Delft University of Technology
Master of Science Thesis in Computer and Embedded Systems Engineering

Efficient Eye Tracking and Gaze Estimation with Near-Eye Event Cameras

Konstantin Stefanov Mirinski

TU Delft Delft University of Technology

Efficient Eye Tracking and Gaze Estimation with Near-Eye Event Cameras

Master of Science Thesis in Computer and Embedded Systems Engineering

Embedded Systems Group
Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology
Van Mourik Broekmanweg 6, 2628 XE Delft, The Netherlands

Konstantin Stefanov Mirinski

3rd July 2026

Author
Konstantin Stefanov Mirinski
Title
Efficient Eye Tracking and Gaze Estimation with Near-Eye Event Cameras
MSc Presentation Date

July 7th, 2026

Graduation Committee

dr. Qing Wang (chairman)	Delft University of Technology
dr. Guohao Lan	Delft University of Technology
dr. Jie Yang	Delft University of Technology

Abstract

Eye tracking is a key enabling technology for wearable and extended-reality devices, but conventional frame-based systems struggle to capture the eye's rapid motion within the tight compute and power budgets of such hardware. Event cameras, which report per-pixel brightness changes asynchronously at microsecond resolution, are a natural fit for this setting, yet most event-based methods rely on heavy neural networks that are impractical to deploy on resource-constrained devices. This thesis presents a lightweight, training-free pipeline for near-eye gaze estimation that maps the tracked pupil to a point of gaze on the screen. The pupil is first detected in gray-scale frames using a purely geometric procedure of thresholding, morphological filtering, and ellipse fitting, and its center is then propagated between frames directly on the event stream by a points-to-edge template tracker, providing high-frequency updates without reconstructing an image. The pupil observation is mapped to gaze using a polynomial regressor. Evaluated on two near-eye datasets, the geometric detector matches a model-based baseline and approaches a supervised segmentation network, while the full system runs end-to-end in under a millisecond on a CPU - several orders of magnitude cheaper than learned alternatives. The work trades some gaze accuracy for the ability to run without a GPU or training data, at the cost of a small number of per-subject detection thresholds, offering a practical path toward efficient, deployable event-based eye tracking.

Preface

This thesis marks the end of my Master's journey at Delft University of Technology. It reflects a long process of learning - about eye tracking and event-based vision, but also about research itself: countless hours of debugging, reading and re-reading papers, refining ideas that did not work, and trying new ones until they did. The result on these pages is only part of the story; the rest is everything I learned getting here, and the people who supported me along the way.

First and foremost, I would like to thank my daily supervisor, Dr. Guohao Lan, for his patience and support throughout this thesis. I genuinely enjoyed our meetings - not least his jokes, which made even the more difficult stages of the project enjoyable - and his guidance was invaluable in developing this work and keeping it on the right track.

I am also grateful to my friends and my girlfriend, who made the journey through the Master's a joy regardless of the hardships that came with it. Their company and encouragement meant more than I can express here.

Finally, I would like to thank my family for their unwavering support throughout my studies, which has been the steady ground beneath every step of this journey.

Konstantin Stefanov Mirinski

Delft, The Netherlands
3rd July 2026

Contents

Chapter 1

Introduction

1.1 Research Background and Motivation

1.1.1 Eye Tracking and Applications

The human eye is one of the most dynamic organs in the human body. It is capable of executing rapid and precise movements that reflect underlying neurological and cognitive states of the human organism as a whole. There are three primary types of movements: saccades, fixations, and smooth pursuits. According to the definition given in [38], saccades refer to rapid movements of the eye that suddenly change the eye's focal point from one location to another, whereas fixation means the eye maintains a focus on a stationary target. During fixation, involuntary movements called microsaccades [30] may occur. On the other hand, smooth pursuit movements are much slower tracking movements designed to track moving targets along the line of sight. Eye-tracking technology leverages this natural behavior to extract eye features of interest [1], and subsequently determine the point of gaze (PoG) based on these features, transforming eye behavior into quantifiable, insightful data. Near-eye and full-face sensors capture the raw signals, while algorithms and models process them to recover the dynamic parameters of eye movement. This capability has established eye tracking as an active research domain with applications across many scientific and practical fields [11].

Eye tracking has proven valuable across diverse domains, where the precision, latency, and robustness of the system directly affect the outcome. The requirements differ across domains, as the following examples illustrate.

Healthcare and medical application. In this domain, eye tracking serves both as an intraoperative control signal and as a diagnostic instrument [26], [34]. In refractive surgery, laser ablation must compensate for involuntary eye motion that persists even under attempted fixations, so modern excimer laser systems incorporate high-rate eye trackers that reposition the beam or halt treatment when the eye moves outside an acceptable range. According to the study performed in [26], activating the eye tracking during LASIK [2] improved postoperative refractive outcomes relative to surgery performed with the eye tracker disabled. This illustrates the precision and latency demands of this setting.

Eye tracking is also being investigated as a non-invasive screening tool for neurodevelopmental and neurodegenerative conditions. For autism spectrum disorder, gaze-based measures such as preferential fixation on geometric over social content have been validated on large toddler cohorts as a screen for a subtype of the condition. In Parkinson's disease, oculomotor abnormalities such as hypometric saccades, prolonged saccadic latency, increased anti-saccade errors, and impaired smooth pursuit have been linked to the loss of dopamine-producing neurons that normally help control eye movement. Since eye tracking can measure these changes objectively, it has been proposed as a tool for detecting the disease early and tracking how it progresses. However, no single measure is diagnostic on its own, since some of these abnormalities also appear in related Parkinsonian disorders.

Driver or operator monitoring. A second, contrasting class of applications uses eye tracking to monitor the alertness of an operator engaged in a safety-critical task, most prominently driving, where fatigue and

[1] Such as the pupil, eyelid, or eyelash.

[2] LASIK (Laser-Assisted In Situ Keratomileusis) is the name of the surgery, which reshapes the cornea with a laser to correct vision.

inattention are leading contributors to accidents. The most established alertness measure is PERCLOS, the proportion of time within an observation window during which the eyelids are at least 80% closed, introduced and validated in early driving-simulator research [49]. PERCLOS and related indicators such as blink duration and eye-closure frequency rise with sleep loss and circadian misalignment [3], an effect confirmed across vigilance tests, simulated driving, and on-road driving [1]. Attention monitoring relies on a complementary signal, the direction of gaze relative to the road, which exposes distraction when the eyes leave the forward scene. In contrast to the surgical setting, these applications depend on comparatively slow ocular dynamics: eyelid droops, blink patterns, and prolonged fixations evolve over hundreds of milliseconds to seconds. They are typically aggregated over windows on the order of a minute. Conventional video frame rates, therefore, suffice to capture them, and a high sampling rate confers little additional benefit. This contrast is central to the present work, as it demonstrates that the value of high temporal resolution is determined by the application rather than being universal.

Extended Reality (XR). The most demanding wearable setting is XR (which includes both Virtual Reality (VR) and Augmented Reality (AR)), where the eye tracker is built into a head-mounted display worn directly in front of the eyes, a device that runs on a small battery with a limited compute budget. Gaze plays two roles here. It first acts as a way to interact with the scene, since a user can select or move virtual objects simply by looking at them, though this raises the classic Midas-touch problem of telling a deliberate selection apart from a glance that merely lands on an object [37]. Its second role is unique to XR. Gaze drives foveated rendering, where the headset renders only the small patch the user is looking at in full detail and lets the surrounding image stay coarse, which sharply lowers the cost of filling a wide, high-frame-rate display [14]. For this to work, the sharp region must stay locked to the eye. If it lags behind a fast saccade, the user glimpses the blurry periphery, and the illusion breaks. Albert et al. found that the system can hide a delay of only about 50 to 70 ms before such artifacts become noticeable [3], and staying within that budget during rapid saccades often means predicting where the eye will land before it arrives [5]. Tracking motion this fast and responding within milliseconds, all on a resource-constrained device, sets XR well apart from drowsiness monitoring.

Speed alone, however, is not enough in this setting. Because the display is worn on the head, it runs on a small battery with a limited compute budget, so the eye tracker must be not only fast but also light on power and processing. Here, conventional cameras struggle on two fronts at once. Their bandwidth and power limits hold realistic update rates to only a few hundred Hertz [4], and pushing them faster would only deepen the energy cost that a wearable can least afford. Satisfying both needs together, namely high temporal resolution and low resource consumption, is the central tension that motivates this thesis and shapes its second objective: making the approach practical on resource-constrained, wearable devices.

1.1.2 Eye Tracking Challenges

Beyond these applications, building an effective eye tracker raises several technical challenges. They arise because the eye moves extremely fast, so it must be both located accurately and sampled quickly enough to catch those movements. Three stand out: keeping up with fast motion, staying within a limited computational budget, and remaining reliable in uncontrolled conditions.

High-speed motion. Saccades can drive angular velocities above 700°/s [22], leaving only milliseconds to capture a shift in gaze and pushing the sensor toward high sampling rates and low latency. Spatial precision matters just as much, since the pupil occupies a small region that must be localized accurately to recover both deliberate and involuntary movements, and an off-axis camera in a head-mounted device makes that harder [51].

Computational efficiency. Eye tracking increasingly runs on wearables such as headsets and smart glasses, where computing capabilities and energy budgets are limited. Efficient algorithms and pipelines, therefore, become as decisive as accuracy for actual deployment.

Environmental robustness. Performance must survive changing illumination, reflections, occlusion by lids and lashes, anatomical differences between users, and confounds like glasses or make-up, any of

[3] Circadian misalignment is a mismatch between the body's internal clock and its actual sleep-wake schedule, such as that caused by night-shift work or jet lag, which promotes sleepiness during intended waking hours.

which can break feature detection [12]. Variation in users and tasks widens this gap further. A usable tracker has to handle this whole range, not just the conditions it was tuned on.

1.1.3 Progression from Traditional to Event-based Approaches

To address those challenges, eye tracking has progressed from invasive methods, such as the scleral search coil [29] and electrooculography (EOG) [39], towards non-invasive camera-based methods that now dominate human eye-tracking applications [16]. One of the most common commercial techniques is the Pupil-Center Corneal Reflection (PCCR) method [15], enabled by pairing an infrared (IR) light source with an IR-sensitive camera [44]. The IR light produces a clear pupil image along with a small reflection on the cornea, known as a glint, and gaze direction is recovered from the vector between the pupil center and that glint [20]. Other modern methods utilize pupil tracking, which identifies the pupil center and tracks it under IR illumination [2].

Yet, frame-based eye-tracking systems pose fundamental limitations. A conventional camera captures the full image on a fixed clock, and most commercial trackers run at only 50 to 100 Hz [41]. This sampling frequency is sufficient for slow movements, but a saccade happens within a handful of milliseconds. At these rates, it is sampled only a few times, and its trajectory is poorly resolved. Raising the frame rate is the obvious solution [10], yet it carries the whole image with it - every pixel is read out and processed on every frame, regardless of whether the eye has moved. Therefore, the cost rises in three ways at once: power consumption, data volume, and computation. These are exactly the resources that are limited on a mobile or XR headset. A faster frame camera thus buys temporal resolution only by worsening efficiency, leaving the underlying tension between speed, power, and precision unresolved. This trade-off is what motivates the usage of a sensor that captures fast motion in the intervals between frames, while remaining inactive when the eye is stationary.

Event cameras, also referred to as Dynamic Vision Sensors (DVS), take a fundamentally different approach and are a natural fit for near-eye tracking. Their design is bio-inspired, modeled on how the retina responds to change rather than recording static images. Unlike a conventional camera, which samples the whole sensor at a fixed frame rate, an event camera gives each pixel its own circuit that works independently and asynchronously [13]. A pixel reacts only when the brightness it sees (measured on a logarithmic scale) changes by more than a fixed threshold. At that instant, it emits an event: the pixel's coordinate, a timestamp accurate to microseconds, and the polarity of the change, meaning whether the brightness went up or down. Formally, let $L(x, y, t) = \log I(x, y, t)$ denote the log brightness at pixel (x, y) and time t. A pixel generates an event as soon as the change in L since its last event reaches a fixed contrast threshold C:

$$L(x, y, t) - L(x, y, t - \Delta t) = p\,C, \qquad p \in \{-1, +1\}, \tag{1.1}$$

where Δt is the time elapsed since that pixel last fired and the polarity p records the sign of the change. Each event is therefore a tuple $e_i = (x_i, y_i, t_i, p_i)$, and a recording of N events forms the stream

$$\mathcal{E} = \{e_i\}_{i=1}^{N} = \{(x_i, y_i, t_i, p_i)\}_{i=1}^{N}. \tag{1.2}$$

This operating principle gives event cameras several key advantages for eye tracking: microsecond-level latency, high temporal resolution, high dynamic range (above 120 dB), and significantly reduced power consumption and data throughput since static scenes are ignored [13]. Taken together, these properties make event cameras well-suited for capturing eye movements with minimal motion blur and high temporal precision, directly addressing the major limitation of their frame-based counterparts. Table 1.1 summarizes how event cameras address the speed, data, and power limitations of frame-based sensors.

The sparse, asynchronous form of event data calls for new tracking algorithms. Some methods accumulate events into frame-like representations so that standard tools such as convolutional neural networks (CNNs) can process them [19], occasionally combining these with periodic frames for added context [54]. Others fit a parametric model of the eye that covers the pupil and corneal glints, and update it as new events arrive [4]. Fully event-driven methods instead operate directly on the stream, achieving kilohertz update rates well beyond those of frame-based systems [27], [48], [40]. Recently, Spiking Neural Networks (SNNs) have attracted interest as a neuromorphic match for sparse, asynchronous data, with the promise of efficient and stable tracking [21].

Table 1.1: **Comparison of frame-based and event-based cameras.** Event cameras address the core limitations of frame-based sensors relevant to eye tracking: low temporal resolution, data redundancy, motion blur, and sensitivity to lighting conditions.

Property	Frame-based camera	Event camera
Sampling	Fixed frame rate, 50–100 Hz on commercial trackers (a few hundred Hz on high-speed models)	Asynchronous per-pixel, microsecond timestamps, effective kHz update rates
Latency	Bound by the frame period (tens of ms)	On the order of microseconds
Data output	Full frame every cycle, largely redundant when the eye is static	Sparse, only brightness changes are transmitted
Motion blur	Significant during saccades	Minimal
Dynamic range	Limited (around 60 dB)	High (above 120 dB)
Power and bandwidth	High, scales with frame rate and resolution	Low, scales with scene activity
Illumination	Often needs active IR for pupil contrast	Can track under ambient light

1.1.4 Research Gap and Motivation

Despite the recent advances in the field, current event-based eye tracking methods each come with their own trade-offs. For example, all methods from the AIS 2024 Challenge on the 3ET++ dataset [48] rely on end-to-end neural networks that are too computationally intensive for low-latency deployment on resource-constrained hardware. Similarly, while FACET [9] achieves strong pupil detection accuracy, it depends on high-end hardware and TensorRT optimizations, making it impractical for wearable deployment. E-Gaze [27], on the other hand, combines image-processing techniques for pupil tracking with a deep learning model for gaze estimation, achieving a good accuracy-frequency tradeoff, but its dependence on task-specific parameters does not allow generalization across subjects and settings.

A clear gap, therefore, remains between what current systems can do and what resource-constrained wearables actually require: tracking that is fast, accurate, and efficient enough. Closing this gap calls for lightweight event-based algorithms that fully exploit the temporal and spatial richness of the event stream while remaining practical to deploy.

1.2 Research Objectives and Contributions

This thesis project addresses the aforementioned challenges by building an event-based, near-eye gaze estimation pipeline that maps the tracked pupil onto on-screen gaze coordinates while preserving the ultra-high update rates and tight compute and power budgets that make event-based sensing attractive for next-generation, low-latency applications. The pipeline follows the conventional two-stage decomposition: the pupil is first localized within the camera frame, and its observation is then mapped to a point of gaze on the screen. Rather than treating either stage in isolation, the work is driven by a single overarching goal - to reach usable gaze accuracy under the compute and power budget of a wearable device - and pursues three concrete objectives toward it:

1. **Efficient, training-free pupil extraction.** Localize the pupil from gray-scale frames and propagate it at high frequency on the event stream, without learned weights or training data, and quantify how closely this approaches learned and model-based baselines.

2. **Accurate pupil-to-gaze mapping.** Translate per-frame and per-event pupil observations into on-screen gaze, and characterize how this mapping generalizes across subjects and across the field of view.

3. **Deployability on resource-constrained hardware.** Keep the end-to-end pipeline light enough in latency, parameters, and operations to run in real time on a CPU, and characterize the resulting accuracy-efficiency trade-off.

This work focuses on monocular eye tracking, relying on the assumption that the two eyes move together, a behavior known as conjugate eye movement [6]. In the near-eye setup, the eye fills most of the camera's field of view (FoV). Hence, the pupil, iris, eyelid, eyelashes, and eyebrow are the main features in the image, with minimal inference from the surrounding scene.

Scientific Contributions

The scientific contributions of this thesis address the objectives above and are summarized as follows:

- **Lightweight, training-free pupil detection and event-based tracking pipeline.** A purely model-based pipeline is proposed that detects the pupil from gray-scale frames via thresholding, morphological filtering, and ellipse fitting, then tracks the pupil center at high frequency between frames using a points-to-edge template tracker on the event stream. The pipeline requires no training data, runs end-to-end in under 1 ms on a CPU, though it relies on a small set of detection thresholds that are set per subject.
- **Scale-invariant outlier filtering for event-based pupil tracking.** Three reliability gates - blink exclusion, a residual gate, and a drift bound - are expressed as fractions of the mean pupil radius, making the tracker robust to eyelid and eyelash contamination.
- **Spatial analysis of gaze estimation error across the field of view.** A spatial heatmap analysis shows that the eye-to-screen mapping is most linear near the screen center and degrades toward the periphery, explaining where the polynomial regressor is reliable; restricting evaluation to a central $40^\circ \times 20^\circ$ window accordingly reduces the mean DoD by 40%.
- **CPU-only, event-based gaze estimation.** The complete pipeline is benchmarked against recent near-eye methods, reaching a comparable multi-kilohertz update rate on a CPU with several orders of magnitude fewer parameters and operations, where prior systems of similar frequency rely on GPU-class hardware or learned weights.

1.3 Report Outline

This thesis report presents the research in full: a review of the relevant literature, the design of the methodology, the implementation of the algorithms, and the experiments used to validate and evaluate the performance. It ends with a discussion of the results and the limitations. The remaining work is split across five chapters:

- **Chapter 2.** This chapter surveys the eye tracking literature, from pupil tracking and gaze estimation methods to available datasets, event data representations, and state-of-the-art algorithms.
- **Chapter 3.** This chapter describes the architecture of the proposed eye tracking and gaze estimation pipeline. It covers the three main stages: image-based pupil detection, high-frequency event-based pupil tracking between image frames, and gaze estimation.
- **Chapter 4.** This chapter evaluates the proposed system through a series of experiments covering pupil extraction, gaze estimation, and computational performance. It also describes the dataset, implementation details, and evaluation metrics used throughout.
- **Chapter 5.** This chapter discusses and interprets the experimental results, analyzing the performance of the pupil extraction and gaze estimation stages, the accuracy-latency tradeoff, and a comparison against other event-based eye tracking systems.
- **Chapter 6.** This chapter summarizes the key contributions and findings of the thesis, reflects on the limitations of the proposed approach, and outlines directions for future work.

Chapter 2

Related Work

2.1 Eye Tracking Approaches

Determining where a person looks is conventionally divided into two stages. First, the distinctive features of the eye, such as the pupil, the iris, and the corneal reflections (glints), are located in the image. Second, these features are processed into an input representation from which a gaze direction or an on-screen point of gaze (PoG) is estimated. The two stages are not equally forgiving: the precision of feature localization sets a ceiling on the accuracy of the gaze mapping that follows, since no downstream model can recover information that was lost when the pupil was first located. This thesis follows the same decomposition, tracking the pupil from near-eye recordings and then mapping the tracked pupil to a PoG. Following the survey of Hansen [16], the methods used in both stages range from explicit geometric modeling, which fits a parametric description of the eye, to data-driven learning, which maps pixels directly to the quantity of interest. The detection stage is conventionally split into model-based and appearance-based methods, while gaze estimation is more often divided into three families, as described below. The two extremes sit at opposite ends of this trade-off: model-based methods are interpretable and cheap to compute, while appearance-based methods are more robust but require higher computational capabilities.

2.1.1 Eye Tracking Paradigms

Locating the pupil is the foundation on which any gaze estimate rests, so it is worth examining how the two paradigms address it.

Model-based approaches. Model-based methods [28], [46], [4] rely on an explicit geometric description of the pupil, typically a circle or an ellipse, and recover its parameters from low-level image cues. Under the off-axis IR illumination common to near-eye trackers, the pupil appears as a dark, well-separated blob, which makes it amenable to classical image processing [42]. More elaborate edge-based variants, such as the family of detectors evaluated by Fuhl et al. [12], trace candidate pupil edges and reject outliers through robust fitting to cope with partial occlusion by the eyelids and lashes. The appeal of these methods is that they are interpretable, cheap to compute, and yield a compact parametric output ready for the gaze stage. Their weakness is fragility: they often rely on controlled illumination and hand-tuned thresholds that do not transfer across subjects or sessions, and they degrade under reflections, make-up, glasses, blinks, and occlusion.

Appearance-based approaches. Appearance-based methods [8], [25], [54], [48] instead learn a mapping directly from the raw pixels of the eye region to the quantity of interest, bypassing explicit feature detection. Modern instances are dominated by convolutional neural networks that either segment the pupil region or regress its location end to end. By learning from data rather than relying on fixed heuristics, these methods are markedly more robust to variation in eye shape, illumination, and occlusion. This robustness comes at a price: they demand large annotated datasets, benefit from high-resolution input, and carry a substantial computational and memory footprint, which is difficult to satisfy on resource-constrained wearables.

2.1.2 Frame-based Eye Tracking Methods

Both paradigms above have overwhelmingly been developed for conventional frame-based cameras, which sample the full sensor on a fixed clock. The vast majority of deployed eye trackers [36], [31], [25], [23] are of this kind, pairing a gray-scale or RGB camera with active IR illumination so that the pupil appears as a bright, high-contrast region that is easy to segment. This is also the setting in which the frame-based detection stage of this work operates. While such cameras are adequate for slow eye movements, they impose three structural limitations on tracking. First, their temporal resolution is bounded by the frame rate, typically 50 to 100 Hz, which is too coarse to resolve the trajectory of a saccade, during which angular velocity can exceed 700 degrees per second [22]. Second, fast motion induces motion blur that smears the pupil boundary precisely when accuracy matters most. Third, the sensor emits a full, largely redundant frame every cycle, inflating bandwidth, power, and computation regardless of whether the eye has moved. These are exactly the costs that a wearable device can least afford.

2.1.3 Gaze Estimation Approaches

Once the pupil and related features have been localized, the second stage maps them to a gaze direction or an on-screen PoG. The techniques fall into three families that, again, span the range from explicit geometric modeling to purely learned regression [16], [22].

Geometric, model-based estimation. Geometric methods build an explicit, often three-dimensional, model of the eye and derive gaze from the relative configuration of detected features. The most widespread commercial technique is Pupil-Center Corneal-Reflection (PCCR), in which one or more IR sources produce glints on the cornea, and the gaze vector is recovered from the displacement between the pupil center and the glints [15], [17]. With a calibrated geometry, these methods can be highly accurate and are relatively robust to head movement, but they require dedicated IR illumination, reliable glint detection, and careful per-user calibration of the eye model.

Interpolation- and regression-based estimation. Interpolation-based methods forgo an explicit eye model and instead fit a generic parametric mapping from the measured pupil coordinates to screen coordinates, with the parameters obtained from a short calibration in which the subject fixates known targets [22]. A common and effective choice is polynomial regression [4], [54], which expands the pupil coordinates into polynomial features and fits an independent regressor per screen axis. More expressive learned regressors, including recurrent networks such as LSTMs [27], can replace the fixed polynomial and exploit the temporal structure of the input, at the cost of more training data and more computational capabilities. Such mappings are simple to deploy and cheap to evaluate, which makes them attractive for resource-constrained settings; their main cost is a dependence on per-user, and often per-session, calibration, together with limited extrapolation beyond the calibrated region. This is the family adopted in the present work, where the tracked pupil is mapped to on-screen gaze through a learned regressor.

Appearance-based estimation. Appearance-based gaze estimation learns the mapping from eye, or even full-face, images directly to gaze using deep neural networks trained on large datasets collected under varied head pose and illumination, such as MPIIGaze [53] and ETH-XGaze [52]. These methods dispense with glints and explicit calibration geometry and generalize well across appearance, but they are data-hungry, computationally demanding, and can struggle on subjects and conditions unseen during training. While appearance-based gaze estimation is mature for frame-based cameras, its counterpart for near-eye event data remains comparatively underexplored.

2.2 Near-eye Event-based Eye Tracking Datasets

The quality of a learning-based eye tracking system is strongly tied to the quality and diversity of its training data [33]. Because event cameras are recent and near-eye recording is demanding, only a handful of datasets exist, and they differ substantially in scale, annotation, and the diversity of captured eye movements. Beyond scale and diversity, the label space of a dataset shapes how broadly the trained model can generalize: datasets with continuous gaze labels, such as EV-Eye [54], support general-purpose eye tracking but tend to yield higher prediction error, while those with discrete labels, such as EVB-Eye

[4], can achieve lower error but only within a narrow, task-specific set of gaze targets. The following describes the most prominent event-based eye tracking datasets.

EyeTrAES [40]. The dataset claims to target wearable and mobile applications, offering event-based recordings suited for low-latency tracking. However, its narrow field of view allows the authors to assume a circular rather than elliptical pupil shape, which limits its applicability to more general near-eye settings.

3ET++ (AIS 2024 Challenge) [48]. A purely event-based dataset created for a tracking competition. A major downside is that it provides only pupil-center annotations; the absence of pupil region and shape information limits its use in applications that require an explicit pupil contour or detailed PoG estimation.

EVB-Eye [4]. One of the pioneering datasets to pair gray-scale frames with event data for near-eye gaze tracking. Fitting an eye model from both modalities can map gaze directions. The dataset's point of gaze ground truth is limited to a sparse, discrete set of targets, making it highly task-specific and unsuitable for general-purpose gaze estimation. A further issue is that the ground-truth labels do not account for the latency between a target change and the eye's response, leaving some annotations unreliable without preprocessing. This dataset is used in the experiments described in this thesis.

EV-Eye [54]. This is the most comprehensive large-scale, multi-modal near-eye dataset to date, collected from 48 participants and comprising over 1.5 million gray-scale images and 2.7 billion events, together with gaze references from a Tobii Pro Glasses 3 tracker. It captures a wide range of natural eye movements, including fixations, saccades, and smooth pursuits. Its scale, subject diversity, and rich multi-modal annotation make it well suited for both developing and rigorously evaluating the pupil-tracking and gaze-estimation methods proposed here, and it is the second dataset used in this work. One limitation is that neither the gray-scale image stream nor the event stream is synchronized with the Tobii glasses used for ground-truth recording, causing misalignments between the sensor data and the gaze labels. Steps taken to mitigate this are described in Section 3.3.

2.3 Event Data Representation for Pupil Tracking

Raw event data is a sparse, asynchronous sequence of (x, y, t, p) tuples, a form that is incompatible with the dense, grid-structured inputs expected by conventional computer-vision and deep-learning pipelines [13]. Converting the event stream into a suitable representation is therefore a critical pre-processing step that directly shapes downstream accuracy, latency, and robustness. Several families of representations have emerged, each striking a different balance between simplicity and the preservation of temporal information.

- **Event frames.** The most straightforward approach accumulates events, sliced either by a fixed time window or by a fixed event count, into a two-dimensional histogram, so that each pixel encodes the aggregate activity at that location [48], [54], [27], [4]. These accumulated frames can then be used in different ways. The approach in [48] accumulates windows of 2000 events and applies conventional image processing directly to the resulting frames, fitting an ellipse to the pupil and thereby preserving its full geometry (height, width, and rotation angle). The method in [54], adopted in this research, instead accumulates much smaller windows of 20 events around the pupil and extracts only the pupil center, inferring the direction of eye movement from the polarity and location of the triggered events. The latter approach is considerably cheaper to compute, since it processes fewer events per event frame and avoids the sequence of image processing techniques and ellipse fitting, but this efficiency comes at the cost of discarding the pupil's geometric information, such as its shape and orientation.

  Event frames are simple to construct and compatible with standard image-processing and CNN tooling, but the accumulation collapses the precise timing within each window, sacrificing the fine-grained temporal dynamics that might be required for eye tracking [45].

- **Voxel grids.** Voxel-grid representations retain more of the temporal structure by partitioning the stream into several temporal bins, yielding a three-dimensional volume that preserves a coarse

ordering of events in time [56]. This recovers some of the temporal resolution lost by single event frames, at the cost of larger, multi-channel inputs that increase computation.

- **Time surfaces.** Time-surface representations encode, at each pixel, the timestamp of the most recent event, weighted by an exponential decay so that recent activity dominates [24]. This compactly captures recent motion history, but the decay is sensitive to the speed of the motion that produced it, so identical structures moving at different velocities yield markedly different surfaces, which complicates the learning of velocity-invariant features [55].

As mentioned above, the method developed in this thesis adopts the event-frame representation, following the high-frequency tracking strategy of EV-Eye [54]. Each frame-based pupil detection serves as a template that defines a ring-like region around the current pupil, and only the events falling within that region are accumulated into a frame, from which the pupil center is then extracted. The choice is driven primarily by computational efficiency: processing only a small batch of events around the pupil and estimating a single point, rather than building a dense representation over the full sensor, keeps the per-frame cost low. This aligns directly with the central objective of this research, namely the development of a lightweight eye tracking system.

2.4 State-of-the-art in Event-based Eye Tracking

Event-based eye tracking has emerged as a promising alternative to frame-based approaches, offering ultra-low latency, high temporal resolution, and reduced data redundancy. Existing systems, however, face a recurring trade-off between computational efficiency and tracking robustness. They can be grouped by how they exploit the event stream and how they prioritize different aspects of the problem.

Hybrid frame-event methods. EV-Eye [54] combines frame-based segmentation with an event-driven template-updating mechanism: after an initial pupil template is obtained from a frame, incoming events are matched to local edge segments and used to update the pupil center in real time, suppressing noise from irrelevant activity. This preserves microsecond-scale temporal resolution and improves stability over earlier work, but its reliance on a deep-learning-based pupil segmentation network partially forfeits the ultra-low-latency advantage, even though the U-Net it uses is relatively lightweight compared to other deep models. The pipeline in this thesis adopts a similar hybrid philosophy. It uses frames as templates and events for high-frequency updates, but relies on classical image processing to track the pupil from the gray-scale frames, which keeps computation lower but can reduce robustness under noise.

Pure event methods. E-Gaze [27] demonstrates high-frequency gaze tracking by applying morphological operations directly to event frames, achieving a favorable accuracy-frequency trade-off. Its performance, however, depends heavily on task-specific, hand-tuned parameters (example in Figure 2.1), which limits generalization, and its eye-part segmentation is vulnerable to adhesion between the pupil and surrounding structures such as the eyelid. EyeTrAES [40] similarly simplifies the problem by assuming a circular pupil and slicing events adaptively for low-latency tracking, an assumption that breaks down under oblique viewing and that underuses the temporal and polarity information of the stream.

End-to-end learned methods. The 3ET++ AIS 2024 Challenge [48] popularized end-to-end neural networks that regress pupil centers directly from raw event streams. These models achieve strong benchmark accuracy but are computationally intensive, which constrains latency and tracking frequency, and their lack of explicit geometric constraints makes them behave as black boxes that can fail under severe occlusion. FACET [9] narrows this gap by regressing ellipse parameters from causal event volumes, recovering detailed shape information, but its accuracy is obtained through high-end hardware and TensorRT optimization, which is unsuitable for wearable deployment. Neuromorphic alternatives based on Spiking Neural Networks [21] are a natural match for sparse, asynchronous data and promise efficient processing, but depend on bio-inspired hardware that is not yet widely available, while broader studies have begun to evaluate image-based face and eye tracking with event cameras more generally [19].

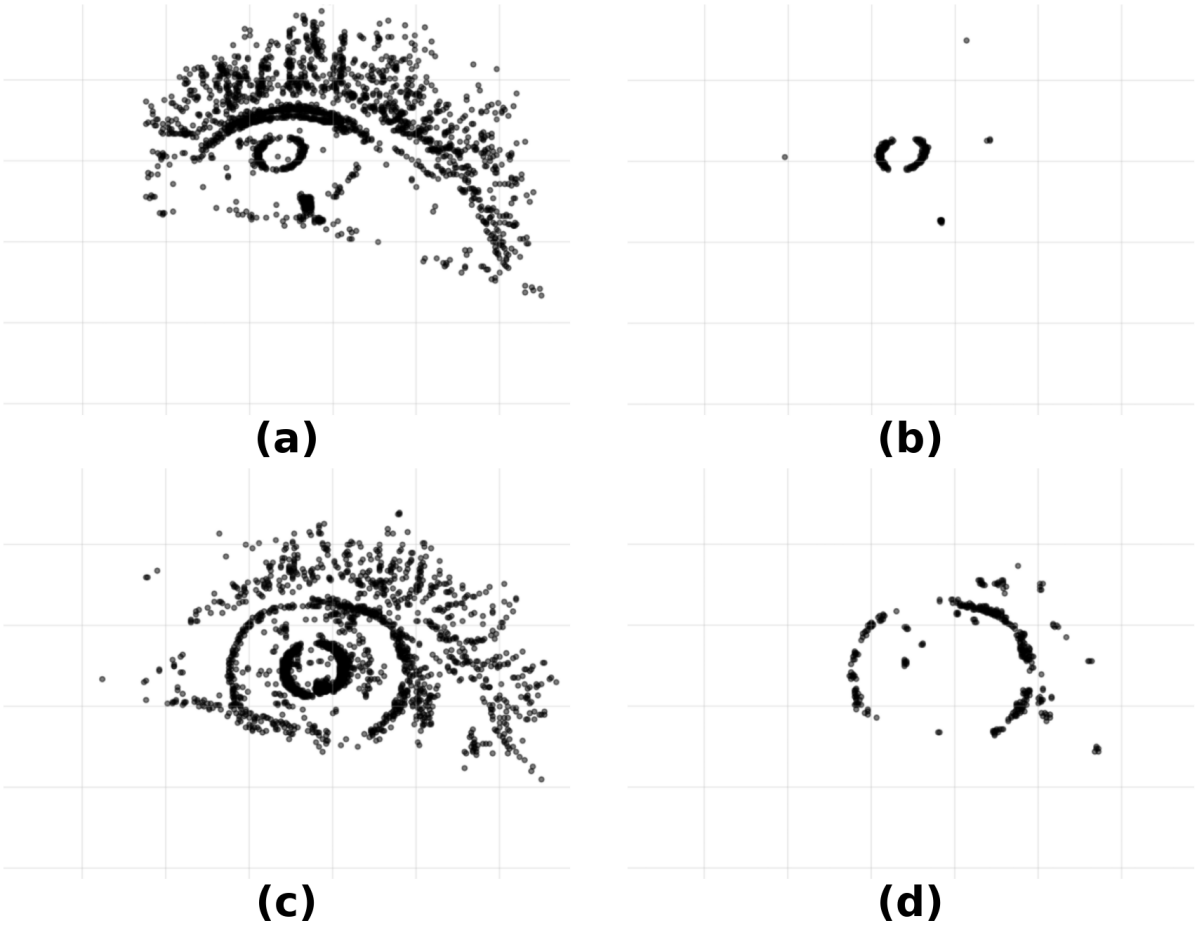


Figure 2.1: **Effect of the masking pipeline used in E-Gaze [27] on two arbitrary event sets of 2000 events each.** (a) and (c) show the raw event sets before filtering. (b) and (d) show what remains after applying the noise, eyelid, and eyelash masks with identical parameters in both cases. In (b), the masks isolate the pupil region cleanly, while in (d) the same parameters fail to recover a usable pupil signal. This shows that a single fixed parameter set does not generalize across event sets, so parameters must be hand-tuned.

The remaining gap. Across these systems, a consistent tension emerges. Methods that are accurate and fast tend to depend on either task-specific tuning or specialized, high-end hardware. This research targets that gap with an event-based pipeline that maps the tracked pupil to on-screen gaze while remaining lightweight enough for resource-constrained, low-latency deployment, and characterizes how well such a mapping generalizes across subjects and fields of view.

Chapter 3

Methodology

3.1 System Overview

This chapter outlines the architecture of the proposed eye tracking and gaze estimation system. It takes the gray-scale image frames and the event stream produced by the event camera and estimates the subject's PoG. Its design is driven by the complementary nature of the two modalities: the gray-scale images carry rich spatial appearance from which the pupil can be localized, but are produced at a low, fixed rate, whereas the event stream is sparse in appearance yet capable of tracking the pupil movement between two subsequent gray-scale images.

Figure 3.1 gives an overview of the complete pipeline, which is organized into three stages. The first stage, pupil detection from gray-scale image frames (Section 3.2), localizes the pupil in each frame by thresholding, morphological filtering, and contour fitting, yielding one pupil ellipse per frame together with a validity flag that discards unreliable detections. The second stage, event-based pupil extraction (Section 3.3), uses each frame detection as a template and updates it from the events that arrive before the next frame, producing a dense sequence of pupil ellipses whose effective rate far exceeds that of the frames. The third stage, gaze estimation (Section 3.4), maps the resulting pupil observations to PoGs.

3.2 Pupil Detection from Gray-Scale Image Frames

The first stage of the proposed pipeline performs pupil extraction from the image channel of the event data stream. In each frame, the pupil presents as a dark elliptical blob, a consequence of the off-axis infrared illumination. The pupil region is extracted by binarizing the input gray-scale frame with a constant threshold, suppressing noise through one or more successive morphological opening operations, and applying binary segmentation to identify candidate points along the pupil contour. This procedure is formalized as follows:

$$\mathcal{C}_{\text{img}} = \mathbf{K}\big(\gamma_{S_\sigma}^{n}\big(\tilde{H}_\theta(I)\big)\big) = \{C_1, \ldots, C_m\} \tag{3.1}$$

where I is the input image and $\tilde{H}_\theta$ is an inverted unit step that marks pixels below the threshold θ as foreground; γ denotes morphological opening with parameters S_σ for defining its structuring element, in our case a discretized circle parameterized by its radius σ, and n which denotes the number of successive morphological openings; $\mathbf{K}$ is a function that extracts the pupil contours by tracing the boundaries of the connected foreground regions in the binary image it receives, via border following [42].

Applying the operations above to a single frame yields a list of candidate contours $\mathcal{C}_{\text{img}} = \{C_1, \ldots, C_m\}$, each tracing the boundary of one connected dark region that remained after the thresholding and the morphological filtering. An ellipse is then fit to each candidate contour C_i [1], yielding a corresponding set of ellipses $\mathcal{E} = \{E_1, \ldots, E_m\}$. Under off-axis infrared illumination, the pupil is expected to be the most prominent dark region in the frame [17]. However, in practice, extraneous structures often produce large spurious candidates. For instance, the smart glasses frame consistently produces dark regions along the lower image boundary, while hair or other peripheral occlusions tend to cluster near the upper corners. Additionally, during a blink, the pupil may be partially or fully occluded, potentially leaving no geometrically valid candidate or only a small region in the frame. To suppress these artifacts, each ellipse E_i is subject to the following filters, and any candidate failing one or more criteria is discarded.

[1] A candidate contour is considered for ellipse fitting if it contains at least 5 points.

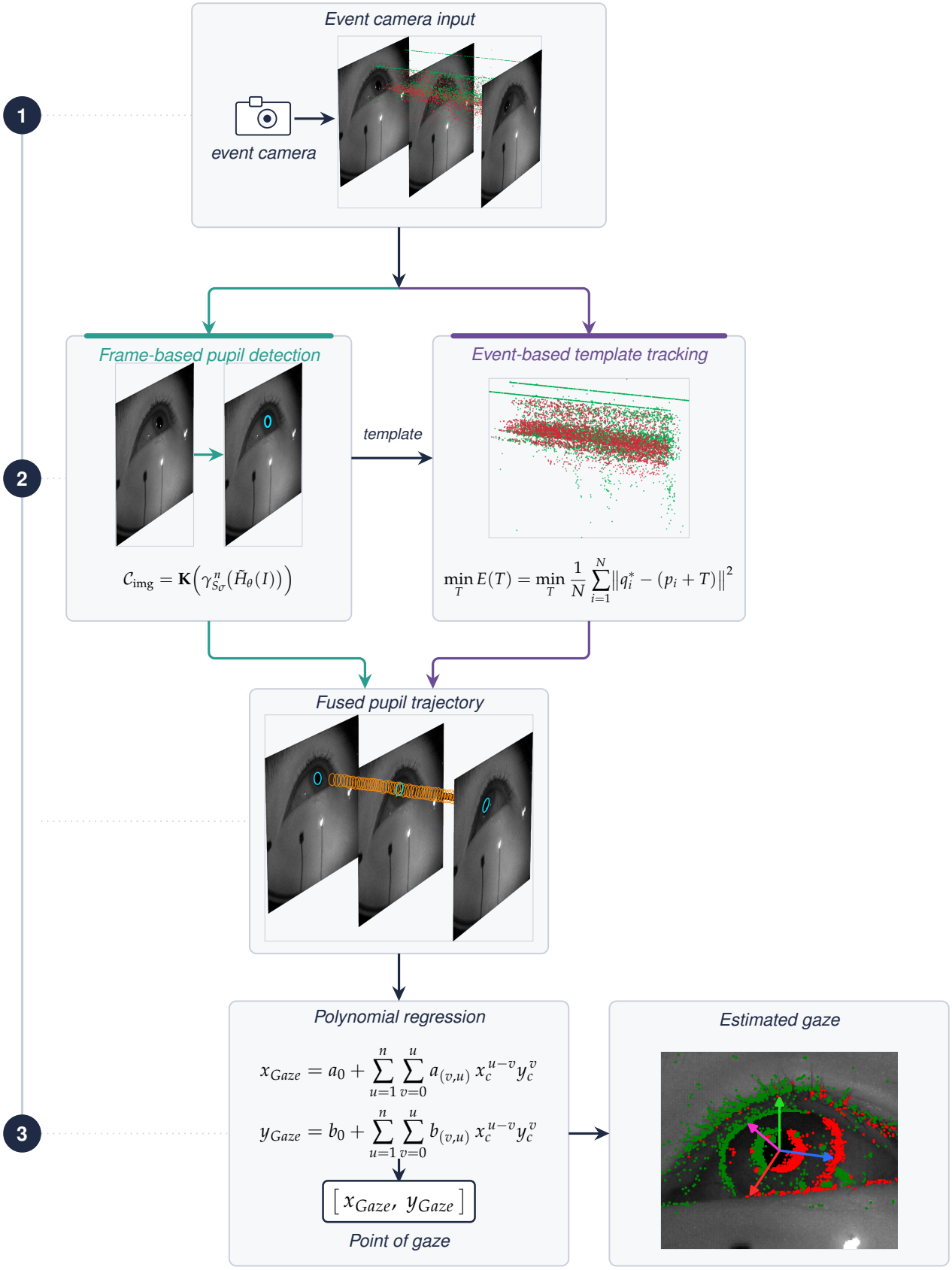


Figure 3.1: **Overview of the proposed eye tracking pipeline.** Starting from gray-scale image frames and event streams as inputs (1), the pipeline then performs model-based pupil detection on the image frames, followed by template-based pupil tracking using the event stream (2). Finally, the point of gaze (PoG) is estimated using a polynomial regressor (3).

1. **Corner triangle filter.** Pupil candidates whose center falls inside a triangular region cut from a chosen image corner are rejected. For a leg length t of a corner triangle, the excluded triangles are:

$$\begin{aligned} \text{upper-left:} \quad & c_x^{(i)} + c_y^{(i)} \leq t, \\ \text{upper-right:} \quad & c_x^{(i)} - c_y^{(i)} \geq W - t, \\ \text{lower-left:} \quad & c_y^{(i)} - c_x^{(i)} \geq H - t, \\ \text{lower-right:} \quad & c_x^{(i)} + c_y^{(i)} \geq W + H - t, \end{aligned}$$

   where $(c_x^{(i)}, c_y^{(i)})$ is the center of E_i, and W and H are the width and height of the image, respectively. A candidate ellipse is kept only if its center does not fall within any of the selected corner triangles. An example of how an ellipse is filtered out can be seen in Figure 3.2.

2. **Ellipse area filter.** The candidate ellipse is kept only if its area lies within an accepted range,

$$A_{\min} \leq A(E_i) \leq A_{\max},$$

   where $A(E_i)$ is the area of the candidate ellipse, $A_{\min}$ is the minimal allowed area, $A_{\max}$ is the maximal allowed area. The lower bound $A_{\min}$ rejects small spurious regions, such as the residual blobs left when the pupil is occluded during a blink. The upper bound $A_{\max}$ is applied only for the few subjects whose data contains large artifacts not captured by the triangle filter; otherwise, it is unused. An example of how the filter affects the pupil extraction can be seen in Figure 3.2.

3. **Ellipse aspect ratio filter.** Defining the aspect ratio as the ratio of the minor to the major axis,

$$\rho(E_i) = \frac{\min(w_i, h_i)}{\max(w_i, h_i)} \in (0, 1],$$

   The candidate is kept only if:

$$\rho(E_i) \geq \rho_{\min}.$$

   This rejects elongated ellipses that might be extracted from frames during blinks. An example of how this filter works can be seen in Figure 3.2.

Among the candidates that satisfy all criteria, the pupil is taken to be the one with the largest contour area. Its center is recorded as the pupil position and passed to the subsequent stage; frames for which no valid candidate remains (e.g., during blinks) are flagged as invalid and excluded from further processing. Figure 3.3 illustrates the intermediate results produced by each stage.

3.3 Event-Based Pupil/Ellipse Extraction

The gray-scale image frames are captured at only 25 fps, far too sparse to resolve the fast dynamics of rapid eye motion at our target latency. However, the event stream between two consecutive frames records every pixel-level intensity change induced by the moving pupil. This can be exploited by using each frame-based detection (Subsection 3.2) as a *template* and updating it from the events that arrive before the next gray-scale image frame, following the high-frequency tracking scheme of EV-Eye [54].

Let the most recent valid frame yield a pupil ellipse with center c and boundary $Q = \{q_1, \ldots, q_M\}$, obtained by sampling M points along the ellipse. We denote by $\bar{\gamma}$ the mean radius of the template, i.e., the average distance of the boundary points to the center,

$$\bar{\gamma} = \frac{1}{M} \sum_{j=1}^{M} \|q_j - c\|. \tag{3.2}$$

Candidate point selection. As the pupil moves, the events relevant to its displacement are those triggered along its boundary, while eyelid and eyelash motion produce events elsewhere in the frame. To isolate the former, we retain only events that fall within a ring-shaped region centered on the template; an event at position (x, y) is added to the candidate set P if:

$$\lambda_1 \bar{\gamma} < \|(x, y) - c\| < \lambda_2 \bar{\gamma}, \tag{3.3}$$

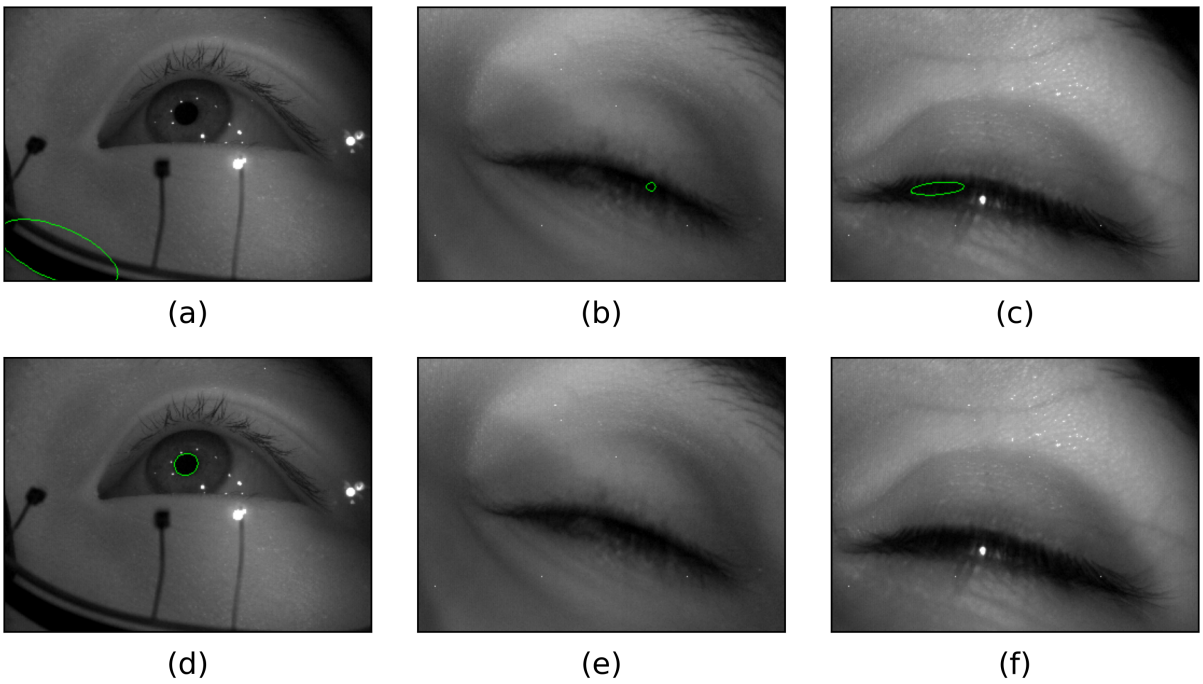


Figure 3.2: **Noisy pupil detections that are filtered out during preprocessing.** Each column pairs a noisy detection (top) with the result after filtering (bottom). In (a), part of the subject's glasses is mistaken for the pupil as the triangle filter fails, while in (d) it is detected correctly once the filter resolves the lower-left triangle. In (b), a blink makes the detection unreliable, and the area filter removes it, leaving none in (e). In (c), a blink produces an elongated ellipse that the aspect-ratio filter discards, leaving none in (f).

where $0 < \lambda_1 < 1$ and $\lambda_2 > 1$ are scale factors defining the inner and outer radii of the ring-shaped region. Incoming events are accumulated until the candidate set reaches a fixed size $N = |P|$, which forms a single tracking batch. N controls the temporal granularity of the tracker, as each batch produces one updated pupil estimate. A small N yields a higher update rate but a noisier estimate, while a large N averages over more events but updates the pupil position less often.

Points-to-edge matching. Each batch is used to estimate the translation T that best aligns the candidate events with the template boundary. Because the events are generated by the pupil shifting in the camera plane, the candidate set P is, to a first approximation, a translated copy of the boundary Q: we therefore seek the single translation T that minimizes the ℓ_2 matching error,

$$\min_T \; E(T) = \min_T \; \frac{1}{N} \sum_{i=1}^{N} \|q_i^* - (p_i + T)\|^2, \tag{3.4}$$

where, for each candidate $p_i \in P$, $q_i^* \in Q$ denotes its nearest boundary point. Since the correspondences q_i^* themselves depend on T, the translation is found by iterating two steps:

1. **Assignment.** For every candidate p_i, find its nearest boundary point $q_i^* \in Q$ by a nearest-neighbor search, and compute its horizontal and vertical offset to that point.
2. **Update.** Compute the mean offset over all candidates, $\Delta T = \frac{1}{N} \sum_{i=1}^{N} (q_i^* - p_i)$, shift every candidate by it, $p_i \leftarrow p_i + \Delta T$, and accumulate it into the running translation, $T \leftarrow T + \Delta T$.

The two steps are repeated until the candidates have settled onto the boundary, that is, until an additional step changes the translation by less than a small fraction τ of its current magnitude (we use $\tau = 0.01$). The template center is then advanced opposite to the boundary translation,

$$c \leftarrow c - T, \tag{3.5}$$

while the ellipse axes and orientation are inherited from the frame template and held fixed; only the center is tracked between frames. The shifted ellipse, together with its timestamp, is emitted as a high-frequency pupil sample.

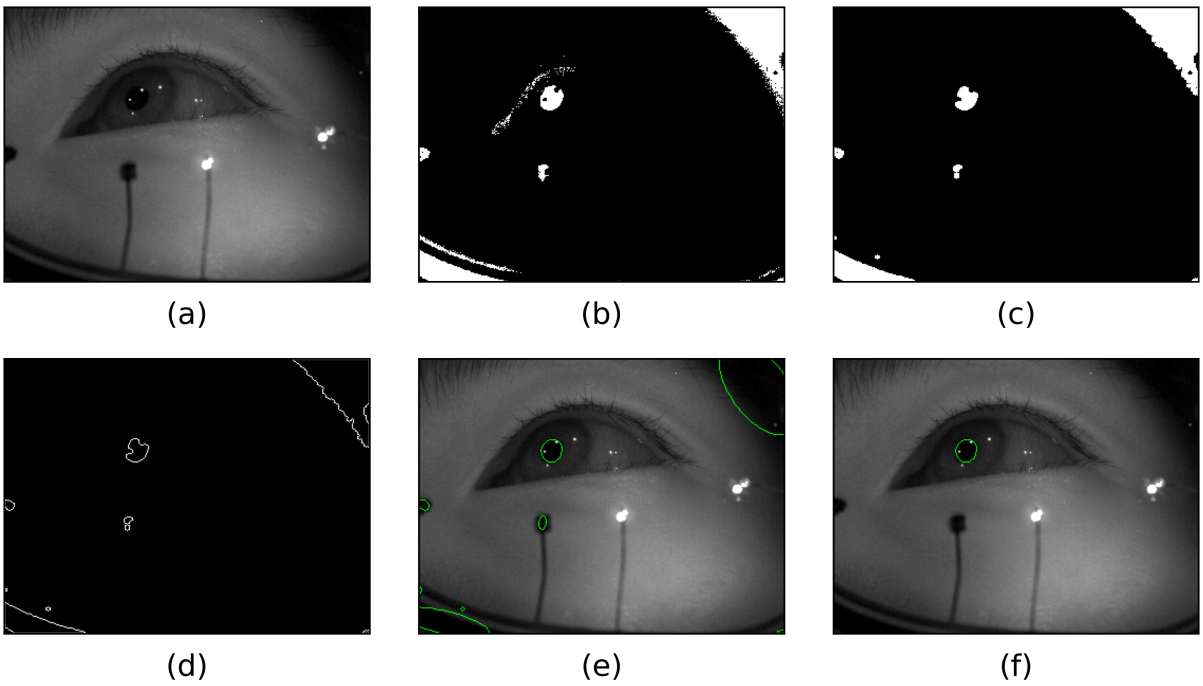


Figure 3.3: **Intermediate stages of the image-based pupil extraction pipeline.** Starting from the input gray-scale frame (a), thresholding (b) and morphological opening (c) isolate dark regions, from which contours (d) and ellipse candidates (e) are extracted. After filtering invalid candidates, the largest remaining ellipse is selected as the pupil (f).

Not every batch yields a reliable update. During a blink, the pupil is occluded, and eyelid motion can still populate the ring-shaped region with spurious events. To suppress these, each batch is subject to the following criteria, and any batch failing one or more is discarded. Throughout, we distinguish the template - the current ellipse estimate that is updated batch by batch from the events - from the anchor c_{anchor}, the center of the most recent gray-scale-frame detection.

1. **Blink exclusion.** Whenever the underlying frame detection fails (no valid ellipse, e.g., during a blink), the template is considered stale, and tracking is suspended until the next valid frame re-anchors it.

2. **Residual gate.** A batch is rejected if its mean points-to-edge residual after matching exceeds a fraction of the template radius, $\frac{1}{N}\sum_i \|q_i^* - (p_i + T)\| > \eta\,\bar{\gamma}$. A large residual indicates that the candidate events do not lie on the pupil ellipse and are therefore likely caused by eyelid or eyelash motion.

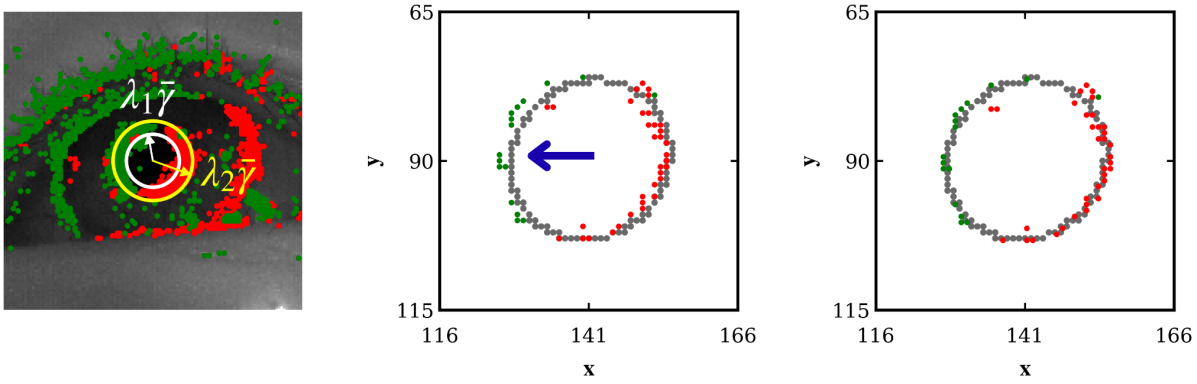
3. **Drift bound.** A batch is rejected if the updated center strays too far from the anchor, $\|c_{\text{new}} - c_{\text{anchor}}\| > \delta\,\bar{\gamma}$, preventing the template from drifting away during long event-only intervals.

Here η and δ are the residual and drift thresholds. They are expressed as fractions of the mean pupil radius $\bar{\gamma}$ rather than in absolute pixels, so that a single setting applies regardless of pupil size or subject. Figure 3.4 illustrates the candidate-selection ring-shaped region and the points-to-edge alignment of one batch.

Applying this procedure to the full event stream yields a dense sequence of pupil ellipses between successive gray-scale frames, raising the effective tracking rate well beyond 25 Hz while remaining anchored to the image-based detections.

3.4 Gaze Estimation Models

The preceding stages reduce each near-eye frame and each high-frequency event batch between frames to a pupil ellipse in the camera image. The final stage maps this pupil observation to a PoG on the screen that the subject is looking at. We consider two models for this mapping. The first is a polynomial regression



(a) Candidate subset (b) Before matching (c) After matching

Figure 3.4: **Example of candidate point selection and points-to-edge matching.** (a) The event points lying between the two concentric circles form the candidate point subset; (b) Points-to-edge matching, where the candidate point subset guides pupil tracking. In this example, the blue arrow is the moving direction of the pupil; (c) The outcome after the points-to-edge matching procedure.

[32] that maps the instantaneous pupil center to a PoG, following the model adopted in [54] and [4]. The second is a recurrent (LSTM) estimator [18] that additionally exploits the full ellipse geometry and the temporal context of consecutive pupil observations, following the model utilized in [27]. While the polynomial regressor maps each pupil center to a PoG independently, it disregards the ellipse geometry and the temporal continuity of consecutive observations, and we found it to generalize poorly in the cross-subject setting on the EVB-Eye dataset. To assess whether this limitation stems from the simplicity of the mapping itself, we additionally consider an LSTM estimator that exploits the full ellipse and a window of preceding observations. We use it as a comparison against the regressor; it is not part of the main pipeline, though it may replace the regressor in the final stage when greater modeling capacity is needed.

3.4.1 Polynomial Regression

This model maps the pupil center $c = (x_c, y_c)$ directly to the PoG through a pair of polynomial functions, one per screen axis. For a polynomial degree n, the gaze coordinates are:

$$x_{Gaze} = a_0 + \sum_{u=1}^{n} \sum_{v=0}^{u} a_{(v,u)} x_c^{u-v} y_c^{v} \tag{3.6}$$

$$y_{Gaze} = b_0 + \sum_{u=1}^{n} \sum_{v=0}^{u} b_{(v,u)} x_c^{u-v} y_c^{v} \tag{3.7}$$

The polynomial degree n is the only structural hyperparameter. Also, due to the subject heterogeneity, e.g., different kappa angles and cornea radii of human eyes, the coefficients are obtained through a subject-dependent calibration [54].

3.4.2 LSTM Gaze Estimator

The LSTM estimator consumes a short sequence of pupil ellipses and predicts the gaze point at the end of the sequence, allowing it to exploit the smooth, correlated trajectory of the moving eye.

Ellipse encoding. Each ellipse, parameterized by its center (c_x, c_y), axis lengths (w, h), and orientation ϕ, is encoded as a fixed-length feature vector. We form the augmented vector:

$$\mathbf{v} = [\, c_x,\ c_y,\ w,\ h,\ \phi,\ 1\,]^\top, \tag{3.8}$$

where the trailing 1 retains the first-order terms, and take the outer product $\mathbf{M} = \mathbf{v}\mathbf{v}^\top$. Since $\mathbf{M}$ is symmetric, its lower triangle (including the diagonal) contains the 21 unique entries, which form the

feature vector. This expansion supplies the network with all pairwise products of the ellipse parameters - this can be interpreted as the second-order polynomial features of the regressor, but applied to the complete ellipse geometry rather than the center alone.

Sequence construction. The encoded ellipses are grouped into fixed-length sliding windows of L consecutive samples in chronological order; the label of each window is the PoG coordinates of its *last* sample. A window is retained only if all L of its samples are valid (a successful detection passing the masks of the previous stages), which guarantees that each sequence represents an uninterrupted stretch of eye motion. The same construction applies whether the samples come from the gray-scale frame detections alone or from the merged stream of frame and high-frequency event ellipses (Section 3.3).

Chapter 4

Experiments & Results

This chapter covers the experimental setup and implementation details of the methods introduced in Chapter 3, along with a thorough evaluation of the proposed eye-tracking pipeline, organized into four sections.

Section 4.1 describes the experimental setup: the two datasets, the concrete configuration of the pipeline, and the metrics used throughout. The remaining three sections evaluate the pipeline stage by stage. Section 4.2 reports the accuracy of the pupil extraction, both from the gray-scale frames and from the event stream, against manually annotated ground truth. Section 4.3 evaluates the end-to-end gaze estimation on both datasets. Finally, Section 4.4 measures the per-stage and end-to-end latency and memory overhead of the system.

The computing device used in this work is equipped with a 16-Core Intel(R) Core(TM) Ultra 9 285H CPU and no GPU. The experiments were conducted in an environment running Python 3.12.13 and TensorFlow 2.21.0. The implementation code can be found *here*.

4.1 Experimental Setup

This section establishes the basis for the evaluation. Section 4.1.1 introduces the two datasets and their labeling, Section 4.1.2 gives the concrete parameter values used to instantiate the pipeline, and Section 4.1.3 defines the pupil-extraction and gaze-estimation metrics reported in the following sections.

4.1.1 Datasets

The proposed pipeline is evaluated on two publicly available event-based eye-tracking datasets, EVB-Eye [4] and EV-Eye [54]. Both are recorded with the DAVIS346 event camera and contain saccadic and smooth-pursuit eye motion in both left and right eyes. The main differences between them are their size and their labeling: EVB-Eye labels each frame with the known on-screen stimulus position, whereas EV-Eye provides independent gaze measurements from a commercial eye tracker. In the experiments, the focus is on the saccadic motion of the left eye.

EVB-Eye

EVB-Eye is a binocular saccade and smooth-pursuit dataset collected from 24 subjects. Two DAVIS346b sensors, mounted roughly 25 cm from each eye and fitted with 25 mm VIS-NIR lenses, simultaneously record 346×260 8-bit gray-scale frames at about 25 fps, and events. The eyes are illuminated with near-infrared light, and the head is held fixed on an ophthalmic headrest. Stimuli are shown on a 40-inch 1920×1080 monitor placed 40 cm away, spanning a field of view (FoV) of $96^\circ \times 64^\circ$. In the saccadic experiment, the subject fixates on a green cross presented for 1.5 s at each cell of an 11×11 grid (121 positions). Since the image channel operates at 25 Hz, each 1.5 s fixation yields 37-38 frames ($25 \cdot 1.5 = 37.5$) per target label. In the smooth-pursuit experiment, the subject follows a target moving along a predictable square-wave trajectory that sweeps the whole screen.

Ground-truth PoG is therefore the *known stimulus position*, encoded directly in each frame's filename as the target row and column. Moreover, each pupil extracted from the event stream inherits the stimulus label of the next gray-scale image frame in time, keeping it consistent with other works [27].

EV-Eye

EV-Eye is a substantially larger dataset collected from 48 subjects. It comprises over 1.5 million near-eye gray-scale images and 2.7 billion events from two DAVIS346 cameras (resolution 346×240, frames at 25 fps), together with 2.7 million gaze references from a Tobii Pro Glasses 3 eye tracker [43]. Notably, gaze in this dataset is *measured* rather than assumed: the Tobii tracker reports the point of gaze at 100 Hz (with $\sim 0.6^\circ$ angular error centrally) as normalized 2D screen coordinates in $[0, 1]$. The acquisition geometry mirrors EVB-Eye - a fixed headrest and a 1920×1080 monitor at 33 cm, giving a FoV of $95^\circ \times 63^\circ$ - with two saccadic sessions (an 11×11 fixation grid, 1.5 s per target) and two smooth-pursuit sessions (a dense square-wave trajectory). Because the Tobii reference is captured by an independent device, it must be temporally aligned to the image and event channel - each Tobii timestamp is converted to the DAVIS clock using the recorded absolute start time, $davis\_us = startime\_us + tobii\_s \times 10^6$. Each gray-scale image frame is assigned to the gaze of its nearest Tobii reference, and likewise, each pupil extracted from the event stream is labeled with the gaze of its closest Tobii reference. Any sample whose nearest Tobii reference is more than 15 ms away is discarded to avoid mislabeled targets.

4.1.2 Implementation Details

This section instantiates the pipeline of Chapter 3 with the concrete parameter values used in all experiments. The three stages - frame-based pupil detection (Section 3.2), event-based template tracking (Section 3.3), and the polynomial gaze regressor (Section 3.4.1) - are configured as described below. Unless stated otherwise, every parameter is held fixed across subjects, eyes, and datasets, and all runs use a single fixed random seed so that the data splits are reproducible.

Pupil detection from gray-scale frames

The detection filters of Section 3.2 are configured with the default values in Table 4.1. The morphological opening uses a single pass ($n = 1$) with an elliptical structuring element of radius σ, i.e., a $(2\sigma + 1) \times (2\sigma + 1)$ kernel. The corner-triangle filter excludes one or more image corners where dark non-pupil regions tend to appear, configured per eye and, for a few subjects, extended to additional corners. The upper area bound $A_{\max}$ is disabled by default and enabled only for the few subjects whose recordings contain large bright artifacts that the triangle filter does not remove. Among all candidates passing every filter, the one with the largest contour area is selected as the pupil.

Because illumination, eyelash density, and glint position vary across subjects, the detection thresholds are not identical for everyone: each evaluated subject overrides at least one of θ, σ, $\rho_{\min}$, or t from the defaults in Table 4.1, within the ranges listed there. These overrides affect only the detection stage; the downstream tracking and gaze parameters are shared by all subjects and are never tuned per subject. The overrides are fixed before evaluation and are not adjusted against the gaze metrics.

Table 4.1: **Frame-based pupil detection parameters.** Defaults are shared across subjects; the rightmost column gives the range spanned by the per-subject overrides.

Symbol	Parameter	Default	Per-subject range
θ	Binarization threshold	15	10–30
σ	Opening kernel radius (px)	3	1–4
n	Number of openings	1	–
t	Corner-triangle leg (px)	100	100–200
$\rho_{\min}$	Min. aspect ratio	0.32	0.25–0.38
$A_{\min}$	Min. ellipse area (px^2)	210	130–300
$A_{\max}$	Max. ellipse area (px^2)	disabled	enabled for few subjects

Gaze estimation models

Polynomial regressor. The polynomial mapping of Equations (3.6) and (3.7) is realized as two independent ordinary least-squares regressors, one per screen axis, fit in closed form on the polynomial-expanded pupil center with no regularization. The structural hyperparameter - the polynomial degree n - is swept over $\{2, 3, 4, 5, 6\}$, and results are reported per degree. To prevent the polynomial from extrapolating to

off-screen coordinates, predictions are clipped to the valid label range: pixel bounds $[0, W] \times [0, H]$ for EVB-Eye and the normalized $[0, 1]^2$ range for EV-Eye.

LSTM estimator. The recurrent estimator consumes sequences of $L = 10$ consecutive pupil observations, each encoded as a 21-dimensional vector, as explained in Section 3.4.2. A window is formed only when all Lframes in it are valid, guaranteeing temporal continuity, and is labeled with the PoG of the last ellipse. Similar to the architecture of the network proposed in E-Gaze [27], ours is a single LSTM layer of 128 units followed by three fully connected ReLU layers of decreasing width $(64, 32, 16)$ and a final linear layer that outputs the two PoGs. L_1 regularization ($\lambda = 10^{-4}$) is applied to the kernel weights of every layer. The model is trained with Adam under an exponentially decaying learning rate (initial 5×10^{-4}, decay 0.98 every 1000 steps), a batch size of 32, and a cap of 150 epochs; training is halted by early stopping on the validation loss with a patience of 20 epochs.

Data splitting and evaluation protocol

Not all subjects from each dataset are included in the evaluation: 11 subjects from EVB-Eye and 20 from EV-Eye were selected uniformly at random from the full populations, and all subsequent per-subject tuning and evaluation were performed on this fixed set. In this work, we evaluate under two protocols: a *single-subject* protocol, in which each subject is calibrated and tested on their own recording, and a *cross-subject* protocol, in which a shared model is trained across subjects and tested on a held-out one. In addition, the evaluation of gaze estimation on the high-frequency event stream is described.

Single-subject protocol. Each subject is calibrated and evaluated on their own recording, with no data shared across subjects, and both models are fit on the raw pupil coordinates without input standardization, since a single recording spans one consistent pupil-coordinate range. The split differs by model. For the polynomial regressor, the recording is divided into a calibration half and an evaluation half (50/50 by default; the fraction may differ in some experiments), using a leakage-free, dataset-dependent scheme. For EVB-Eye, where many frames share an identical stimulus label, the split is stratified per label, holding out the evaluation fraction of the frames of each of the 121 targets and fitting on the rest (similar strategy to [4]). For EV-Eye, whose Tobii labels are continuous and whose temporally adjacent frame and event samples are near-duplicates, a uniformly random split would leak near-identical samples across the two halves and inflate the reported accuracy; we therefore use a *temporal block* split, dividing the recording into 20 contiguous time blocks and assigning whole blocks to calibration or evaluation, so that no two temporally adjacent samples fall on opposite sides of the split. The LSTM, a sequence model trained by gradient descent, instead follows a chronological three-way split: the first 70% of the recording trains the network, the next 15% forms a calibration segment, and the final 15% is held out for evaluation. The calibration segment always serves as the validation set for early stopping during training; in the *fine-tuned* (calibrated) condition it is additionally used to adapt the trained network (reduced learning rate 2×10^{-5}, 20 epochs, recurrent layer frozen so that only the dense head is re-estimated), whereas in the *zero-shot* condition the trained network is evaluated directly. Because the evaluation segment is identical in both conditions, the fine-tuning gain is measured on the same held-out data.

Cross-subject protocol. A shared model is trained on the pooled data of all but one subject and evaluated on the held-out subject, repeating over all folds in a leave-one-subject-out scheme (11 folds for EVB-Eye). Because pupil-coordinate ranges differ across subjects, the pooled inputs are standardized with statistics computed on the training subjects only. Unlike the single-subject protocol, no fraction of the held-out subject is needed to train the model, since it is already trained on the other subjects. We report two conditions on the held-out subject. In the *zero-shot* condition, it is used purely for evaluation. In the *fine-tuned* (calibrated) condition, part of the held-out subject's data is used to adapt the shared model and the remainder for evaluation, with the calibration fraction drawn using the same leakage-free scheme as before — stratified per label for EVB-Eye and contiguous time blocks for EV-Eye. The two models differ in how much of the subject they use and how they adapt. The LSTM calibrates on a 40% fraction and is evaluated on the remaining 60%, fine-tuned exactly as in the single-subject protocol. The regressor instead reuses the same calibration/evaluation split as in the single-subject protocol and has no separate adaptation step: it is simply re-fit on the pooled training data augmented with the calibration fraction.

Event stream evaluation. The high-frequency event stream is evaluated only when using the regressor: the regressor needs only the pupil center, which the event-based extraction method provides, whereas the LSTM input encodes the full pupil geometry (center, width, height, rotation), meaning that our approach is not suitable for LSTM evaluation on the event tracking since it does not preserve that geometry. The polynomial is calibrated on the 25 Hz frame pupil centers of the calibration blocks and then scored both on the held-out frame centers and on the event-derived centers of the evaluation blocks. Reporting the frame and event DoD under one calibration isolates the accuracy cost of using the high-frequency event stream rather than the frame stream. For the field of view experiments, training and evaluation are restricted to a centered $40^\circ \times 20^\circ$ window, obtained by converting the angular extent to label units with the per-dataset pixels-per-degree factor.

4.1.3 Evaluation Metrics

Pupil Extraction Metrics

The pupil extraction stage is evaluated against the manually annotated ground-truth pupils. To compare a predicted ellipse with its annotation, both are rasterized into a filled binary mask at the sensor resolution: let $\hat{M}$ denote the set of pixels enclosed by the predicted ellipse and M the set enclosed by the ground-truth ellipse. We report three metrics.

Intersection over Union (IoU). The IoU measures the spatial overlap between the predicted and ground-truth pupil regions, as the ratio of the area of their intersection to the area of their union,

$$\mathrm{IoU} = \frac{|\hat{M} \cap M|}{|\hat{M} \cup M|}. \tag{4.1}$$

It ranges in $[0, 1]$, where 1 denotes a perfect overlap and 0 no overlap, and penalizes both missed pupil pixels and pixels wrongly assigned to the pupil.

F1 score (Dice coefficient). The F1 score, equivalent to the Dice coefficient for binary masks, also quantifies the similarity between the two regions but weights the overlap relative to the average region size,

$$\mathrm{F1} = \frac{2\,|\hat{M} \cap M|}{|\hat{M}| + |M|}. \tag{4.2}$$

It also ranges in $[0, 1]$ and is monotonically related to the IoU (F1 $\geq$ IoU), but is more forgiving of small boundary mismatches; we report both to remain comparable with other works [54].

Pixel error (PE). While IoU and F1 assess the predicted pupil *region*, the downstream gaze estimation consumes only the pupil *center*. The pixel error therefore measures the localization accuracy directly, as the Euclidean distance in pixels between the predicted center $\hat{c}$ and the ground-truth center c,

$$\mathrm{PE} = \|\hat{c} - c\|_2. \tag{4.3}$$

Unlike the overlap metrics, a lower PE is better, and it is reported in pixels at the sensor resolution.

Gaze Estimation Metrics

Predicted gaze points are evaluated against the ground-truth gaze targets. In this work, we utilize two metrics: the distance error (DE), which measures positional accuracy, and the Difference of direction (DoD), which measures angular accuracy. This metric is similar to the "gaze estimation error" in existing eye tracking literature [47], [52], [53].

Distance Error (DE). The distance error measures localization accuracy directly in the label space, as the Euclidean distance between the predicted and ground-truth gaze points,

$$\mathrm{DE} = \|\hat{g} - g\|_2, \tag{4.4}$$

where a lower value is better. We report DE only for the EVB-Eye dataset, whose labels are pixel coordinates on the stimulus display and therefore yield a distance with a concrete physical scale. For EV-Eye, the labels are normalized to the $[0, 1]$ FoV, so a Euclidean distance in this space carries no interpretable unit and is not directly comparable; there, we rely on the angular metric, described below.

Difference of Direction (DoD). It measures the angle, in degrees, between the predicted and ground-truth gaze directions. We model the eye as the origin of a pinhole camera and the display as a frontal plane that subtends a FoV of $\alpha_x \times \alpha_y$ degrees - the stimulus display for the EVB-Eye dataset, and the Tobii glasses' FoV for EV-Eye. A label is first expressed as a normalized coordinate $p = (p_x, p_y) \in [0,1]^2$ on this plane - used directly for the EV-Eye, and obtained by dividing the pixel coordinate by the display resolution for the EVB-Eye dataset - and then mapped to a unit viewing-ray direction,

$$\mathbf{d}(p) = \frac{\mathbf{r}(p)}{\|\mathbf{r}(p)\|}, \qquad \mathbf{r}(p) = \left((2p_x - 1)\tan\tfrac{\alpha_x}{2},\ (2p_y - 1)\tan\tfrac{\alpha_y}{2},\ 1\right). \tag{4.5}$$

The DoD is then the angle between the direction vectors of the predicted and ground-truth gaze points,

$$\mathrm{DoD} = \arccos\big(\mathbf{d}(\hat{g}) \cdot \mathbf{d}(g)\big), \tag{4.6}$$

reported in degrees, where a lower value is better.

4.2 Pupil Detection Performance

Pupil detection is evaluated against manually annotated ground-truth ellipses at both stages of the pipeline: the frame-based detector (Section 3.2) and the event-based template tracker (Section 3.3). We compare our pipeline against two baselines reported by EV-Eye [54]: the learning-based U-Net segmentation network and the model-based detector of EVB-Eye [4]. Frame-based detection is scored with IoU, F1, and center pixel error (PE); the event-based tracker, which has no per-event geometry ground truth, is scored with PE only. The aggregate results are summarized in Table 4.2, and the per-subject distributions in Figures 4.1 and 4.2.

On the frame-based task, our model-based detector matches the model-based EVB-Eye baseline, coming out marginally ahead on all three metrics - IoU 0.8432 vs. 0.8360, F1 0.9127 vs. 0.9075, and PE 1.05 vs. 1.30 px. A valid hypothesis may be that this improvement over EVB-Eye stems from our evaluation using fewer subjects than the EV-Eye benchmark, from the stronger candidate-filtering applied during pupil extraction (Section 4.1.2), or from a combination of both. As expected, the learned U-Net remains ahead (IoU 0.9187, PE 0.64 px), since it is a supervised segmentation network; our method closes most of the gap with a purely geometric pipeline. Figure 4.1 shows that this holds consistently across subjects rather than being driven by a few easy recordings.

Event-based pupil tracking. Because no event-wise ground truth exists, the event tracker is evaluated against the manually annotated frames, following the protocol of EV-Eye [54]. For each annotated frame, the tracker is initialized from the pupil detected in the immediately preceding gray-scale frame and then run forward over all events occurring between that frame and the annotated one, accumulating $N = 20$ events per update. The last pupil center produced before the annotated frame is compared against that frame's ground-truth center, and the PE is their Euclidean distance. Since the event tracker updates only the pupil center while inheriting the ellipse geometry from the template, the spatial overlap metrics IoU and F1 would not faithfully reflect tracking accuracy: they would largely measure the template's fixed shape rather than the tracked motion. PE, which depends solely on the center localization, is therefore the appropriate metric for the event stream.

Under this protocol, our template-matching tracker attains a PE of 1.58 px (Figure 4.2), close to the U-Net's 1.20 px and roughly 4.9× lower than the model-based EVB-Eye tracker's 7.70 px.

Method	Frame-based segmentation			Event-based
	IoU	F1	PE (px)	PE (px)
EV-Eye U-Net	0.9187	0.9560	0.64	1.20
Angelopoulos model-based	0.8360	0.9075	1.30	7.70
Ours	*0.8432*	*0.9127*	*1.05*	*1.58*

Table 4.2: **Pupil tracking (frame- and event-based) and results compared against manual ground-truth annotations.** Frame-based metrics are IoU, F1, and center pixel error (PE); event-based tracking has no per-event geometry ground truth, so only PE is reported.

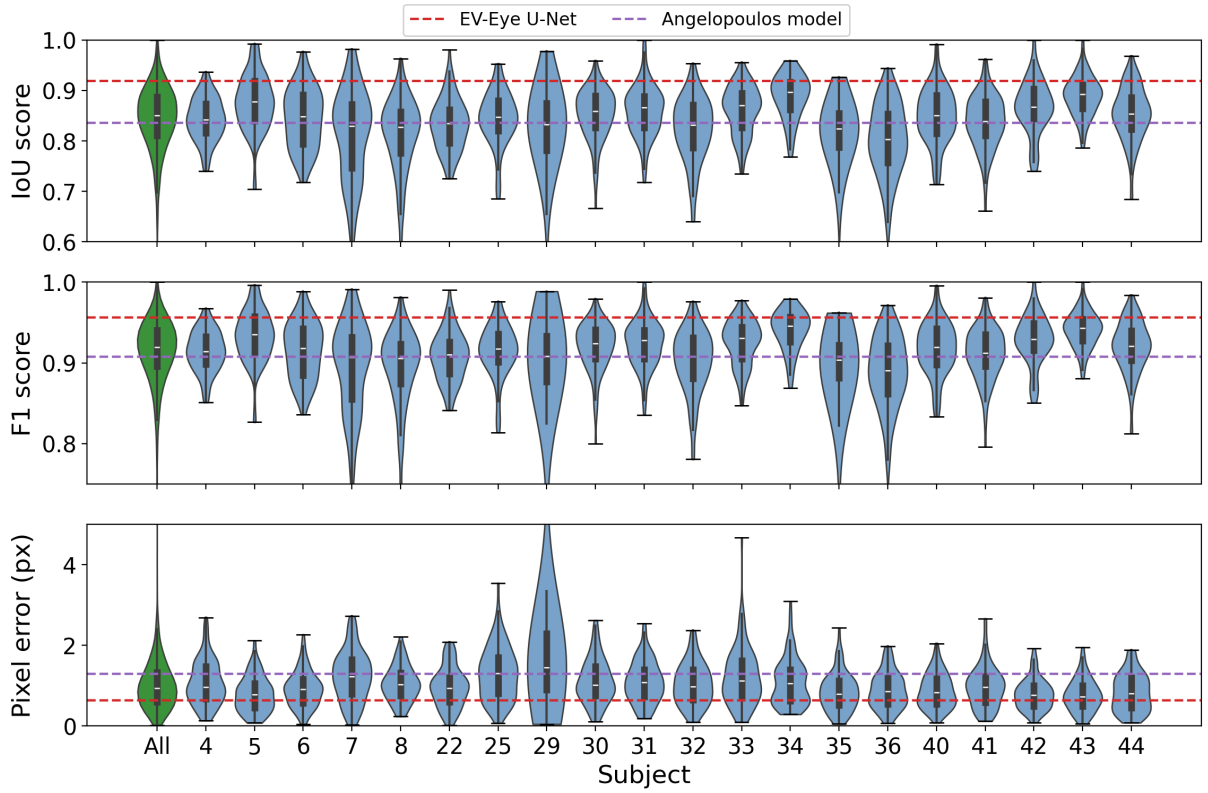


Figure 4.1: **Per-subject frame-based pupil detection accuracy.** Each violin shows the per-frame distribution of one metric for one subject, comparing our model-based detector against the manual ground-truth annotations: IoU and F1 (higher is better) and center pixel error (PE, lower is better).

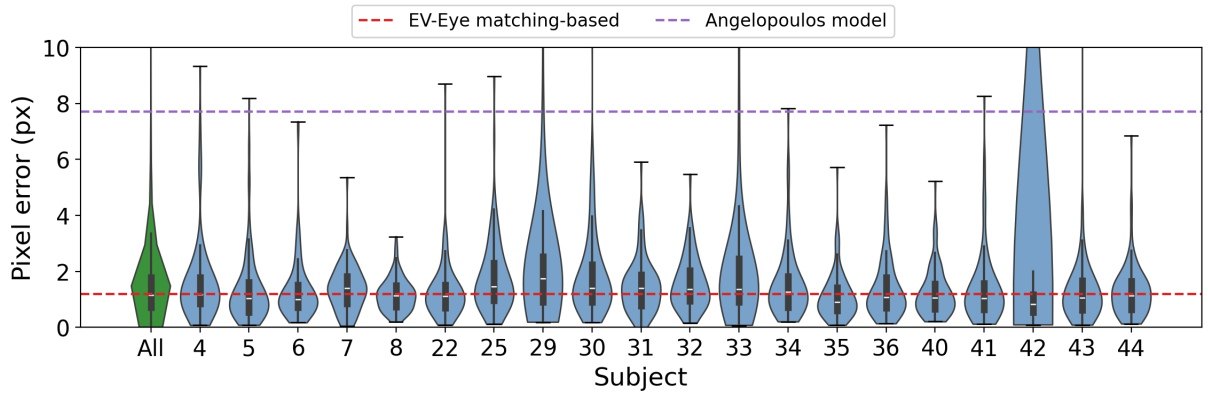


Figure 4.2: **Per-subject pixel error (PE) of the event-based pupil tracker.** PE is measured against the manually annotated frames following the EV-Eye protocol [54]: the tracker is initialized from the previous frame, run over the intervening events, and its last center is compared to the annotated center.

4.3 Gaze Estimation Performance

This section evaluates the end-to-end gaze estimation, from the extracted pupil to the predicted point of gaze, under the protocols and splits defined in Section 4.1.2. Accuracy is reported with the gaze estimation metrics of Section 4.1.3: the difference of direction (DoD) for both datasets, and additionally the distance error (DE) for EVB-Eye, whose pixel-coordinate labels give it a concrete physical scale. The two datasets are reported separately: Section 4.3.1 covers EVB-Eye and Section 4.3.2 covers EV-Eye.

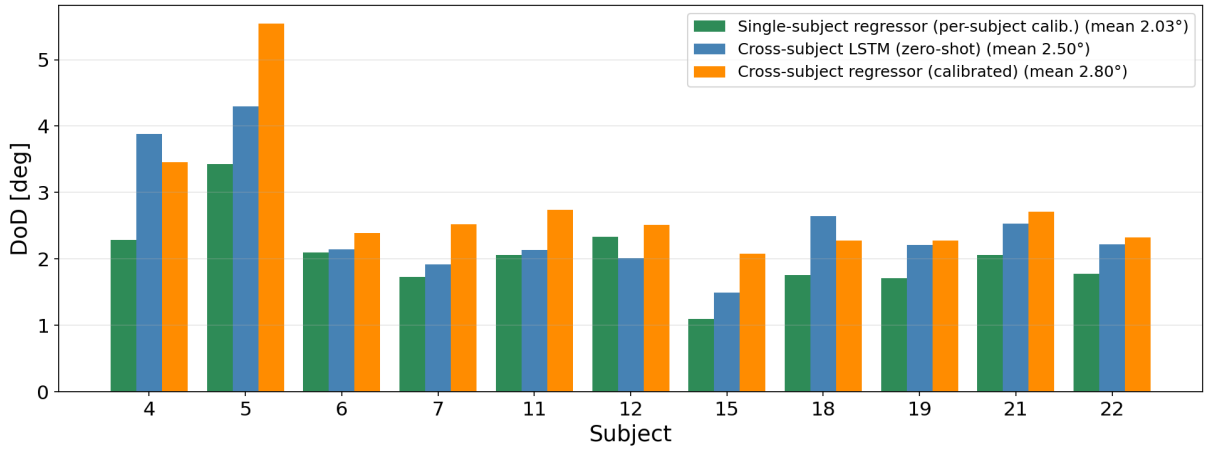


Figure 4.3: **Per-subject DoD on EVB-Eye for the three configurations.** Each group of bars is one subject; bars are the cross-subject LSTM (zero-shot), the calibrated cross-subject regressor, and the per-subject single-subject regressor. The single-subject regressor is best for nearly every subject, and subject 5 is the dominant outlier.

4.3.1 EVB-Eye

The results reported here are obtained from the gray-scale frame pupil centers only. The two models are compared on frames because the LSTM input is a 21-dimensional vector derived from the pupil geometry, and the event-based extraction does not recover this geometry independently: each event-extracted pupil inherits the ellipse shape and orientation of its anchor frame, which makes the geometric component of the LSTM input unreliable. Restricting the comparison to frames keeps the input valid for both models; the accuracy of the event stream itself is evaluated separately for the regressor in Section 4.3.2.

Table 4.3 summarizes the three configurations, and Figure 4.3 breaks the DoD down per subject. Across all configurations, subject 5 is a consistent outlier whose error is roughly twice the population median, which inflates every across-subject mean; the remaining subjects are tightly clustered.

Table 4.3: **EVB-Eye gaze accuracy across the three configurations.** Across-subject mean $\pm$ standard deviation over the 11 evaluated subjects.

Setting	Model	Calibration	DoD (°)	DE (px)
Single-subject	Regressor (degree-6)	per-subject	$\mathbf{2.03 \pm 0.55}$	$\mathbf{39.7 \pm 10.6}$
Cross-subject	LSTM	none	2.50 ± 0.81	48.5 ± 14.8
Cross-subject	Regressor (degree-6)	calibrated	2.80 ± 0.93	54.7 ± 17.6

In the cross-subject setting, the recurrent estimator outperforms the polynomial regressor even though it receives no calibration data from the held-out subject: the zero-shot LSTM reaches a mean DoD of 2.50°, against 2.80° for the regressor that is additionally fine-tuned on a calibration fraction of the held-out subject (Section 4.1.2). This is attributed to the richer input the LSTM consumes: rather than mapping a single instantaneous pupil center, it exploits the full ellipse geometry and the temporal context of consecutive observations, which lets it learn a shared eye-to-screen mapping that transfers across subjects. The instantaneous, center-only regressor cannot capture this structure, and the small per-subject calibration fraction is not enough to close the gap. This accuracy comes at a cost, however: the LSTM is a substantially heavier model, with far more parameters and a recurrent forward pass per prediction, so it demands considerably more computational capacity than the closed-form polynomial regressor.

When each subject is instead calibrated on their own recording, the polynomial regressor is the most accurate of the three, with a mean DoD of 2.03° - below both cross-subject configurations. This is expected, as a subject-specific calibration fits the mapping directly to that subject's eye geometry and camera placement, removing the inter-subject variability the cross-subject models must generalize over, and a high-degree polynomial can still be fit reliably on held-out blocks of the same recording. The regressor depends on this calibration; however, without any per-subject data, it reaches a 2.88° cross-subject regressor mean, which is slightly worse than the calibrated one (2.80°). The LSTM is therefore

preferable when calibration is impractical, attaining the best zero-calibration accuracy, whereas the lightweight regressor is the better choice when using a per-subject calibration.

4.3.2 EV-Eye

A consistent pattern emerges across subjects: the gaze error is not uniform over the screen but grows from the center toward the periphery. We first quantify this with a field of view comparison (Figure 4.4), then break it down spatially for a representative subject (Figures 4.5 and 4.6). The results are obtained with a second-degree regressor.

Field of view restriction. Figure 4.4 compares the per-subject frame DoD over the whole field of view against a restriction to the central $40^\circ \times 20^\circ$ window. Consistent with prior works [23], [50], [4], [27], the experiments are conducted on this central $40^\circ \times 20^\circ$ FoV. Restricting the calibration and evaluation to this central window reduces the mean DoD from 10.98° to 6.61° - a 40% improvement - and the gain holds for every one of the 20 subjects. This indicates that a large part of the whole-FoV error originates from the screen periphery, where the eye-to-screen mapping is most nonlinear, and the calibration data is sparsest.

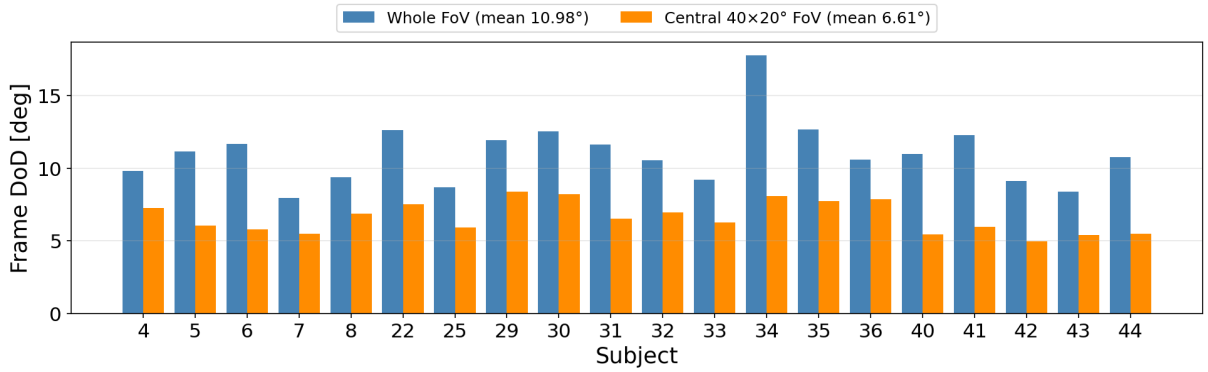


Figure 4.4: **Per-subject gaze accuracy over the whole FoV versus the central $40^\circ \times 20^\circ$ window (EV-Eye).** Each pair of bars is the mean frame DoD for one subject; restricting to the central FoV lowers the across-subject mean from 10.98° to 6.61° and improves every subject.

Spatial error distribution. To locate where the error accumulates, Figure 4.5 maps the mean DoD over a 6×6 grid of gaze-label positions, in degrees of field of view, for a representative subject (subject 7). The error is lowest in the central cells - around 4–8° near $(0, 0)$ - and rises sharply toward the edges and corners, exceeding 25° in some of the outermost cells. Figure 4.6 shows the corresponding number of evaluated (frame and event) ellipses per cell. The peripheral cells are more variable — some achieve low error, others are among the worst — but the central region is consistently the most accurate. This is expected, as the eye-to-screen mapping is most linear near the center of the FoV, where the gaze angle is small, and the polynomial regressor can fit it reliably.

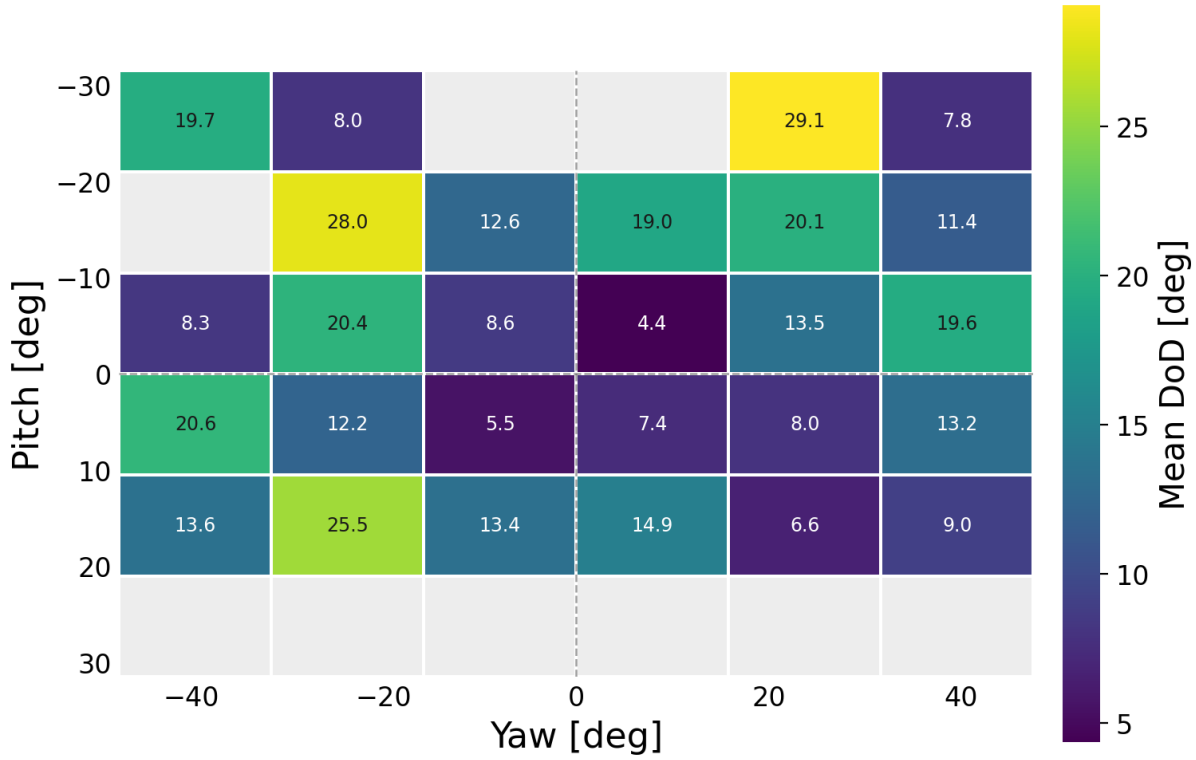


Figure 4.5: **Spatial distribution of gaze error (EV-Eye, subject 7).** Mean DoD per cell over a 6×6 grid of gaze positions in degrees of FoV (degree-2 regressor, blocks split). Error is lowest at the center and grows toward the periphery; gray cells contain no evaluation samples.

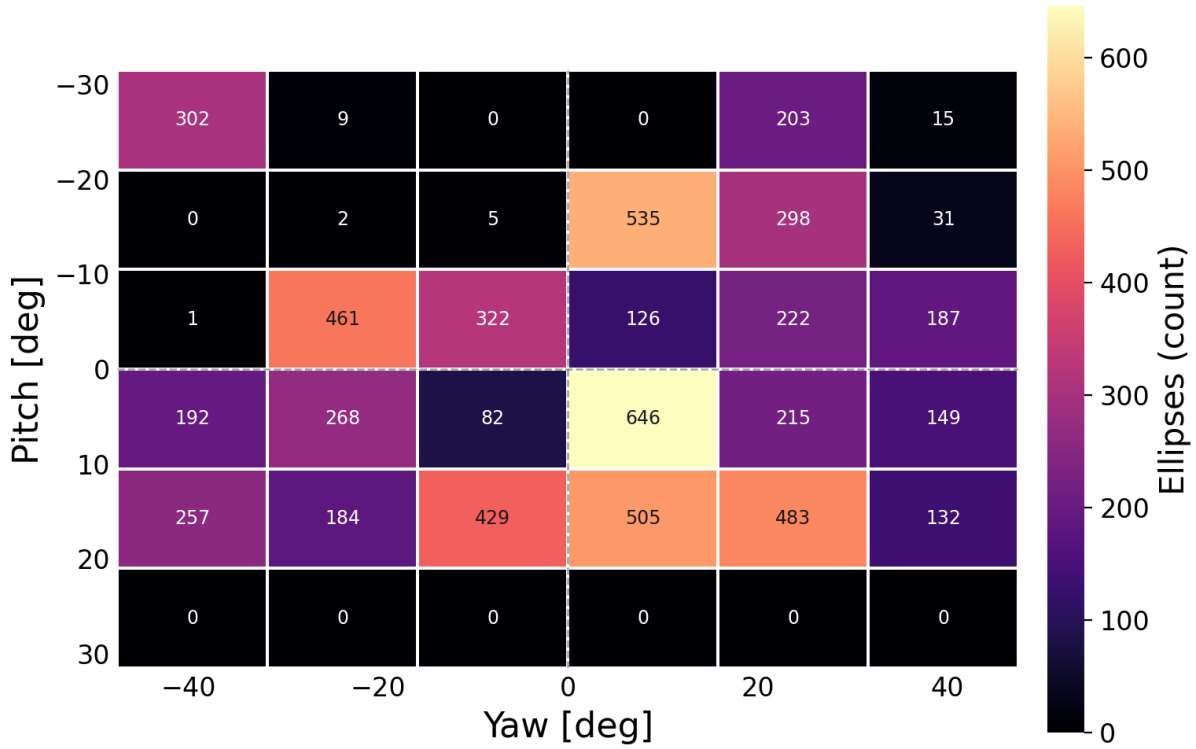


Figure 4.6: **Sampling density across the FoV (EV-Eye, subject 7).** Number of evaluated frames and event ellipses per cell, for the same grid as Figure 4.5. Samples concentrate near the screen center and thin out toward the periphery.

4.4 Latency and Efficiency Performance

A key motivation for using a classical, model-based pipeline over learned alternatives is computational efficiency. This section quantifies efficiency in two ways: the absolute per-stage latency of the proposed pipeline and a comparison of parameters and FLOP counts against recent event-based eye tracking methods.

Table 4.4 reports the per-stage processing latency and throughput of the pipeline, measured on EV-Eye subject 7 (left eye, saccadic), with image I/O excluded. The two pupil extraction stages are of comparable cost: frame-based detection takes 0.198 ms and event-based template tracking 0.222 ms. The polynomial regressor is the fastest stage at 0.087 ms. End-to-end, one full pipeline iteration — both pupil-extraction paths, the regressor, and the intermediate noise-flagging, masking, and normalization overhead — takes 0.656 ms, which is why it exceeds the sum of the three isolated stages. This corresponds to a throughput of 1524.9 Hz, well above the 25 Hz camera frame rate, meaning compute is not the bottleneck on the frame path.

Table 4.5 places these figures in the context of recent event-based eye tracking methods. Among the fully learned detectors, EV-Eye's detection network is the heaviest at 17.27 M parameters and 40.11 GFLOPs, with FACET (3.92 M, 3.44 GFLOPs) and TennSt (0.81 M, 5.49 GFLOPs) in the few-million-parameter, few-GFLOP range. The LSTM gaze estimator considered in this work is substantially lighter than the learned detectors (0.09 M parameters, 2.0×10^{-3} GFLOPs), but it does not do the same job as them: it replaces only the pupil-to-gaze mapping and reuses the same classical detection front end rather than a learned one. As a result, it stays within an order of magnitude of the classical regressor and several orders below the learned detectors, despite carrying roughly four orders of magnitude more parameters. The classical pipeline remains the lightest overall at 14 parameters and 4.7×10^{-4} GFLOPs - several orders of magnitude below the fully learned detectors in both metrics. This makes it suitable for deployment on resource-constrained hardware where such models would be impractical, at the cost of the accuracy gap reported in Section 4.2.

Table 4.4: **Per-stage processing latency and throughput of the proposed pipeline.** Measured on EV-Eye subject 7 (left eye, saccadic); image I/O is excluded. Frequency is the reciprocal of mean latency and reflects the maximum processing rate of each stage in isolation; it is computed from unrounded latencies, so it differs slightly from the reciprocal of the rounded millisecond values shown.

Stage	Mean latency (ms)	Frequency (Hz)
Pupil extraction from image	0.198	5044.6
Pupil extraction from events	0.222	4506.5
Pupil → Point of Gaze (regressor)	0.087	11452.7
Whole system (end-to-end)	0.656	1524.9

Table 4.5: **Computational cost comparison of event-based eye tracking methods.** Parameters (learned weights) and GFLOPs per inference are reported alongside a qualitative latency assessment. GFLOPs count each multiply and add separately; for the proposed pipeline, they cover float-only pupil tracking plus one gaze-model inference, and the non-parametric tracking stage contributes zero parameters. The proposed classical pipeline requires several orders of magnitude fewer operations than any learned alternative.

Methods	Parameters	GFLOPs	Latency & Compute Cost
EV-Eye [54]	17.27 M	40.11	High (Detection) / Low (Updating)
FACET [9]	3.92 M	3.44	Medium
TennSt [35]	0.81 M	5.49	Medium
Our pipeline with LSTM	0.09 M	2.0×10^{-3}	Low
Our pipeline with Regressor	14	4.7×10^{-4}	Very low

Chapter 5

Discussion

This chapter interprets the experimental results of Chapter 4, connecting them back to the research objective of an efficient and accurate event-based near-eye gaze pipeline. It discusses what the pupil-extraction and gaze-estimation results reveal about the design, situates the system among recent near-eye gaze estimators, and reflects on the accuracy-efficiency trade-off that defines the approach.

5.1 Pupil Extraction

The frame-based detector is competitive with the model-based EVB-Eye baseline on every metric (IoU 0.8432 vs. 0.8360, F1 0.9127 vs. 0.9075, PE 1.05 vs. 1.30 px), and the per-subject distributions in Figure 4.1 show that this holds consistently across subjects. A purely geometric, training-free detector thus reaches the accuracy of a hand-built model-based reference, while closing most of the gap to the supervised U-Net (0.9187 IoU, 0.64 px PE). The remaining gap to the U-Net is expected, since a segmentation network trained on annotated pupils can learn appearance cues that thresholding and ellipse fitting cannot. The practical takeaway is that, for the pupil-localization sub-problem, the accuracy cost of dropping the learned model is small, whereas the compute saving is large (Section 5.3).

The event-based tracker attains a PE of 1.58 px, close to the U-Net's 1.20 px and roughly 4.9$\times$ lower than the model-based EVB-Eye tracker's 7.70 px. This validates the central premise of the pipeline, namely that the pupil center can be propagated between frames directly on the event stream with sub-two-pixel error and without reconstructing an image. It also exposes a structural limitation: because the tracker inherits the ellipse geometry of its anchor frame and only updates the center, the event-derived observation carries a reliable center but not a trustworthy shape or orientation. This is precisely why the event stream could be scored only for the regressor and not for the geometry-dependent LSTM.

An important design parameter of the event tracker is the number of events N accumulated before each pupil update, fixed at $N = 20$ in all reported experiments. This value governs a direct trade-off between temporal resolution and spatial stability. A small N emits pupil estimates more frequently and with lower latency, since fewer events must be collected before an update, but the fit is then constrained by fewer event points and is more sensitive to event noise, raising the per-update pixel error. A large N averages over more evidence and yields a steadier fit, but lowers the effective update rate and, because the pupil may move appreciably while the batch is being filled, blurs fast motion across the accumulation window - partially defeating the high-temporal-resolution advantage that motivates using the event stream in the first place. The chosen $N = 20$ is consistent with existing research [4], [54] - it reflects a balance between these effects. To characterize this trade-off empirically, the tracker was additionally evaluated at $N = 12$ and $N = 40$. Increasing the batch to $N = 40$ yielded a slight accuracy gain, lowering the event-tracker pixel error from 1.58 px to 1.47 px, but reduced the event-driven update rate from 4506.5 Hz to 4019.9 Hz. Conversely, decreasing it to $N = 12$ raised the update rate to 5576.2 Hz at the cost of a higher pixel error of 1.80 px.

5.2 Gaze Estimation

The gaze results reveal a clear ordering between the different calibration regimes. On EVB-Eye, the per-subject regressor is the most accurate configuration at 2.03° DoD, below both cross-subject settings. This tells us that the dominant source of gaze error is not the simplicity of the mapping but the variation

between subjects: differences in eye geometry and in how the camera sits relative to each eye. Once a per-subject calibration is available to absorb that variation, a simple closed-form polynomial is enough, and nothing more elaborate is needed.

When the per-subject calibration is taken away, the picture changes in a revealing way. The zero-shot LSTM reaches 2.50° using no data at all from the held-out subject, yet still beats the cross-subject regressor at 2.80°, even though the regressor is given a calibration fraction of that subject. The reason is the richer observation the LSTM consumes: instead of a single pupil-center coordinate, it sees the full ellipse geometry and the temporal context of several consecutive frames, and from these it can learn a shared eye-to-screen mapping that generalizes across people. The instantaneous, center-only regressor simply has no way to represent that structure. This is exactly the behavior that motivated introducing the LSTM in the first place: the regressor's weakness across subjects is partly a limitation of model capacity, not only a lack of calibration. The catch is cost - the LSTM is several orders of magnitude heavier than the polynomial.

One configuration is deliberately missing from the EVB-Eye gaze evaluation: the regressor is never scored on the event-derived pupil centers of this dataset, because its labels cannot be trusted at event time. EVB-Eye supplies a single static gaze label per frame, so an event that fires between two frames is given the label of the following image. Events, however, exist precisely because the eye is moving - the pupil is mid-saccade when they are generated, far from the static target that the inherited label describes. The label thus records where the eye is about to land, not where it is looking at that instant. The effect is huge: calibrating and evaluating the regressor on a single subject's EVB-Eye events yields a mean DoD of roughly 23°, about a quarter of the display's 96° horizontal field of view and an order of magnitude worse than the 2.03° frame result. This is a labeling artifact rather than a failure of the tracker, and it is the reason the high-frequency event-stream gaze accuracy is reported only on EV-Eye, whose continuous Tobii references can actually be matched to the event timestamps.

The EV-Eye results add a spatial dimension to the analysis. Restricting evaluation to the central $40° \times 20°$ window lowers the mean DoD from 10.98° to 6.61° — a 40% reduction that holds for all 20 subjects — and the per-cell heatmap shows the error climbing from 4–8° near the center to above 25° at some of the edges. The cause is largely geometric: near the center the pupil faces the camera, and a change in gaze produces a large, clearly visible displacement of a round, well-formed pupil, whereas toward the periphery the pupil foreshortens into a thin, squished ellipse, so the same angular change maps to a smaller, noisier movement of its center. This is made worse by our event tracker, which carries only the pupil center between frames and discards the ellipse's width, height, and orientation, leaving it with the least information precisely where gaze is hardest to recover. A further factor sets EV-Eye apart from EVB-Eye: its Tobii labels are continuous, so the regressor must predict gaze over a dense range of positions rather than the fixed grid of discrete targets used in EVB-Eye. Together, these effects explain why prior near-eye work also limits evaluation to a central field of view, and they point to peripheral calibration density, mapping nonlinearity, and shape-aware tracking as the concrete levers for closing the gap.

5.3 Efficiency and Comparison with the State of the Art

The efficiency results are the strongest argument for the classical design. End-to-end, the pipeline processes one sample in 0.656 ms (1524.9 Hz), comfortably above the 25 Hz frame rate. In parameter and FLOP terms, the classical pipeline (14 parameters, 4.7×10^{-4} GFLOPs) sits several orders of magnitude below the fully learned detectors (EV-Eye at 17.27 M / 40.11 GFLOPs, and the lighter FACET and TennSt in the few-million-parameter range). Even the LSTM gaze head, at 0.09 M parameters, only replaces the pupil-to-gaze stage and still adds negligible FLOPs relative to a full detection network.

Table 5.1 places the system among recent near-eye gaze estimators along the axes that can be compared directly: input modality, system update frequency, and hardware platform. Reported gaze accuracy is deliberately omitted from the comparison, because each method evaluates it on a different dataset, field of view, and protocol, so the numbers are not directly comparable; the gaze accuracy of this pipeline is reported separately in Section 4.3. Along the comparable axes, the trade-off is clear: the methods that achieve high or kilohertz update rates do so on GPU-class hardware (NVGaze, EyeNet, E-Gaze, EV-Eye), or rely on a theoretical pupil-update rate (Angelopoulos et al.), whereas the proposed pipeline reaches a comparable 1,525 Hz on a CPU and without any training. The contribution is therefore not an accuracy record but a favorable position on the efficiency-deployability axis: a real-time, training-free, event-based estimator that needs neither a GPU nor learned weights, in the regime that matters for resource-constrained wearables.

Table 5.1: **Near-eye gaze estimation comparison.** Input modality, system update frequency, and hardware platform for recent near-eye methods. Reported gaze accuracy is omitted because each method evaluates it on a different dataset and protocol, making the values not directly comparable. Event / Frame indicates the input modality.

Method	Event	Frame	Freq. (Hz)	Hardware platform
EyeNet [50]		✓	83	NVIDIA 1080Ti GPU
NVGaze [23]		✓	1000	NVIDIA Titan GPU
Angelopoulos et al. [4]	✓	✓	10000	Theoretical
E-Gaze [27]	✓		950	NVIDIA 3070 GPU
EV-Eye [54]	✓	✓	2600	NVIDIA 3090 GPU
This pipeline	✓	✓	1525	CPU

Taken together, the three result groups tell one consistent story. The classical front end is competitive with a learned one for pupil localization - a fair comparison, since both are scored against the same manual annotations under one protocol - the gaze stage is bounded primarily by calibration and by the nonlinearity of the eye-to-screen mapping at the periphery, and the whole system is cheap enough to run on a CPU in real time. The efficiency comes at the price of accuracy relative to GPU-based estimators, and the next chapter discusses how that gap might be narrowed.

Chapter 6

Conclusion & Future Work

This thesis set out to build an event-based, near-eye gaze estimation pipeline that maps a tracked pupil to on-screen gaze while preserving the high update rate and low compute budget that make event sensing attractive for wearable, low-latency applications. The proposed system detects the pupil in gray-scale frames with a training-free geometric detector, propagates the pupil center between frames at high frequency on the raw event stream with a points-to-edge ICP tracker, and maps the pupil observation to a point of gaze with a polynomial regressor. Across two datasets, it matches the model-based pupil-detection baseline, closes most of the gap to a supervised segmentation network, and does so several orders of magnitude more cheaply end-to-end in 0.656 ms on a CPU, with no training and a parameter and FLOP budget significantly below any learned alternative.

6.1 Limitations

Several limitations temper these results. First, the pupil extraction is validated against only a small set of manually annotated frames - on the order of a hundred per subject - so the reported IoU, F1, and pixel-error figures rest on a limited sample of the full recordings and may not capture rarer failure cases. Second, the event tracker propagates only the pupil center between frames: the point-to-edge matching does not preserve the geometry of the pupil - its width, height, and orientation - so the high-frequency stream cannot supply the shape information that a geometry-aware gaze model would need. Third, neither dataset is an ideal benchmark. EV-Eye provides no true ground-truth gaze direction; its references come from a commercial Tobii eye tracker, which carries its own error on the order of 0.6° [43]. EVB-Eye, in turn, is not continuous and was designed for a task-specific stimulus: its labels switch to the next target instantaneously, which does not account for the subject's reaction time and the latency of the eye in actually moving there [7], so frames immediately after a target change are labeled with a gaze the eye has not yet reached. Fourth, the frame detector is not fully automatic: several detection parameters - such as the binarization threshold, the opening-kernel size, and the minimum aspect ratio - are tuned by hand on a per-subject basis to cope with differences in illumination, eyelash density, and glint position. This manual tuning limits how readily the pipeline scales to new users and recordings, since each requires some hand adjustment rather than working out of the box. Finally, the pipeline is evaluated only on recorded data, not deployed on head-mounted hardware, so its real-time latency and power draw on an embedded device are never measured.

6.2 Future Work

These limitations point to several concrete directions. The first is to strengthen the evaluation of the pupil extraction. Validating it on a larger, more densely annotated set - ideally produced semi-automatically to reduce labeling effort - would give tighter estimates of detection accuracy and surface rare failures. A second direction is to recover the full pupil ellipse natively from the event stream, instead of inheriting its shape from the anchor frame. Keeping the pupil's width, height, and orientation between frames would let geometry-aware models like the LSTM run on the high-frequency stream, and would provide the shape cues that matter most at the periphery, where accuracy is currently weakest. The catch is efficiency: since the whole point of the event stream is its high update rate, any such method is only worthwhile if it stays cheap enough to preserve that rate. A third direction is a better benchmark.

Collecting a continuous near-eye event dataset with a higher-accuracy gaze reference, and with labeling that accounts for saccadic reaction time, would remove the dataset artifacts identified above and allow the high-frequency event stream to be assessed fairly. Finally, the manual, per-subject tuning of the detection parameters could be removed, which currently limits how easily the pipeline scales to new users. Rather than hand-setting the binarization threshold, kernel size, and aspect-ratio bounds for each subject, these could be selected automatically - for example by a short self-calibration that searches the parameter ranges against a detection-quality criterion, or by adapting them online from the statistics of the incoming frames. Automating this step would let the system work out of the box on an unseen recording while keeping the detector lightweight and training-free. Beyond these, deploying and power-profiling the pipeline on real head-mounted hardware would close the gap between the simulation study presented here and the wearable setting that motivates it.

6.3 Concluding Remarks

In summary, this work shows that a training-free, event-based near-eye pipeline can locate the pupil about as accurately as a learned model while using a tiny fraction of the computation, and can still produce usable gaze estimates in real time on a CPU. The goal was never to set a new accuracy record. Instead, the pipeline makes a deliberate trade: it gives up some gaze accuracy in return for running without a GPU, and within the limited compute and power budget of a wearable device. As event cameras make their way into consumer near-eye hardware, lightweight pipelines like this one - together with the directions described above for narrowing the remaining accuracy gap - offer a practical route to gaze estimation that is fast, efficient, and able to run where heavier learned models still cannot.

Bibliography

[1] Takashi Abe. Perclos-based technologies for detecting drowsiness: current evidence and future directions. *Sleep Advances*, 4(1):zpad006, 2023.

[2] Amer Al-Rahayfeh and Miad Faezipour. Eye tracking and head movement detection: A state-of-art survey. *IEEE journal of translational engineering in health and medicine*, 1:2100212–2100212, 2013.

[3] Rachel Albert, Anjul Patney, David Luebke, and Joohwan Kim. Latency requirements for foveated rendering in virtual reality. *ACM Transactions on Applied Perception (TAP)*, 14(4):1–13, 2017.

[4] Anastasios N Angelopoulos, Julien NP Martel, Amit PS Kohli, Jorg Conradt, and Gordon Wetzstein. Event based, near eye gaze tracking beyond 10,000 hz. *arXiv preprint arXiv:2004.03577*, 2020.

[5] Elena Arabadzhiyska, Okan Tarhan Tursun, Karol Myszkowski, Hans-Peter Seidel, and Piotr Didyk. Saccade landing position prediction for gaze-contingent rendering. *ACM Transactions on Graphics (TOG)*, 36(4):1–12, 2017.

[6] Bruce Bridgeman and Lawrence Stark. *The theory of binocular vision: Ewald Hering (1868)*. Springer, 1977.

[7] Roger HS Carpenter. *Movements of the Eyes, 2nd Rev*. Pion Limited, 1988.

[8] Oliver Deane, Eszter Toth, and Sang-Hoon Yeo. Deep-saga: a deep-learning-based system for automatic gaze annotation from eye-tracking data. *Behavior Research Methods*, 55(3):1372–1391, 2023.

[9] Junyuan Ding, Ziteng Wang, Chang Gao, Min Liu, and Qinyu Chen. Facet: Fast and accurate event-based eye tracking using ellipse modeling for extended reality. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pages 10347–10354. IEEE, 2025.

[10] Stefan Dowiasch, Peter Wolf, and Frank Bremmer. Quantitative comparison of a mobile and a stationary video-based eye-tracker. *Behavior research methods*, 52(2):667–680, 2020.

[11] Andrew T Duchowski. A breadth-first survey of eye-tracking applications. *Behavior Research Methods, Instruments, & Computers*, 34(4):455–470, 2002.

[12] Wolfgang Fuhl, Marc Tonsen, Andreas Bulling, and Enkelejda Kasneci. Pupil detection for head-mounted eye tracking in the wild: an evaluation of the state of the art. *Machine Vision and Applications*, 27(8):1275–1288, 2016.

[13] Guillermo Gallego, Tobi Delbrück, Garrick Orchard, Chiara Bartolozzi, Brian Taba, Andrea Censi, Stefan Leutenegger, Andrew J Davison, Jörg Conradt, Kostas Daniilidis, et al. Event-based vision: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 44(1):154–180, 2020.

[14] Brian Guenter, Mark Finch, Steven Drucker, Desney Tan, and John Snyder. Foveated 3d graphics. *ACM transactions on Graphics (tOG)*, 31(6):1–10, 2012.

[15] Elias Daniel Guestrin and Moshe Eizenman. General theory of remote gaze estimation using the pupil center and corneal reflections. *IEEE Transactions on biomedical engineering*, 53(6):1124–1133, 2006.

[16] Dan Witzner Hansen and Qiang Ji. In the eye of the beholder: A survey of models for eyes and gaze. *IEEE transactions on pattern analysis and machine intelligence*, 32(3):478–500, 2009.

[17] Craig Hennessey, Borna Noureddin, and Peter Lawrence. A single camera eye-gaze tracking system with free head motion. In *Proceedings of the 2006 symposium on Eye tracking research & applications*, pages 87–94, 2006.

[18] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[19] Khadija Iddrisu, Waseem Shariff, Noel E O'Connor, Joseph Lemley, and Suzanne Little. Evaluating image-based face and eye tracking with event cameras. In *European Conference on Computer Vision*, pages 224–240. Springer, 2024.

[20] Chi Jian-Nan, Zhang Peng-yi, Zheng Si-yi, Zhang Chuang, and Huang Ying. Key techniques of eye gaze tracking based on pupil corneal reflection. In *2009 WRI Global Congress on Intelligent Systems*, volume 2, pages 133–138. IEEE, 2009.

[21] Yizhou Jiang, Wenwei Wang, Lei Yu, and Chu He. Eye tracking based on event camera and spiking neural network. *Electronics*, 13(14):2879, 2024.

[22] Anuradha Kar and Peter Corcoran. A review and analysis of eye-gaze estimation systems, algorithms and performance evaluation methods in consumer platforms. *IEEE Access*, 5:16495–16519, 2017.

[23] Joohwan Kim, Michael Stengel, Alexander Majercik, Shalini De Mello, David Dunn, Samuli Laine, Morgan McGuire, and David 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*, pages 1–12, 2019.

[24] Xavier Lagorce, Garrick Orchard, Francesco Galluppi, Bertram E Shi, and Ryad B Benosman. Hots: a hierarchy of event-based time-surfaces for pattern recognition. *IEEE transactions on pattern analysis and machine intelligence*, 39(7):1346–1359, 2016.

[25] Kang Il Lee, Jung Ho Jeon, and Byung Cheol Song. Deep learning-based pupil center detection for fast and accurate eye tracking system. In *European Conference on Computer Vision*, pages 36–52. Springer, 2020.

[26] Yuan-Chieh Lee. Active eye-tracking improves lasik results. *Journal of Refractive Surgery*, 23(6):581–585, 2007.

[27] Nealson Li, Muya Chang, and Arijit Raychowdhury. E-gaze: Gaze estimation with event camera. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(7):4796–4811, 2024.

[28] Yali Li, Shengjin Wang, and Xiaoqing Ding. Eye/eyes tracking based on a unified deformable template and particle filtering. *Pattern Recognition Letters*, 31(11):1377–1387, 2010.

[29] Kristian Lukander. Measuring gaze point on handheld mobile devices. In *CHI'04 Extended Abstracts on Human Factors in Computing Systems*, pages 1556–1556, 2004.

[30] Susana Martinez-Conde, Stephen L Macknik, Xoana G Troncoso, and David H Hubel. Microsaccades: a neurophysiological analysis. *Trends in neurosciences*, 32(9):463–475, 2009.

[31] Maria Laura Mele and Stefano Federici. Gaze and eye-tracking solutions for psychological research. *Cognitive processing*, 13(Suppl 1):261–265, 2012.

[32] Carlos H Morimoto and Marcio RM Mimica. Eye gaze tracking techniques for interactive applications. *Computer vision and image understanding*, 98(1):4–24, 2005.

[33] D Murthy LR, Abhishek Mukhopadhyay, Shambhavi Aggarwal, Ketan Anand, and Pradipta Biswas. Towards precision in appearance-based gaze estimation in the wild. *arXiv e-prints*, pages arXiv–2302, 2023.

[34] Irmingard M Neuhann, Barbara AM Lege, Markus Bauer, Joerg M Hassel, Anton Hilger, and Thomas F Neuhann. Static and dynamic rotational eye tracking during lasik treatment of myopic astigmatism with the zyoptix laser platform and advanced control eye tracker. *Journal of Refractive Surgery*, 26(1):17–27, 2010.

[35] Yan Ru Pei, Sasskia Brüers, Sébastien Crouzet, Douglas Mclelland, and Olivier Coenen. A lightweight spatiotemporal network for online eye tracking with event camera. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 5780–5788, June 2024.

[36] Thies Pfeiffer and Cem Memili. Model-based real-time visualization of realistic three-dimensional heat maps for mobile eye tracking and eye tracking in virtual reality. In *Proceedings of the ninth biennial acm symposium on eye tracking research & applications*, pages 95–102, 2016.

[37] Alexander Plopski, Teresa Hirzle, Nahal Norouzi, Long Qian, Gerd Bruder, and Tobias Langlotz. The eye in extended reality: A survey on gaze interaction and eye tracking in head-worn extended reality. *ACM Computing Surveys (CSUR)*, 55(3):1–39, 2022.

[38] Dale Purves, George J. Augustine, David Fitzpatrick, Lawrence C. Katz, Anthony-Samuel LaMantia, James O. McNamara, and S. Mark Williams, editors. *Neuroscience*. Sinauer Associates, Sunderland, MA, 2nd edition, 2001.

[39] Radwa Reda, Manal Tantawi, Howida shedeed, and Mohamed F Tolba. Analyzing electrooculography (eog) for eye movement detection. In *International Conference on Advanced Machine Learning Technologies and Applications*, pages 179–189. Springer, 2019.

[40] Argha Sen, Nuwan Sriyantha Bandara, Ila Gokarn, Thivya Kandappu, and Archan Misra. Eyetraes: fine-grained, low-latency eye tracking via adaptive event slicing. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 8(4):1–32, 2024.

[41] Niklas Stein, Diederick C Niehorster, Tamara Watson, Frank Steinicke, Katharina Rifai, Siegfried Wahl, and Markus Lappe. A comparison of eye tracking latencies among several commercial head-mounted displays. *i-Perception*, 12(1):2041669520983338, 2021.

[42] Satoshi Suzuki et al. Topological structural analysis of digitized binary images by border following. *Computer vision, graphics, and image processing*, 30(1):32–46, 1985.

[43] Tobii AB. Tobii pro glasses 3: Real-world behavior captured with precision. `https://www.tobii.com/products/eye-trackers/wearables/tobii-pro-glasses-3`.

[44] Niilo V Valtakari, Ignace TC Hooge, Charlotte Viktorsson, Pär Nyström, Terje Falck-Ytter, and Roy S Hessels. Eye tracking in human interaction: Possibilities and limitations. *Behavior Research Methods*, 53(4):1592–1608, 2021.

[45] Haoyang Wang, Ruishan Guo, Pengtao Ma, Ciyu Ruan, Xinyu Luo, Wenhua Ding, Tianyang Zhong, Jingao Xu, Yunhao Liu, and Xinlei Chen. Event camera meets mobile embodied perception: abstraction, algorithm, acceleration, application. *ACM Computing Surveys*, 58(8):1–41, 2026.

[46] Kang Wang and Qiang Ji. Real time eye gaze tracking with 3d deformable eye-face model. In *Proceedings of the IEEE international conference on computer vision*, pages 1003–1011, 2017.

[47] Kang Wang, Hui Su, and Qiang Ji. Neuro-inspired eye tracking with eye movement dynamics. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9831–9840, 2019.

[48] Zuowen Wang, Chang Gao, Zongwei Wu, Marcos V Conde, Radu Timofte, Shih-Chii Liu, Qinyu Chen, Zheng-jun Zha, Wei Zhai, Han Han, et al. Event-based eye tracking. ais 2024 challenge survey. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5810–5825, 2024.

[49] Walter W Wierwille, SS Wreggit, CL Kirn, LA Ellsworth, and RJ Fairbanks. Research on vehicle-based driver status/performance monitoring; development, validation, and refinement of algorithms for detection of driver drowsiness. final report. Technical report, 1994.

[50] Zhengyang Wu, Srivignesh Rajendran, Tarrence van As, Joelle Zimmermann, Vijay Badrinarayanan, and Andrew Rabinovich. Eyenet: A multi-task network for off-axis eye gaze estimation and user understanding. *arXiv preprint arXiv:1908.09060*, 2019.

[51] Zhengyang Wu, Srivignesh Rajendran, Tarrence van As, Joelle Zimmermann, Vijay Badrinarayanan, and Andrew Rabinovich. Magiceyes: A large scale eye gaze estimation dataset for mixed reality. *arXiv preprint arXiv:2003.08806*, 2020.

[52] Xucong Zhang, Seonwook Park, Thabo Beeler, Derek Bradley, Siyu Tang, and Otmar Hilliges. Eth-xgaze: A large scale dataset for gaze estimation under extreme head pose and gaze variation. In *European conference on computer vision*, pages 365–381. Springer, 2020.

[53] Xucong Zhang, Yusuke Sugano, Mario Fritz, and Andreas Bulling. Mpiigaze: Real-world dataset and deep appearance-based gaze estimation. *IEEE transactions on pattern analysis and machine intelligence*, 41(1):162–175, 2017.

[54] Guangrong Zhao, Yurun Yang, Jingwei Liu, Ning Chen, Yiran Shen, Hongkai Wen, and Guohao Lan. Ev-eye: Rethinking high-frequency eye tracking through the lenses of event cameras. *Advances in Neural Information Processing Systems*, 36:62169–62182, 2023.

[55] Shifan Zhu, Zhipeng Tang, Michael Yang, Erik Learned-Miller, and Donghyun Kim. Event camera-based visual odometry for dynamic motion tracking of a legged robot using adaptive time surface. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3475–3482. IEEE, 2023.

[56] Alex Zihao Zhu, Liangzhe Yuan, Kenneth Chaney, and Kostas Daniilidis. Unsupervised event-based optical flow using motion compensation. In *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, pages 0–0, 2018.