. This regressor is trained via a calibration procedure similar to EyeLink's.

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

$$E_{\mathcal{E}}(\boldsymbol{p}) = E_{\mathcal{E}}(x, y) = 0 \tag{1}$$
$$\text{with} \quad E_{\mathcal{E}}(x, y) = a\,x^2 + h\,xy + b\,y^2 + g\,x + f\,y + d$$

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 $\mathcal{D}_{\mathrm{img.}}$. Events near the edge of the pupil are considered candidate points $\mathcal{D}_{\mathrm{evt.}}$. Thus, the model of the ellipse is ultimately updated by the set of points $\mathcal{D} = \mathcal{D}_{\mathrm{evt.}} \cup \mathcal{D}_{\mathrm{img.}}$.

Receiving a frame Under off-axis IR illumination, the pupil appears as a dark blob in the frame (see Fig. 4). 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:

$$\mathcal{D}_{\mathrm{img.}} = \{(x, y) \mid \mathbf{K}(\mathrm{H}_\theta(I) \circ S_\sigma)(x, y) = 1\}, \tag{2}$$

where H_θ is the Heaviside 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 [54], ExCuSe [17], and Else [18].

Receiving an event An event generates a candidate point contributing to the fitting of $\mathcal{E}$ when it falls within a certain distance δ of the last estimated ellipse.

$$\mathcal{D}_{\mathrm{evt.}} = \{(x, y) \mid \boldsymbol{p} = (x, y)^{\mathsf{T}}, \|P_{\mathcal{E}}(\boldsymbol{p}) - \boldsymbol{p}\|_2 < \delta\} \tag{3}$$
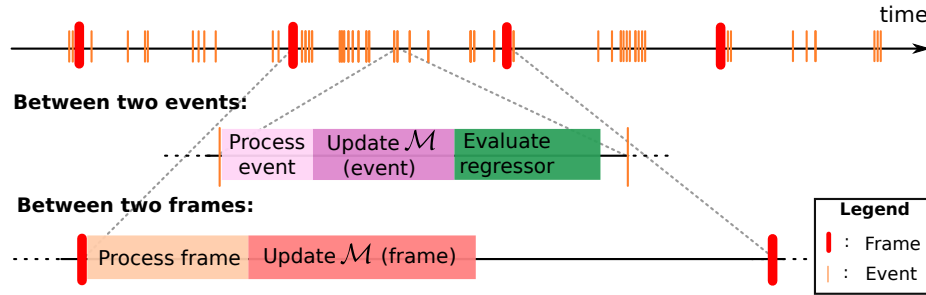
Fig. 3: **The processing flow and time taken by the different stages in our system.** Our system allows events and frames to be processed concurrently and update the same underlying model. Events and frames are shown in time (top row). We also illustrate the operations happening sequentially and concurrently on events (middle) and frames (bottom).

where $P_{\mathcal{E}}(\boldsymbol{p})$ is the projection of a point $\boldsymbol{p}$ on the ellipse, which amounts to solving a system of two equations (one linear and one quadratic) in the specific case of our ellipse parameterization. Our method can perform updates of $\mathcal{M}$ on an event-by-event basis, in which case $\mathcal{D}_{\mathrm{evt.}}$ 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 $\mathcal{D}_{\mathrm{evt.}}$ 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:

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

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

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

and

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

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 (5), 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 (5). 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.

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

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

---

$\bar{A} = \mathrm{Id}_{5\times 5}, \bar{b} = \mathbf{0}_5$ ▷ Init. $\bar{A}$ and $\bar{b}$

$\bar{A}_{\text{inv.}} = \bar{A}^{-1} = \mathrm{Id}_{5\times 5}$ ▷ If using SMW init. $\bar{A}_{\text{inv.}}$

while we are receiving data d **do**

    if d is a frame It **then**

        $\mathcal{D}_{\text{img.}} = \{(x,y) \mid \mathbf{K}(\mathrm{H}_\theta(I^t) \circ S_\rho)(x,y) = 1$

    else if d is an event $E = (\boldsymbol{p} = (x,y)^\intercal, p, t)$ **then**

        $\mathcal{D}_{\text{evt.}} \leftarrow \mathcal{D}_{\text{evt.}} \cup \boldsymbol{p}$ if $\|P_\mathcal{E}(\boldsymbol{p}) - \boldsymbol{p}\|_2 < \delta$

    for all $(x,y) \in \mathcal{D} = \mathcal{D}_{\text{evt.}} \cup \mathcal{D}_{\text{img.}}$ **do**

        $v_{x,y}^\intercal = (x^2, xy, y^2, x, y)^\intercal$ ▷ Eq. (6)

        $A = \sum_{(x,y)\in\mathcal{D}} v_{x,y}\, v_{x,y}^\intercal$ ▷ Eq. (5)

        $b = \sum_{(x,y)\in\mathcal{D}} v_{x,y}$

    if $|\mathcal{D}| > 1$ **then**   ▷ Batch update (frame/acc. evts.)

        $\bar{A} \leftarrow \ \gamma\,\bar{A} + (1-\gamma)\,A$ ▷ Eq. (7)

        $\bar{b} \leftarrow \ \gamma\,\bar{b} + (1-\gamma)\,A$

        $\bar{A}_{\text{inv.}} \leftarrow \bar{A}^{-1}$ ▷ Full-rank inversion

        $\mathcal{E} \leftarrow \bar{A}_{\text{inv.}}\,\bar{b}$

    else   ▷ (Event-based update)

        $\bar{A}_{\text{inv.}} \leftarrow \gamma'\,\bar{A}_{\text{inv.}}$

        $\bar{A}_{\text{inv.}} \leftarrow \bar{A}_{\text{inv.}} - \frac{\bar{A}_{\text{inv.}}\,A\,\bar{A}_{\text{inv.}}}{1 - v_{x,y}^\intercal\,\bar{A}_{\text{inv.}}\,v_{x,y}}$ ▷ SMW

        $\bar{b} \leftarrow \gamma'\,\bar{b} + b$

        $\mathcal{E} \leftarrow \bar{A}_{\text{inv.}}\,\bar{b}$

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. It can be shown to be equivalent to solving a re-weighted least squares problem using all the data points from past frames, weighted by some geometric discount γ (a hyperparameter) of the past.

In the case that $\mathcal{D}^t$ comes from a frame, A^t and b^t can be directly calculated from (5) 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_t,y_t} can nonetheless be computed using Equation (6). The corresponding A^t and b^t for that event are computed using Equation (5). 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 $\bar{A}^{t+1} = \gamma'\,A^t + (1-\gamma')\,\bar{A}^t$, 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,
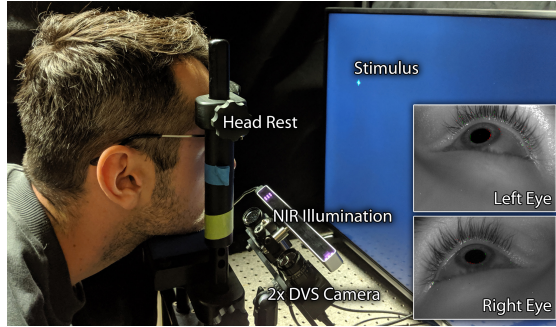
Fig. 4: **The binocular eye tracking system setup used for evaluation.** We collected a dataset with binocular saccadic and smooth-pursuit data on 24 subjects looking at an 11x11 grid of fixation points over a $64 \times 96°$ FoV. Altogether, we collected ~ 30 million events per subject per eye. Two 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.

one can store $(\bar{A}^t)^{-1}$ and use the Sherman-Morrison-Woodbury (SMW) identity [21] 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 3D gaze vector

The output of our gaze tracker is a vector representing the direction a user looks at, i.e. the gaze direction. In practice, we encode it via the 2D screen coordinate the user is looking at. 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 gaze vector to those parameters.

   The 2D screen coordinate position a user is looking at is denoted $(x_s, y_s)^\intercal$. The problem is to find a mapping from $\mathcal{M}$ to $(x_s, y_s)^\intercal$. 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 [44] that consists of phenomenologically regressing the output $(x_s, y_s)^\intercal$ from $\mathcal{M}$ using two 2nd order polynomial functions $G_{\theta^1}|_x$ and $G_{\theta^2}|_y$ (one for each coordinate):

$$G_\theta(\mathcal{M}) = \begin{pmatrix} x_s \\ y_s \end{pmatrix} = \begin{pmatrix} G_{\theta^1}|_x(\mathcal{M}) \\ G_{\theta^2}|_y(\mathcal{M}) \end{pmatrix} \tag{8}$$

   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 $(\mathcal{M}, (x_s, y_s))$ 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. 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 4. 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 [12,34]. 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 $64 \times 96°$.

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 5. 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 5.

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 vector. 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
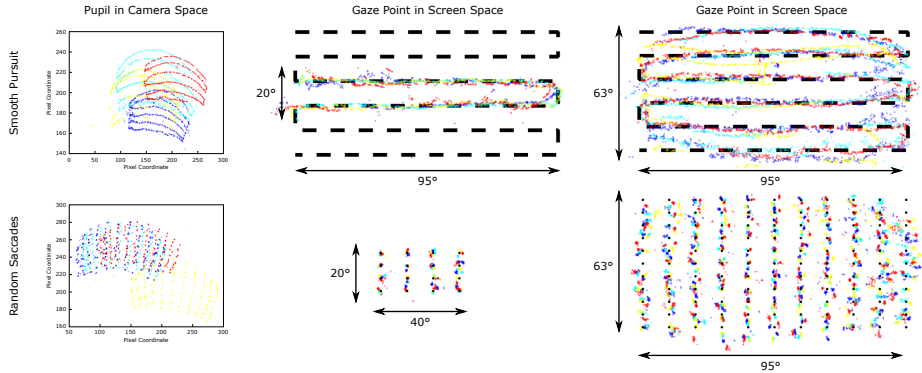
Fig. 5: **Fitted pupil locations for smooth pursuit motion (top row) and random saccadic motion (bottom row)** are shown in camera image space (left column) for four different users (shown in different colors). Estimated gaze vectors in screen space are shown for smaller (center column) and larger (right) fields 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).

set of points. For a similar FoV to EyeLink's, the center $20° \times 40°$ FoV, this means we are testing on 16 points and calibrating on 8. Eyelink uses a 12 point calibration procedure [15] 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. 1,6). 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 6 shows a saccade typically induces more than $\rho = 200$ 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 7 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 $R = \frac{200 \cdot 10^{-3}}{20} = 10$ 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 6 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 ($N = 1$, $\rho = 200$ evts.s^{-1} thus $R = 200$ kHz)
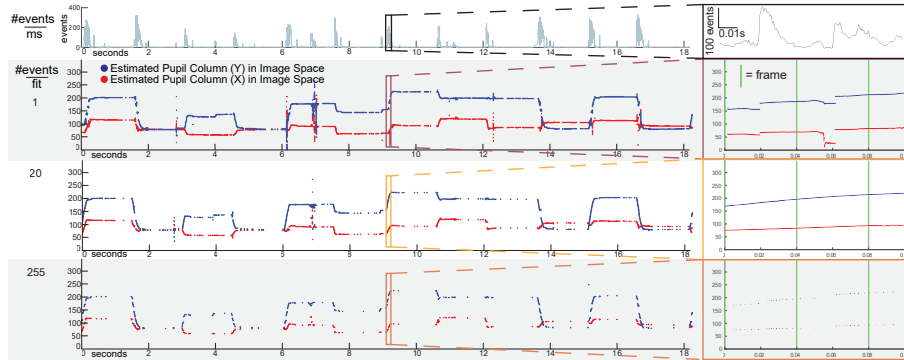
Fig. 6: **The X-Y coordinates of the fitted pupil over time.** Our system can update the pupil's location as fast as every event (row 2), but by considering more events, we can improve robustness and sparsity (row 3). Updating too slowly, however, harms robustness again (row 4). In the top row, a plot 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 the number of events used for each fit. One pupil position is obtained per fit. Hence, fewer fits are produced in the lower plot (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.

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 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 7 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 optimal 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 6: this parameter has a good balance between sparsity, robustness, and speed.

Assessing 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, $< 0.5°$ [14], within a standard field of view, which degrades to $2°$ on a larger field (see bottom row of Figure 7). 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-
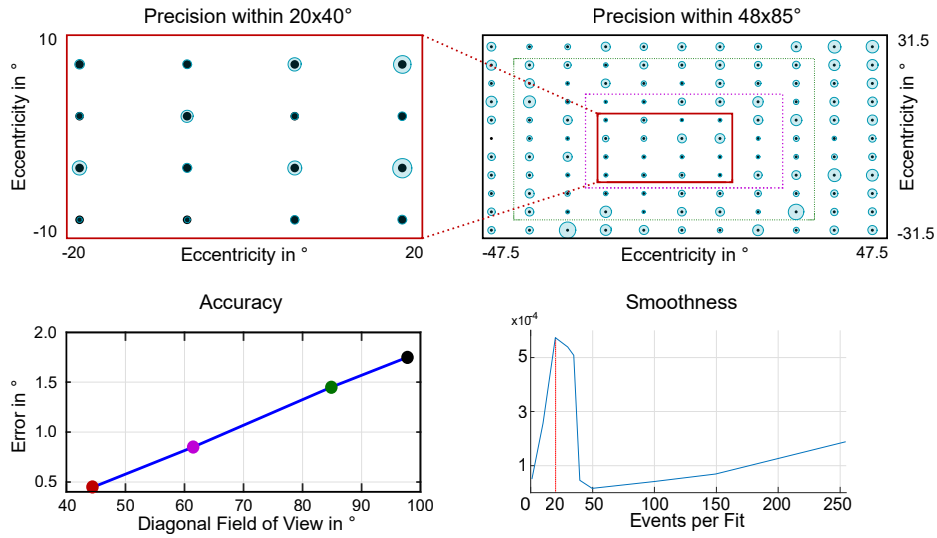
Fig. 7: **Precision and accuracy of our system** depend on the target field of view. The circle diameters (left) represent the precision, defined as the trimmed standard deviation of estimated gaze points centered at their ground truth label, averaged across all subjects. The average precision is $1.6°$ 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, $1.3°$. The best accuracy of a single subject (center-right) ranges from $0.45–1.75°$ for diagonal fields of view between $45–98°$. On the right is a plot illustrating the smoothness, calculated using the ℓ_2-norm of the finite difference gradient approximation, 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

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 6), 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 6 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.

Assessing Precision We calculate precision as the standard deviation, in visual angle, of the estimated pupil centers and plot it on the top row of Figure 7 across all subjects. 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 5. In Figure 7, 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.

We quantitatively assess the precision of our system in Figure 7. A second-degree polynomial is fit on the best-subject for both a small $20 \times 40°$ (top left) and larger $63 \times 95°$ (top right) field of view. The total FoV $64 \times 96°$ spanned by the screen in our experiments is comparatively much larger (and harder to regress) than previously reported in the literature ($26 \times 40°$ etc.) [14], 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 $10\times$ 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 [15]. 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 portable setting may bottleneck tracking speed or consume too much power.

The advantage of using an event-based sensor in near-eye gaze tracking does not solely lie in increasing the update rate at which the pupil position is estimated, but also in its "smart" sampling strategy. When the pupil is quasi-static, it induces no event, which means, in an interrupt based processing pipeline, no processing is "wasted." Thus, eye-tracking systems built on event-based sensors could potentially be much lower-power and have a smaller form factor. Additionally, event-based cameras have a dynamic range of $\approx 130dB$, allowing robustness to different lighting scenarios. Since the future of eye-tracking is undoubtedly in mobile, untethered, "always-on" applications for which power and form-factor are crucial, event-based vision sensors will likely be the devices to fit the bill. Such applications include AR/VR [4,50,20,48,29,57], vision science experiments [42], and the recently demonstrated autofocal eyeglasses [49]. Our system and the method we suggest are a first step in this direction and we are already working on miniaturizing it and embedding it on a low-power mobile platform.

Acknowledgements

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. Albert, R., Patney, A., Luebke, D., Kim, J.: Latency requirements for foveated rendering in virtual reality. ACM Transactions on Applied Perception (TAP) **14**(4), 25 (2017)
5. Baluja, S., Pomerleau, D.: Non-intrusive gaze tracking using artificial neural networks. In: Advances in Neural Information Processing Systems. pp. 753–760 (1994)

6. Cornsweet, T.N., Crane, H.D.: Accurate two-dimensional eye tracker using first and fourth purkinje images. JOSA **63**(8), 921–928 (1973)

7. Crane, H.D., Steele, C.M.: Generation-v dual-purkinje-image eyetracker. Applied Optics **24**(4), 527–537 (1985)

8. Damian, G.: Eye tracking using event-based silicon retina (2007)

9. Davison, A.J., Reid, I.D., Molton, N.D., Stasse, O.: Monoslam: Real-time single camera slam. IEEE Transactions on Pattern Analysis & Machine Intelligence (6), 1052–1067 (2007)

10. Delbrück, T., Linares-Barranco, B., Culurciello, E., Posch, C.: Activity-driven, event-based vision sensors. In: Proceedings of 2010 IEEE International Symposium on Circuits and Systems. pp. 2426–2429. IEEE (2010)

11. Delbruck, T.: Silicon retina with correlation-based, velocity-tuned pixels. IEEE Transactions on Neural Networks **4**(3), 529–541 (1993)

12. Dexl, A.K., Schlögel, H., Wolfbauer, M., Grabner, G.: Device for improving quantification of reading acuity and reading speed. Journal of Refractive Surgery **26**(9), 682–688 (2010)

13. Duchowski, A.T.: Eye tracking methodology. Theory and practice **328**(614), 2–3 (2007)

14. Ehinger, B.V., Gross, K., Ibs, I., Koenig, P.: A new comprehensive eye-tracking test battery concurrently evaluating the pupil labs glasses and the eyelink 1000. BioRxiv p. 536243 (2019)

15. 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.

16. Flickner, M.D., Koons, D.B., Lu, Q., Maglio, P.P., Morimoto, C.H., Selker, E.J.: Method and system for relevance feedback through gaze tracking and ticker interfaces (Jun 10 2003), uS Patent 6,577,329

17. Fuhl, W., Kübler, T., Sippel, K., Rosenstiel, W., Kasneci, E.: Excuse: Robust pupil detection in real-world scenarios. In: International Conference on Computer Analysis of Images and Patterns. pp. 39–51. Springer (2015)

18. Fuhl, W., Santini, T.C., Kübler, T., Kasneci, E.: Else: Ellipse selection for robust pupil detection in real-world environments. In: Proceedings of the Ninth Biennial ACM Symposium on Eye Tracking Research & Applications. pp. 123–130. ACM (2016)

19. Gehrig, D., Rebecq, H., Gallego, G., Scaramuzza, D.: Asynchronous, photometric feature tracking using events and frames. CoRR **abs/1807.09713** (2018), http://arxiv.org/abs/1807.09713

20. Guenter, B., Finch, M., Drucker, S., Tan, D., Snyder, J.: Foveated 3d graphics. ACM Transactions on Graphics (TOG) **31**(6), 164 (2012)

21. Hager, W.W.: Updating the inverse of a matrix. SIAM review **31**(2), 221–239 (1989)

22. Hansen, D.W., Pece, A.E.: Eye tracking in the wild. Computer Vision and Image Understanding **98**(1), 155–181 (2005)

23. Hu, Y., Liu, H., Pfeiffer, M., Delbruck, T.: Dvs benchmark datasets for object tracking, action recognition, and object recognition. Frontiers in neuroscience **10**, 405 (2016)

24. iniVation: Datasheet (2020), DAVIS346 Sensor, maximal value reported in the datasheet

25. Ishikawa, T.: Passive driver gaze tracking with active appearance models (2004)

26. Kim, H., Handa, A., Benosman, R., Ieng, S.H., Davison, A.J.: Simultaneous mosaicing and tracking with an event camera. J. Solid State Circ **43**, 566–576 (2008)

27. Kim, H., Leutenegger, S., Davison, A.J.: Real-time 3d reconstruction and 6-dof tracking with an event camera. In: European Conference on Computer Vision. pp. 349–364. Springer (2016)

28. Kim, J., Stengel, M., Majercik, A., De Mello, S., Dunn, D., Laine, S., McGuire, M., Luebke, D.: 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. Konrad, R., Angelopoulos, A., Wetzstein, G.: Gaze-contingent ocular parallax rendering for virtual reality. ACM Trans. Graph. **39** (2020)
30. Krafka, K., Khosla, A., Kellnhofer, P., Kannan, H., Bhandarkar, S., Matusik, W., Torralba, A.: Eye tracking for everyone. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 2176–2184 (2016)
31. Kundel, H.L., Nodine, C.F., Conant, E.F., Weinstein, S.P.: Holistic component of image perception in mammogram interpretation: gaze-tracking study. Radiology **242**(2), 396–402 (2007)
32. Kytö, M., Ens, B., Piumsomboon, T., Lee, G.A., Billinghurst, M.: 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. Lee, J.H., Park, P.K., Shin, C.W., Ryu, H., Kang, B.C., Delbruck, T.: Touchless hand gesture ui with instantaneous responses. In: 2012 19th IEEE International Conference on Image Processing. pp. 1957–1960. IEEE (2012)
34. Legge, G.E., Madison, C.M., Mansfield, J.S.: Measuring braille reading speed with the mn-read test. Visual Impairment Research **1**(3), 131–145 (1999)
35. Leutenegger, S., Lynen, S., Bosse, M., Siegwart, R., Furgale, P.: Keyframe-based visual–inertial odometry using nonlinear optimization. The International Journal of Robotics Research **34**(3), 314–334 (2015)
36. Li, Y., Wang, S., Ding, X.: Eye/eyes tracking based on a unified deformable template and particle filtering. Pattern Recognition Letters **31**(11), 1377–1387 (2010)
37. Lichtsteiner, P., Posch, C., Delbruck, T.: A 128x128 120 db 15us latency asynchronous temporal contrast vision sensor. IEEE journal of solid-state circuits **43**(2), 566–576 (2008)
38. Liu, H., Moeys, D.P., Das, G., Neil, D., Liu, S.C., Delbrück, T.: Combined frame-and event-based detection and tracking. In: 2016 IEEE International Symposium on Circuits and Systems (ISCAS). pp. 2511–2514. IEEE (2016)
39. Liu, H., Moeys, D.P., Das, G., Neil, D., Liu, S.C., Delbrück, T.: Combined frame-and event-based detection and tracking. In: 2016 IEEE International Symposium on Circuits and Systems (ISCAS). pp. 2511–2514. IEEE (2016)
40. Mahowald, M.: The silicon retina. In: An Analog VLSI System for Stereoscopic Vision, pp. 4–65. Springer (1994)
41. Mead, C.A., Mahowald, M.A.: A silicon model of early visual processing. Neural networks **1**(1), 91–97 (1988)
42. Mele, M.L., Federici, S.: Gaze and eye-tracking solutions for psychological research. Cognitive processing **13**(1), 261–265 (2012)
43. Mitrokhin, A., Fermüller, C., Parameshwara, C., Aloimonos, Y.: Event-based moving object detection and tracking. In: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). pp. 1–9. IEEE (2018)
44. Morimoto, C.H., Mimica, M.R.: Eye gaze tracking techniques for interactive applications. Computer vision and image understanding **98**(1), 4–24 (2005)
45. Mueggler, E., Rebecq, H., Gallego, G., Delbruck, T., Scaramuzza, D.: 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)
46. Neuhann, I.M., Lege, B.A., Bauer, M., Hassel, J.M., Hilger, A., Neuhann, T.F.: 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)
47. Ni, Z., Bolopion, A., Agnus, J., Benosman, R., Régnier, S.: Asynchronous event-based visual shape tracking for stable haptic feedback in microrobotics. IEEE Transactions on Robotics **28**(5), 1081–1089 (2012)

48. Padmanaban, N., Konrad, R., Stramer, T., Cooper, E.A., Wetzstein, G.: Optimizing virtual reality for all users through gaze-contingent and adaptive focus displays. Proceedings of the National Academy of Sciences **114**(9), 2183–2188 (2017)
49. Padmanaban, N., Konrad, R., Wetzstein, G.: Autofocals: Evaluating gaze-contingent eyeglasses for presbyopes. Science advances **5**(6), eaav6187 (2019)
50. Patney, A., Salvi, M., Kim, J., Kaplanyan, A., Wyman, C., Benty, N., Luebke, D., Lefohn, A.: Towards foveated rendering for gaze-tracked virtual reality. ACM Transactions on Graphics (TOG) **35**(6), 179 (2016)
51. Posch, C., Serrano-Gotarredona, T., Linares-Barranco, B., Delbruck, T.: Retinomorphic event-based vision sensors: bioinspired cameras with spiking output. Proceedings of the IEEE **102**(10), 1470–1484 (2014)
52. Ranjan, R., De Mello, S., Kautz, J.: Light-weight head pose invariant gaze tracking. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops. pp. 2156–2164 (2018)
53. Rebecq, H., Horstschaefer, T., Scaramuzza, D.: Real-time visual-inertial odometry for event cameras using keyframe-based nonlinear optimization. In: BMVC (2017)
54. Santini, T., Fuhl, W., Kasneci, E.: Purest: robust pupil tracking for real-time pervasive eye tracking. In: Proceedings of the 2018 ACM Symposium on Eye Tracking Research & Applications. p. 61. ACM (2018)
55. Schraml, S., Belbachir, A.N., Milosevic, N., Schön, P.: Dynamic stereo vision system for real-time tracking. In: Proceedings of 2010 IEEE International Symposium on Circuits and Systems. pp. 1409–1412. IEEE (2010)
56. Shank, M.D., Haywood, K.M.: Eye movements while viewing a baseball pitch. Perceptual and Motor Skills **64**(3_suppl), 1191–1197 (1987)
57. Sitzmann, V., Serrano, A., Pavel, A., Agrawala, M., Gutierrez, D., Masia, B., Wetzstein, G.: Saliency in vr: How do people explore virtual environments? IEEE transactions on visualization and computer graphics **24**(4), 1633–1642 (2018)
58. Smith, B.A., Yin, Q., Feiner, S.K., Nayar, S.K.: Gaze locking: passive eye contact detection for human-object interaction. In: Proceedings of the 26th annual ACM symposium on User interface software and technology. pp. 271–280. ACM (2013)
59. Smith, P., Shah, M., da Vitoria Lobo, N.: Determining driver visual attention with one camera. IEEE transactions on intelligent transportation systems **4**(4), 205–218 (2003)
60. SONY: Datasheet (2019), IMX387 Pregius sensor, SLVS – EC 8 Lane 12 bit 40.4 FPS, operating in typical conditions
61. Sugano, Y., Matsushita, Y., Sato, Y.: 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)
62. Tian, Y.l., Kanade, T., Cohn, J.F.: 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)
63. Vicente, F., Huang, Z., Xiong, X., De la Torre, F., Zhang, W., Levi, D.: Driver gaze tracking and eyes off the road detection system. IEEE Transactions on Intelligent Transportation Systems **16**(4), 2014–2027 (2015)
64. Wang, K., Ji, Q.: 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)
65. Weikersdorfer, D., Adrian, D.B., Cremers, D., Conradt, J.: 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)
66. Wood, E., Baltrušaitis, T., Morency, L.P., Robinson, P., Bulling, A.: 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)

67. Wood, E., Baltrusaitis, T., Zhang, X., Sugano, Y., Robinson, P., Bulling, A.: Rendering of eyes for eye-shape registration and gaze estimation. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 3756–3764 (2015)

68. Young, L.R., Sheena, D.: Survey of eye movement recording methods. Behavior research methods & instrumentation **7**(5), 397–429 (1975)

69. Zhang, X., Sugano, Y., Fritz, M., Bulling, A.: Mpiigaze: Real-world dataset and deep appearance-based gaze estimation. IEEE Transactions on Pattern Analysis and Machine Intelligence **41**(1), 162–175 (Jan 2019). https://doi.org/10.1109/TPAMI.2017.2778103

70. Zhu, W., Deng, H.: 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)

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

Anastasios N. Angelopoulos[1*], Julien N.P. Martel[2*], Amit P.S. Kohli[2], Jörg Conradt[3], and Gordon Wetzstein[2]

[1]University of California, Berkeley, [2]Stanford University, [3]KTH Royal Institute of Technology
gordon.wetzstein@stanford.edu

1 Formulation of the problem for $\mathcal{P}$ and $\mathcal{C}$

1.1 Updating and fitting the parabola $\mathcal{P}$

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

$$E_\mathcal{P}(\boldsymbol{p}) = E_\mathcal{P}(x, y) = 0 \tag{1}$$
$$\text{with} \quad E_\mathcal{P}(x, y) = a'\, y^2 + g'\, y + d' - x$$

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 $\mathcal{P} = (a', g', d')^\intercal$ and it can be fitted similarly to the ellipses representing the pupils by solving:

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

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

$$A' = \sum_{(x,y)\in\mathcal{D}_\text{img.}} v'^1_{x,y}\, v'^1_{x,y}{}^\intercal, \quad b' = \sum_{(x,y)\in\mathcal{D}_\text{img.}} v'^2_{x,y} \tag{3}$$

in which $v'^1_{x,y}$ is now:

$$v'^1_{x,y} = (y^2, y, 1)^\intercal \tag{4}$$

and $v'^2_{x,y}$ is different (due to the asymmetry in x and y):

$$v'^2_{x,y} = x\, v'^1_{x,y}{}^\intercal = (x\, y^2, x\, y, x) \tag{5}$$

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 greay 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:

$$
\begin{aligned}
\mathcal{D}'_{\text{img.}} = \{\boldsymbol{p} = (x,y)^\mathsf{T} \mid{} & \boldsymbol{p} \in \text{HarrisCorner} \circ \text{clip}(I, t_1, t_2), \\
& \text{and } \|\boldsymbol{p} - (x_e, y_e)^\mathsf{T}\|^2 < \rho', \\
& \text{and } y < \frac{\text{rows}}{2}\}
\end{aligned}
\tag{6}
$$

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

$$
\mathcal{D}'_{\text{evt.}} = \{\boldsymbol{p} \mid |E_{\mathcal{P}}(\boldsymbol{p})| < \delta\}
\tag{7}
$$

1.2   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 $\boldsymbol{p} = (x,y)^\mathsf{T}$ on the circle representing the glint in the image plane satisfy the quadric equation:

$$
\begin{aligned}
E_{\mathcal{C}}(\boldsymbol{p}) &= E_{\mathcal{C}}(x,y) = 0 \\
\text{with} \quad E_{\mathcal{C}}(x,y) &= x^2 - 2\,x\,c_x - 2\,y\,c_y + (c_x^2 + c_y^2 - r^2)
\end{aligned}
\tag{8}
$$

Hence we have to fit the parameters $\mathcal{C} = (c_x, c_y, r)^\mathsf{T}$. 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.

$$
\begin{aligned}
\mathcal{D}''_{\text{img.}} = \{\boldsymbol{p} = (x,y)^\mathsf{T} \mid{} & \boldsymbol{p} \in H_{t_3}(I(x,y)) \\
& \text{and } \|\boldsymbol{p} - (x_e, y_e)^\mathsf{T}\|^2 < \rho'\}
\end{aligned}
\tag{9}
$$

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

$$
\mathcal{D}''_{\text{evt.}} = \{\boldsymbol{p} \mid E_{\mathcal{C}}(x,y)^2 < \delta^2\}
\tag{10}
$$

2   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 $(x_e, y_e)^\mathsf{T}$ extracted from the parameters $\mathcal{E}$, and outputs the point $(x_s, y_s)^\mathsf{T}$ in screen coordinates that the user is supposed to look at. Thus, our regressor is a multivariate vector-valued function:

$$
G_\theta(\mathcal{E}) = \begin{pmatrix} x_s \\ y_s \end{pmatrix} = \begin{pmatrix} G_{\theta^1}|_x(\mathcal{E}) \\ G_{\theta^2}|_y(\mathcal{E}) \end{pmatrix}
\tag{11}
$$

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:

$$
G_{\theta^i}|_{x/y}(x_e, y_e) = \alpha_i\,x_e^2 + \gamma_i x_e\,y_e + \beta_i\,y_e^2 + \epsilon_i\,x_e + \zeta_i y_e + \eta_i
\tag{12}
$$

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

$$\arg \min_{\theta^1} \|G_{\theta^1}|_x(x_e, y_e) - x_s\|^2 \tag{13}$$

$$\arg \min_{\theta^2} \|G_{\theta^2}|_y(x_e, y_e) - y_s\|^2 \tag{14}$$

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

$$x_e = \frac{2\,b\,g - h\,f}{h^2 - 4\,a\,b}, \text{ and } y_e = \frac{2\,a\,f - h\,g}{h^2 - 4\,a\,b} \tag{15}$$

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

3   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:

$$\text{Accuracy} = \frac{1}{n} \sum_{i=1}^{n} l_i \tag{16}$$

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 $d_i = (\phi_i, \theta_i)$:

$$l_i = \|\hat{d}_i - d_i\|_2 \tag{17}$$

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 d_i's:

$$\text{Precision} = \sqrt{\frac{1}{n-1} \sum_{i=1}^{n} (\hat{d}_i - \bar{\hat{d}})^2} \tag{18}$$

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

4   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:

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

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

5   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.