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

Efficient Eye Tracking Using Near-Eye Event Cameras: From Event-based Detection to Rapid Updates

Jiaheng Liu

TU Delft Delft University of Technology

Efficient Eye Tracking Using Near-Eye Event Cameras: From Event-based Detection to Rapid Updates

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

Jiaheng Liu
J.Liu-58@student.tudelft.nl
solarplover@outlook.com

August 26th, 2025

Author
Jiaheng Liu (J.Liu-58@student.tudelft.nl)
(solarplover@outlook.com)

Title
Efficient Eye Tracking Using Near-Eye Event Cameras: From Event-based Detection to Rapid Updates

MSc Presentation Date
August 26th, 2025

Graduation Committee

dr. Qing Wang (chairman)	Delft University of Technology
dr. Guohao Lan	Delft University of Technology
dr. Xucong Zhang	Delft University of Technology

Abstract

Eye tracking is a cornerstone technology for next-generation human-computer interaction, particularly in Extended Reality (XR), and other healthcare applications. However, traditional frame-based eye tracking systems are constrained by latency, power consumption, and motion blur. Event cameras offer a promising alternative with their high temporal resolution, high dynamic range and low data redundancy, but existing event-based methods often struggle to balance tracking accuracy, computational efficiency, and robustness, especially on resource-constrained mobile hardware.

This thesis addresses these challenges by proposing a novel, purely event-based eye tracking pipeline designed for high-frequency performance and robust accuracy within a strict computational budget. The pipeline accepts only event streams and estimates the pupil region in the field of view. The core contribution is a dual-state framework that synergistically combines a deep learning-based pupil detector with a lightweight, rapid template updater. For robust detection, a lightweight, attention-augmented segmentation network, named PupilUNet, is developed. It leverages a truncated MobileNetV3 Small encoder and a parameter-free attention mechanism to accurately segment the pupil boundary from Speed-Invariant Time Surface (SITS) representations, which provide a stable input by normalizing for motion speed. To overcome the scarcity of annotated data, a comprehensive framework is introduced to augment a large-scale training dataset from limited initial labels. Once a high-confidence pupil template is detected, the system transitions to a rapid updating mode, employing an optimized, vectorized point-to-edge matching algorithm to track the pupil at kilo-Hertz frequencies with millisecond latency. A dynamic control logic monitors tracking quality and seamlessly reverts to the robust detection mode when necessary, ensuring both speed and resilience.

Experimental results on the EV-Eye dataset validate the pipeline's effectiveness. The PupilUNet detector achieves a P5 accuracy of 96.3% (pupil center error < 5 pixels), while the rapid updater operates with an average latency of approximately 1 ms. The lightweight PupilUNet model contains merely 0.177 M parameters and inferences within 0.553 GFLOPs. The fully integrated system sustains a P5 accuracy of 85.2% while achieving a peak tracking frequency of over 960 Hz. This work demonstrates a practical and efficient solution that successfully navigates the trade-offs between accuracy and latency, establishing a new baseline for high-performance, event-based eye tracking on mobile and embedded systems.

Preface

I have always been drawn to human-centered technology, developing systems that work with us rather than against us. When Apple released the Vision Pro and XR started becoming reality, I became curious about the technologies making these interfaces feel natural and intuitive.

Eye tracking caught my attention because of a striking mismatch: our eyes move incredibly fast, yet most tracking systems can barely keep up. Then I discovered event cameras. These neuromorphic sensors work like biological retinas, responding to changes rather than capturing full frames. It felt like the missing piece.

The idea that we could create eye tracking systems inspired by how our visual system works was compelling. Instead of struggling against the natural dynamics of human vision, why not embrace them? This thesis explores that possibility, using bio-inspired sensors to build truly efficient, human-centered eye tracking for the next generation of interactive technologies.

I want to express my sincere gratitude to my graduation committee: dr. Qing Wang (chairman), dr. Guohao Lan, and dr. Xucong Zhang for their valuable guidance and expertise throughout this research. Special thanks go to my daily supervisor, dr. Guohao Lan, for his patience, thoughtful guidance, and considerate support during the challenging moments of this thesis. His insights and encouragement were instrumental in shaping this work. I extend my sincere appreciation to the creators of the EV-Eye dataset and the wider research community dedicated to event-based vision and eye tracking. This work is built upon the groundwork laid by numerous researchers who have generously shared their data and findings, and their contributions are invaluable. I am grateful to TU Delft and the Faculty of Electrical Engineering, Mathematics and Computer Science for providing the resources and academic environment that made this research possible. My heartfelt appreciation goes to Yijian Lu for his warm companionship and encouragement throughout this journey. His support made the long hours and difficult moments more bearable. In general, this thesis represents months of learning, debugging, and discovery. The journey has taught me as much about persistence and as it has about eye tracking.

Jiaheng Liu

Delft, The Netherlands
21st August 2025

Contents

Chapter 1

Introduction

1.1 Research Background and Motivation

1.1.1 Eye Movements Tracking & Applications

The human eye represents one of the most dynamically active organs in the body, performing three primary types of movements: saccades, fixations (accompanied by involuntary microsaccades [51]), and smooth pursuits. According to Kowler's definition [27], a saccade refers to a rapid shift of the eye's focal point from one selected location to another, while fixation denotes the eye maintaining focus on a stationary target. Smooth pursuit represents the third category, characterized by continuous tracking of moving targets along the line of sight. Eye tracking technology harnesses these natural movements to determine pupil position and subsequent gaze direction, transforming inherent eye behavior into insightful data. Algorithms and models are deployed to capture input from near-eye or full-face sensors reporting the dynamic parameters of eye movements. This capability has established eye tracking as a prominent research domain with far-reaching potential across multiple scientific and practical applications, as detailed below.

The significance of eye tracking technology spans diverse domains, demonstrating substantial impact in modern technology and healthcare applications. Within Extended Reality (XR) systems, including both augmented and virtual reality, eye tracking has emerged as a cornerstone technology. Following Apple's release of the Vision Pro headset in June 2023 [3], XR technologies have experienced unprecedented growth and are anticipated to fundamentally transform human perception and interaction with cyber-physical environments [55]. Eye tracking provides a revolutionary paradigm for human-computer interfaces [44, 25], enabling XR devices to accurately interpret users' emotional states [65] and intentions without requiring manual input. Through this technology, XR users can intuitively select virtual targets using their natural line of sight [29], access customized content that responds to their gaze patterns [28, 10], and authenticate their identity through reflexive eye movements [36, 26]. These compelling capabilities establish eye tracking as the foundational framework for next-generation XR developments.

Healthcare and medical applications represent another critical domain where eye tracking technology demonstrates its human-centered value. Precision laser

eye surgery procedures depend on accurate real-time eye position data provided by sophisticated eye tracking systems [46]. Moreover, abnormal eye movement patterns serve as valuable diagnostic indicators for neurological conditions, including Parkinson's disease [52] and autism spectrum disorders [9]. Eye movement analysis also provides crucial insights into psychological states, supporting advanced research methodologies in psychology [40]. These diverse medical and diagnostic applications underscore eye tracking's essential role in advancing healthcare innovation and patient care.

1.1.2 Eye Tracking Challenges

Despite the promising applications, eye tracking presents fundamental technical challenges that stem from the dramatic motion dynamics and the stringent constraint of spatiotemporal resolution in ocular monitoring. These challenges can be categorized into three primary perspectives: high-speed motion tracking demands, the need for computational efficiency, and environmental robustness requirements.

First, the high-speed nature of eye movements creates substantial tracking demands encompassing both motion capture and hardware performance requirements. During saccades, angular velocity can exceed 700°/s [22], generating intense requirements for rapid motion detection. Simultaneously, the confined ocular region necessitates critical spatial accuracy within the limited field of view, demanding precise spatiotemporal resolution to effectively capture both voluntary and involuntary eye movements for reliable gaze estimation and biometric analysis. To address these motion characteristics, eye tracking systems must achieve high sampling frequencies and low latency to ensure real-time motion capture, necessitating sensors with high sampling rates and systems capable of processing large data throughput without substantial delays.

Second, limited computational resources present a significant constraint, as eye-tracking applications typically operate on mobile computing devices such as headsets or smart glasses with limited computational resources and restricted energy budgets. Consequently, the efficiency of software algorithms and processing architectures becomes critical for practical implementation.

Third, environmental robustness represents a crucial challenge, as eye tracking systems must maintain reliable performance across diverse real-world scenarios, including varying lighting conditions, different user populations, and diverse stimulus tasks that may affect tracking effectiveness. To address these multifaceted challenges, effective eye tracking solutions must simultaneously achieve high-speed motion capture with spatial precision, optimize computational efficiency for mobile deployment, and ensure robust performance across varied environmental conditions.

1.1.3 Evolution from Traditional to Event-based Methods

In response to these challenges, eye tracking technology has evolved from invasive early methods using contact lenses and electrooculography (EOG) probes [41] toward non-intrusive camera-based systems that now dominate human eye tracking applications [41]. The integration of infrared (IR) light sources with IR-sensitive cameras [59] represents a key advancement, enabling the Pupil-Center

Corneal Reflection (PCCR) method, which is currently the most prevalent commercial technique [14]. PCCR systems utilize IR illumination to create bright pupil appearances and corneal reflections (glints), with algorithms determining gaze direction by calculating the vector between the pupil center and corneal reflection [45]. Modern systems also employ pupil tracking, which identifies and tracks the pupil center under IR illumination [1]. Since eyelashes and reflections can distort pupil shape, sophisticated algorithms using binary thresholding and ellipse fitting ensure accurate pupil boundary detection, which is critical for system precision.

However, frame-based systems face fundamental limitations. Current frame-based methods and commercial products operate at modest frequencies of 50-100 Hz [58]. High temporal resolution for capturing rapid saccades requires elevated frame rates [8], creating substantial power consumption, data throughput, and computational overheads, which are particularly problematic for mobile and XR applications. Additionally, fast saccades introduce motion blur, degrading tracking accuracy. These limitations highlight the need for more efficient sensing paradigms that address the trade-offs between temporal resolution, power efficiency, and tracking precision.

Distinguished from traditional frame-based cameras, bio-inspired event-based cameras, also known as Dynamic Vision Sensors (DVS), represent a paradigm shift for near-eye tracking applications. Unlike conventional cameras that capture entire frames at fixed rates, event cameras feature independent pixels operating asynchronously [47]. Each pixel monitors significant changes in log-scale brightness and triggers events containing pixel coordinates, precise timestamps (with microsecond resolution), and brightness polarity (increase or decrease). Event cameras offer several key advantages for eye tracking: microsecond-level latency, exceptional temporal resolution, high dynamic range ($> 120dB$), and significantly reduced power consumption and data throughput since static scene regions are ignored [53]. These characteristics make event cameras exceptionally well-suited for capturing highly dynamic eye movements with minimal motion blur and high fidelity, directly addressing the major limitations of their frame-based counterparts.

The unique, sparse nature of event data necessitates novel eye-tracking algorithms. Common strategies include converting asynchronous event streams into frame-like or other representations for processing by conventional computer vision techniques such as convolutional neural networks (CNNs) [17], or developing hybrid methods that fuse temporal event information with contextual data from periodic frames [67]. Foundational approaches [2] fit parametric eye models (including pupil and corneal glints) to incoming events, continuously updating model states. More advanced, purely event-based algorithms [32, 57, 61, 7] perform tracking exclusively on event streams by slicing and accumulating events into customized representations for tracking algorithms. This enables continuous eye feature estimation at kilohertz frequencies, far exceeding traditional system capabilities and fully leveraging event camera advantages. Additionally, researchers are exploring Spiking Neural Networks (SNNs) [21], which are neuromorphic algorithms naturally suited for processing asynchronous and sparse event data, showing promise for highly efficient and stable eye-tracking solutions.

1.1.4 Research Gap and Motivation

Despite the advances, current near-eye pure event-based eye tracking approaches face significant limitations in computational efficiency and robustness. For instance, E-Gaze [32] achieves high-frequency accurate gaze tracking through morphological operations on event frames but relies heavily on task-specific experimental parameters, limiting its generalization capability. Similarly, EyeTrAES [57] assumes circular pupil shapes in event frames, which may not hold in real-world scenarios, while their event representations lack critical temporal and polarity information, underutilizing the event camera's potential. State-of-the-art methods from the AIS 2024 Challenge [61] on the 3ET++ dataset [5, 61] employ end-to-end neural networks that are computationally intensive, constraining system latency and tracking frequency. Although FACET [7] achieves optimal pupil detection accuracy with low latency, this performance depends on high-end processors and TensorRT optimizations, remaining impractical for mobile computing devices. A significant gap therefore persists between current near-eye eye tracking capabilities and the requirements for efficient, robust, accurate, and fast solutions suitable for mobile and XR applications. This necessitates the development of lightweight, efficient event-based eye tracking algorithms that can fully leverage event camera advantages while being explainable and robust. Such solutions would bridge the theoretical benefits of event-based sensing with real-world deployment demands, enabling next-generation eye tracking systems that are simultaneously fast, accurate, energy-efficient, and robust.

1.2 Research Objectives and Contributions

This thesis project addresses the aforementioned challenges by designing a purely event-based, near-eye tracking pipeline that delivers ultra-high update rates and robust accuracy within strict mobile-compute and power budgets, ideally suited for next-generation, low-latency applications. The primary objective is to create a robust system capable of detecting and tracking the pupil at high frequency while maintaining computational latency in the millisecond range.

The research focuses on monocular eye tracking under the assumption that both eyes exhibit synchronized movement patterns, also known as conjugate eye movements [15]. The near-eye configuration ensures that the eye occupies the majority of the camera's field of view (FoV), creating an optimal sensing environment where the pupil, iris, eyelid, eyelashes, and eyebrow constitute the primary visual elements while minimizing external distractions.

The fundamental goal centers on accurate localization of the pupil region within the FoV, as this serves as the cornerstone for subsequent applications, including gaze estimation and eye movement-based authentication systems. Accurate pupil shape in elliptical models and pupil center coordinates are expected to fit the reference as precisely as possible. While gaze direction determination falls outside the scope of this work, owing to its dependency on specific experimental calibration procedures, the pupil tracking results provide the essential foundation for such applications. Previous research has demonstrated that gaze direction and point-of-gaze can be effectively derived from pupil tracking data

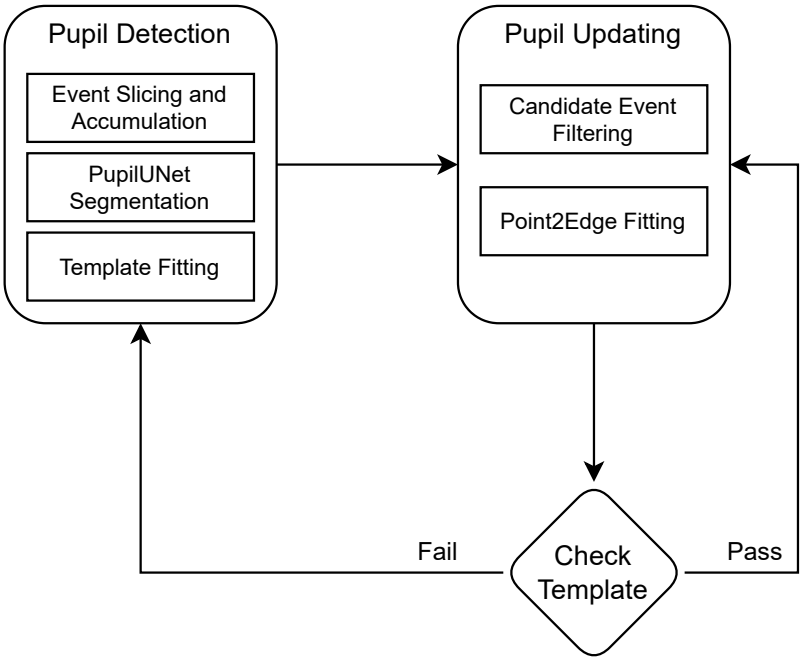


Figure 1.1: **The overview of the proposed pipeline from pupil detection to updating.**

through regression techniques [67] or LSTM networks [32].

Consequently, pupil detection and tracking are the most critical components of this research, requiring the development of novel algorithms that can effectively process event-based data streams while maintaining the spatiotemporal precision and computational efficiency demanded by real-world applications. The tracking of the pupil is updating the template after detection using the upcoming events. The pipeline overview, including event-based detection and rapid template updates, is visualized in Figure 1.1. The successful implementation of this system will advance the research in near-eye event-based eye tracking and establish a foundation for next-generation human-computer interaction technologies.

Scientific Contributions

The scientific contributions of this thesis are summarized as follows:

- **Efficient, Explainable Pupil Detection-Tracking Pipeline:** An efficient and explainable pupil detection-tracking framework is proposed, capable of kilo-Hertz tracking frequency with latency as low as the millisecond level. At its core, the lightweight **PupilUNet** segmentation network directly maps event representations to precise pupil regions. Events are sliced and accumulated into **dual-channel, causality-aware Speed-Invariant Time Surfaces (SITS)** that preserve both event polarity and temporal dynamics. An optimized lightweight template updater ensures

fast, robust tracking following deep learning-based detection. This integration achieves high accuracy under challenging conditions and remains feasible for real-time deployment on resource-constrained hardware.

- **Real-Data-Driven Augmentation and Training Framework:** A complete framework is developed to address the scarcity of annotated event-based datasets. Limited elliptical pupil annotations are used to train a frame-based segmentation model, generating pupil region masks from fixed-size event windows (5,000 events) in the EV-Eye dataset [67]. During training, an efficient caching mechanism reduces I/O overhead, accelerates training, and expands the adequate dataset size.
- **Lightweight, Attention-Augmented PupilUNet Architecture:** The PupilUNet network, derived from the classic U-Net architecture, is optimized through a **truncated MobileNetV3 encoder** that concentrates on shallow and mid-level features while discarding deeper, computationally expensive layers. A minimal-overhead attention mechanism is integrated into skip connections to enhance feature fusion and sharpen edge focus. Training with a compound loss function that combines regional accuracy, boundary precision, and confidence weighting ensures precise pupil edge localization with minimal computational cost, enabling deployment on mobile and embedded systems.

1.3 Outline

This thesis presents a comprehensive account of the research, encompassing a review of relevant literature, the design of the methodology, the implementation of algorithms, and extensive experiments to validate and evaluate both performance and efficiency. The work concludes after a comparative discussion. The structure is organized into five subsequent chapters:

- Chapter 2: reviews the relevant literature on eye tracking, highlighting the evolution from traditional frame-based methods to event-based systems. It critically analyzes existing datasets, event data representations, and state-of-the-art algorithms, identifying the research gap that this thesis aims to address.
- Chapter 3: details the design and methodology of the proposed dual-state eye tracking pipeline. This includes the system architecture, the data augmentation framework for generating training samples, the lightweight PupilUNet segmentation model, the rapid template updating mechanism, and the final integrated pipeline.
- Chapter 4: presents the comprehensive experiments and evaluations conducted to validate the proposed system. It covers the implementation details, the training and testing of the pupil detector, the optimization of the updater, and the performance assessment of the final integrated pipeline on real-world event streams.
- Chapter 5: provides a thorough discussion and analysis of the experimental results. It interprets the performance of the detector and updater, examines the accuracy-latency trade-off of the integrated system,

and compares its performance against other prominent event-based eye tracking methods.

- Chapter 6: concludes the thesis by summarizing the key contributions and findings. It also discusses the limitations of the current work and suggests potential directions for future research in the field of event-based eye tracking.

Chapter 2

Related Works

This chapter provides a comprehensive review of the existing literature on eye tracking, beginning with an exploration of fundamental eye movement paradigms and culminating in a detailed analysis of state-of-the-art event-based methodologies. The primary objective is to contextualize the challenges inherent in traditional eye-tracking systems and to establish the motivation for adopting event cameras for efficient near-eye pupil tracking. This review will systematically evaluate key datasets and event data representations, critically analyze current methods, and identify the research gaps that the proposed work aims to address.

2.1 Eye Movement Tracking Paradigms

Eye tracking is the technology of measuring the point of gaze or the motion of the pupil relative to the head. Accurate, rapid, and robust eye tracking is a vital tool in plenty of applications, ranging from human-computer interaction (HCI) and extended reality (XR) to driver monitoring systems and medical diagnostics [54]. Besides end-to-end solutions from sensor data to neural network output, the typical process of determining a person's gaze mainly involves two fundamental stages. First, key parts of the eye, such as the pupil, iris, and corneal reflections (glints), are detected and segmented. Second, the features extracted from these parts are used to build a model that can infer the pupil's position and, ultimately, the direction of gaze.

Eye tracking methodologies are broadly categorized into two main types:

- **Model-based methods** (e.g. [60, 33]) rely on creating an explicit geometric model of the eye. By identifying features like the pupil and iris contours or reflections from a known light source on the cornea, these methods can calculate the eye's orientation. While often accurate, they necessitate strict calibration procedures and controlled illumination, which can be cumbersome in real-world scenarios.

- **Appearance-based methods** (e.g. [6, 31]) take a more direct approach, learning a mapping from the visual appearance of the eye directly to the gaze coordinates. These methods, often employing traditional computer vision techniques or, more recently, deep neural networks (DNNs), can be

more robust to variations in eye shape and environment. However, they typically demand high-resolution images and significant computational resources to process the rich visual data.

While the ultimate goal of many eye-tracking systems is to estimate gaze, the detection and tracking of the pupil is the most fundamental and significant preceding step. The accuracy and robustness of pupil tracking directly dictate the performance ceiling for any subsequent gaze estimation. This thesis will therefore concentrate on the foundational challenge of near-eye pupil tracking. By focusing on this critical stage, a highly efficient and robust system can be developed, which can serve as a solid base for future gaze estimation frameworks. This focused approach allows for a deeper investigation into the core problems of high-speed detection and resilience to real-world challenges, without the added complexities and subject-specific calibrations inherent in gaze mapping.

2.2 Frame-based Eye Tracking Methods

The vast majority of conventional eye trackers [50, 42, 30, 24, 64] are frame-based, meaning they rely on grayscale or RGB cameras that capture a sequence of images at a fixed frame rate. To enhance the contrast between the pupil and the iris, these systems commonly utilize active infrared (IR) illumination. This technique makes the pupil appear as a bright, easily distinguishable circle or ellipse, simplifying the detection task for image processing algorithms. A prominent example of commercial frame-based systems is the Tobii Pro Glasses 3 [58]. These wearable eye trackers are equipped with multiple cameras and IR illuminators to capture a wide field of view and track eye movements with high precision. While such systems offer robust performance in many research and industrial settings, they are still bound by the limitations of frame-based technology.

Traditional frame-based eye-tracking methods face several key limitations despite their widespread use:

- **Limited Temporal Resolution**: Frame-based cameras are rate-limited by their frames per second (FPS), which is typically below 200 Hz for even high-speed models. This temporal ceiling makes it difficult to capture the dynamics of rapid eye movements like saccades, which can reach velocities of up to 700°/s [22].

- **Data Redundancy**: By capturing full frames at fixed intervals, these cameras generate a massive amount of redundant data, as much of the scene (and the eye) remains unchanged between consecutive frames. This leads to inefficiencies in data transmission, storage, and processing.

- **Motion Blur**: Fast eye movements can result in significant motion blur in standard camera frames, degrading the sharpness of the pupil boundary and complicating accurate detection.

- **Sensitivity to Lighting**: Although IR illumination helps, these systems can still be sensitive to ambient light conditions, which can interfere with pupil detection.

Figure 2.1: **The event camera DAVIS346. [19]**

These limitations highlight the need for a new sensing paradigm that can overcome the speed and data efficiency bottlenecks of frame-based systems.

2.3 Event Cameras in Eye Tracking

Event cameras, also known as neuromorphic or dynamic vision sensors (DVS), represent a paradigm shift in visual sensing, inspired by the workings of the human retina [13]. Unlike traditional cameras that capture entire frames, event cameras feature independent pixels that operate asynchronously. Each pixel monitors the light intensity falling upon it and generates an event only when a significant change in brightness (either positive or negative) is detected. An event encodes the pixel's coordinates (x, y), a precise timestamp t (with microsecond resolution), and the polarity of the brightness change ($p \in \{-1, 1\}$). Hence, an event stream involving N events can be defined as a set of N quadruples:

$$\mathcal{E} = \{e_i\}_{i=1}^{N} = \{x_i, y_i, t_i, p_i\}_{i=1}^{N}. \tag{2.1}$$

Figure 2.2 shows the visualization of the events in 3 dimensions with pixel coordinates and timestamps, which demonstrates the event camera's sensitivity to motion.

The event camera used in this thesis is DAVIS346 (Figure 2.1), a specific type of event camera that is particularly well-suited for this research. The DAVIS346 not only outputs a stream of asynchronous events but can also provide simultaneous grayscale frames, making it ideal for developing and validating hybrid datasets. It features a resolution of $346x260$ pixels, a high dynamic range of 120 dB, a short latency of approximately 20 μs, and the capability of generating up to 12M Events per second [19].

The unique operating principle of event cameras offers several compelling advantages for eye tracking:

- **High Temporal Resolution and Low Latency**: With microsecond-level temporal resolution, event cameras can capture the full dynamics of fast eye movements without motion blur.
- **Low Data Redundancy**: By only transmitting information about changes, event cameras drastically reduce the amount of redundant data, leading to lower bandwidth and power consumption.

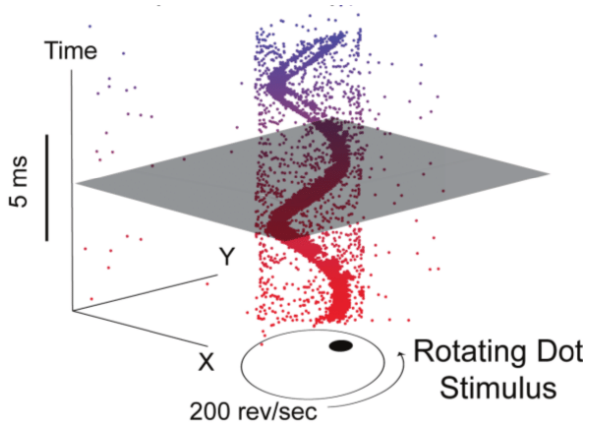


Figure 2.2: **An event camera in front of a rapidly rotating dotted disk generates an event stream. The moving dots trigger a clear event stream in the three-dimensional space, combining pixel coordinates and time axis. [12]**

- **High Dynamic Range (HDR)**: Event cameras can handle a much wider range of lighting conditions than traditional cameras, making them more robust to challenging illumination environments.
- **No Need for IR Lighting**: The high sensitivity to temporal contrast means that event cameras can often track the pupil boundary effectively under ambient light, potentially eliminating the need for power-consuming IR illuminators.

Some research explores processing event data directly through neuromorphic computing[4, 21]. However, this approach often requires novel, bio-inspired processors to fully harness its potential, which are not yet widely available for mobile applications. Therefore, this thesis focuses on developing efficient algorithms for conventional computing hardware.

2.4 Near-Eye Event-based Eye Tracking Datasets

The development of robust, learning-based eye tracking methods is critically dependent on the availability of high-quality datasets. In recent years, several datasets featuring near-eye event data have been introduced.

EVBEye's dataset [2]: This was one of the pioneering datasets to combine frame and event data for gaze tracking. By fitting an eye model using both modalities, it could map gaze directions. However, a significant limitation of EVBEye is the simplicity of its data collection scenario; it primarily consists of smooth pursuit eye movements, which does not provide enough diversity to train models for robust, general-purpose tracking.

EyeTrAES's dataset [57]: This dataset also provides event-based data for eye tracking. It claims its wearable and mobile data collection setup, but the field of view is rather small, so that it can assume a circular pupil shape.

3ET++ [61]: This is a pure event-based dataset created for a tracking challenge. A major drawback is that it only provides pupil center labels. This level of annotation with no pupil region and shape information is insufficient to evaluate pupil detection. For applications that require point-of-gaze (PoG) rendering or a more detailed understanding of the eye's state, the pupil center is not enough.

EV-Eye [67]: It is the most comprehensive large-scale, multimodal eye-tracking dataset available. It was collected from 48 participants and contains over 1.5 million near-eye binocular grayscale images and 2.7 billion event samples from two DAVIS346 cameras. The dataset captures a wide range of natural eye movements, including fixations, saccades, and smooth pursuits. Due to its large size, diversity of subjects and movements, and rich multi-modal data, the EV-Eye dataset was selected for this thesis. It provides an ideal foundation for developing and rigorously evaluating a novel pupil-tracking algorithm.

2.5 Event Data Representations for Pupil Tracking

The conversion of raw event streams into suitable representations for deep learning architectures represents a fundamental challenge in event-based pupil tracking. Raw event data, comprising sequences of (x, y, t, p) tuples, inherently lacks compatibility with conventional deep learning frameworks such as CNNs, which are designed to process dense, grid-structured inputs [63]. Consequently, transforming sparse, asynchronous event data into appropriate representations becomes a critical preprocessing step that directly impacts downstream performance.

Several distinct approaches have emerged to address this representational challenge, each offering unique advantages while introducing specific limitations. Examples of each event representation are illustrated in Figure 2.3.

- **Event Frames** represent the most straightforward approach to event data conversion [35]. This method slices events based on either fixed temporal windows or predetermined event counts, subsequently accumulating these slices into 2D histograms according to specific processing rules like contribution and decay [20]. In this representation, each pixel value encodes the cumulative effect of events occurring at that spatial coordinate. While this approach offers computational simplicity and ease of implementation, it fundamentally sacrifices the precise temporal information contained within each window, potentially limiting the representation's capacity to capture fine-grained temporal dynamics essential for accurate pupil tracking.

- **Voxel Grid** representations provide a more sophisticated approach by leveraging the natural three-dimensional structure of event data. Raw events, inherently existing as points in spatiotemporal coordinates (x, y, t), can be systematically organized into dense, regular volumetric structures through temporal partitioning [69]. This methodology achieves a balanced compromise between temporal resolution preservation and computational efficiency when compared to processing raw three-dimensional event data directly. However, the resulting multi-stack volumes increase computa-

tional latency during neural network processing, potentially limiting real-time applications.

- **Time Surface (TS)** representations offer an alternative paradigm that emphasizes temporal context preservation through spatially-distributed timestamp encoding. A Time Surface constructs a two-dimensional map storing the timestamp of the most recent event at each pixel location, enhanced by an exponential decay kernel for temporal weighting. Following Macanović et al.'s formulation [38], each pixel value is computed according to (2.2), where τ represents a scene-dependent tunable parameter, t denotes the reference timestamp, and $T(x, y)$ captures the temporal context of the most recent proximate event preceding time t. The exponential decay function ensures that recent events maintain high pixel intensities while historical events gradually fade [48]. While this approach effectively encodes recent motion history, its representation exhibits velocity-dependent characteristics that pose significant challenges for feature extraction. Specifically, fast-moving pupils generate markedly different temporal patterns compared to slow-moving ones, creating substantial difficulties for segmentation models attempting to learn velocity-invariant features from Time Surface representations.

$$Value(x, y; t) = e^{-\frac{|t-T(x,y)|}{\tau}}. \tag{2.2}$$

To address the velocity dependency inherent in standard Time Surface representations, Manderscheid et al. introduced the Speed-Invariant Time Surface (SITS) [39]. The standard Time Surface suffers from a critical limitation: its exponential decay $\exp(-\lambda \cdot \Delta t)$ depends entirely on the temporal interval Δt between events, causing identical objects moving at different velocities to produce drastically different representations. During rapid saccadic eye movements, pupil edges generate high-intensity, sharp features due to small Δt values, while smooth pursuit movements create dim, blurred representations with larger temporal intervals. This velocity-dependent variability forces neural networks to allocate learning capacity to accommodate speed-induced appearance changes rather than focusing on intrinsic geometric properties.

SITS addresses this challenge by shifting temporal decay dependency from physical time to event time. When an event (x, y, t, p) occurs, instead of computing the temporal difference $\Delta t = t - t_{\text{last}}$ from the previous event around location (x, y), SITS calculates the number of events ΔN_{events} that occurred within the local neighborhood during this interval. The SITS value is then computed as:

$$\text{SITS}(x, y, t) = \exp(-\lambda \cdot \Delta N_{\text{events}}) \tag{2.3}$$

This formulation effectively normalizes velocity influences: during rapid movements, although Δt is small, the burst of global events results in large ΔN_{events} values, while slow movements produce small ΔN_{events} despite longer temporal intervals. Consequently, SITS generates consistent pupil edge representations across varying movement speeds, encoding intrinsic geometric shapes rather than transient motion states. This property proves particularly advantageous for segmentation tasks, as it enables networks to learn stable, velocity-independent geometric contours while improving model robustness and generalization capability.

2.6 State-of-the-Art in Event-based Eye Tracking

Event-based eye tracking has emerged as a promising alternative to traditional frame-based approaches, offering the potential for ultra-low latency, high temporal resolution, robust performance, and reduced data redundancy. However, existing methods face a fundamental trade-off between computational efficiency and tracking robustness, with different approaches prioritizing distinct aspects of this challenge.

Early work in this domain, conducted in EV-Eye [67], introduced a hybrid event-frame methodology that combines event data with traditional frames through a template updating mechanism. After applying frame-based segmentation and obtaining pupil templates, the algorithm maps candidate event points to their corresponding local edge segments in real time, updating contour continuity and suppressing noise from irrelevant events. This targeted update reduces computational load, preserves temporal resolution at the microsecond scale, and supports accurate gaze estimation in dynamic conditions. While this approach demonstrates improved stability and accuracy compared to previous work [2], the dependency on frames inherently sacrifices the ultra-low latency advantages that make event cameras particularly attractive for real-time eye tracking applications.

Moving toward pure event-based solutions, E-Gaze [32] demonstrated the feasibility of high-frequency gaze tracking using morphological operations applied directly to event frames. Although this method shows promise in controlled laboratory settings and performs well on the specific dataset from [2], its practical applicability is limited by several factors. The approach relies heavily on task-specific experimental parameters that hinder generalization to diverse real-world conditions. More critically, eye part segmentation struggles with common challenges such as eyelid-pupil or pupil-glint adhesion, which can cause significant tracking degradation.

Similarly, EyeTrAES [57] simplifies the tracking problem by assuming circular pupil geometry in event representations. While this assumption reduces computational complexity, it fails to account for the reality that pupils often appear elliptical, particularly under oblique viewing angles. Furthermore, event frame representation underutilizes the rich temporal dynamics and polarity information that constitute the primary advantages of event cameras over conventional sensors.

Since these morphological approaches [32, 57] have not been tested on the EV-Eye dataset, trials are done on this selected dataset following a similar morphological method. The pipeline of morphological processing, filtering and ellipse fitting is visualized in Figure 2.4. In Figure 2.4(a), a fixed number of events are sliced and accumulated into an event frame, which is then applied to a clustering algorithm named DBSCAN [11] in Figure 2.4(b). Based on thresholds including contour area, circularity, hull area, hull circularity, solidity, and density, clusters are filtered in Figure 2.4(c), leaving pupil arcs alone like Figure 2.4(d). After fitting ellipses to the remaining arcs and going through posterior filtering to discard implausible ellipses, the result should be the pupil region in Figure 2.4(e). However, this is only an ideal scene. Adhesion of eye parts is common as displayed in Figure 2.5, when the morphological approaches

can hardly handle the cases. The robustness and adaptation of such methods are unsatisfactory according to the trials. Furthermore, the need for tailored parameters and thresholds heavily confines their practicability.

Apart from finding pupils via morphological operations and classic image processing techniques, the recent 3ET++ Challenge [61] showcased a different paradigm, with leading solutions employing complex end-to-end deep neural networks that directly regress pupil centers from raw event streams. Despite achieving impressive performance metrics, these regression-based approaches suffer from inherent architectural limitations. These approaches lack explicit geometric shape modeling, making the models behave like a black box. When severe occlusions occur, for instance, the centroid of the event distribution shifts dramatically, leading to significant prediction errors. This vulnerability stems from the regression paradigm's tendency to map events directly to coordinate predictions without incorporating geometric constraints that could provide stability under challenging conditions. Additionally, the computational intensity of these deep learning models introduces significant latency overhead, reducing tracking rates and limiting their suitability for mobile deployment scenarios where real-time performance is essential.

Combining geometric pupil shape modeling, FACET [7] attempts to regress ellipse parameters from fast causal event volumes, providing more detailed elliptical parameter estimation compared to 3ET++ methods. It addresses latency issues through hardware acceleration using technologies like NVIDIA TensorRT on an NVIDIA RTX3090. While achieving both high detection accuracy and reduced latency, FACET's end-to-end neural network architecture presents its own challenges, as it shares the same regression-based limitations regarding explainability and robustness. The complexity of the Feature Pyramid Network (FPN) with multiple detection heads architecture during training and deployment further complicates its computation and practical implementation.

These limitations across existing methods reveal a fundamental paradigmatic divide in event-based pupil tracking between regression and segmentation approaches. Regression-based methods, as demonstrated in the 3ET++ Challenge submissions and FACET, attempt to map events directly to coordinate predictions but may struggle under occlusion and noise conditions due to their lack of explicit geometric constraints. This absence of shape modeling creates a critical weakness when dealing with partial occlusions or noisy event data.

In contrast, segmentation-based approaches offer a fundamentally different strategy by modeling pupil shape explicitly, thereby naturally preserving geometric continuity even under adverse conditions. This paradigmatic difference becomes particularly evident when examining pupil occlusion resilience. The Swift-Eye framework [66] exemplifies the advantages of segmentation methodologies, achieving Intersection over Union (IoU) higher than 0.7 and F1-score higher than 0.81 even in large occlusion scenes with a dedicated strategy to tackle occlusions. These performance gains occur precisely in scenarios where regression methods fail due to centroid shift issues, highlighting the importance of explicit geometric modeling. However, the complex neural network architecture limits computational latency.

The evidence suggests that segmentation-based approaches provide a more robust foundation with slightly more latency for event-based eye tracking by explicitly modeling pupil geometry rather than relying on direct coordinate regression. This geometric modeling capability, combined with efficient template

fitting mechanisms, offers the potential to achieve both the robustness required for real-world deployment and the computational efficiency necessary for real-time applications. The integration of segmentation models with proven template updating strategies, such as the points-to-edge mechanisms demonstrated in EV-Eye [67], presents a promising direction for addressing the current limitations in event-based eye tracking systems.

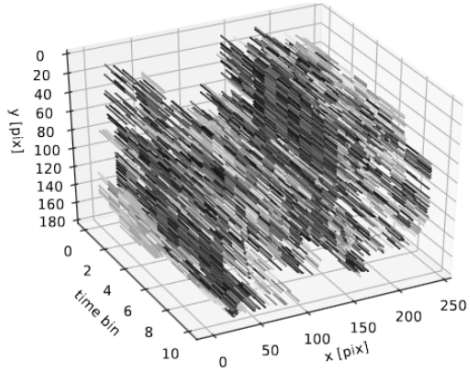
2.7 Summary

The review of the existing literature reveals a clear trajectory in eye tracking research, moving from traditional, frame-based systems with inherent limitations in speed and data efficiency towards the promising paradigm of event-based sensing. While event cameras offer significant advantages, current methods still face challenges in computational efficiency, robustness, and practicality for mobile applications. A critical analysis of datasets, event representations, and state-of-the-art algorithms highlights a specific need for a method that can:

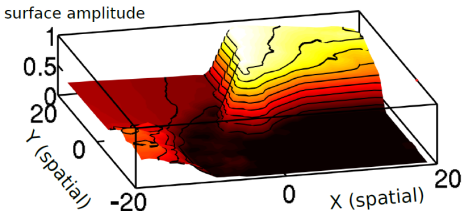
- Robustly segment the pupil region, rather than simply regressing to a center point, to improve robustness and provide more detailed region information.
- Utilize an event representation that is invariant to motion speed to create stable features for the learning algorithm.
- Achieve high performance on a lightweight, efficient neural network architecture suitable for edge devices.

This thesis proposes to address these challenges. By feeding causal Speed-Invariant Time Surface (SITS) representations into a segmentation model followed by ellipse fitting, the proposed work aims to achieve highly efficient and accurate pupil detection. This approach is designed to be robust and explainable, setting a new baseline for near-eye tracking on resource-constrained platforms. The focus on detection and segmentation lays the groundwork for a subsequent rapid template updating mechanism, pointing towards a complete, high-frequency eye-tracking system.

(a) Event Frame

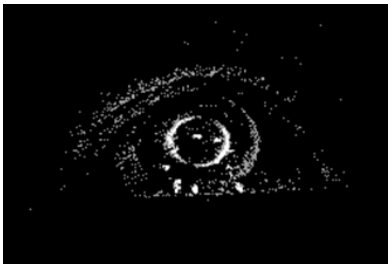


(b) Voxel Grid



(c) Time Surface

Figure 2.3: **(a) shows a simple event frame accumulated from 5000 consecutive events in the EV-Eye dataset [67]. Red pixels represent positive events while green pixels represent negative events. (b) shows the event voxel grid from [13], where the raw event stream is discretized into temporal bins. (c) shows a time surface from [38] as a 3D contour map with highlighted motion contours. Higher amplitude values correspond to more recent events.**



(a) Event Frame

(b) Clustering

(c) Contour Filtering

(d) Pupil Arcs

(e) Ellipse Fitting and Posterior Filtering

Figure 2.4: **Trial pipeline on EV-Eye dataset using morphological operations to detect pupil.**

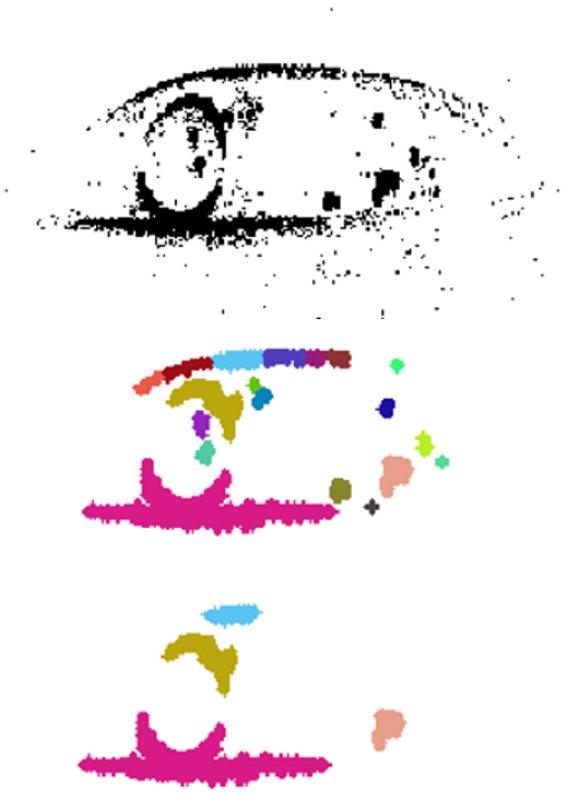


Figure 2.5: **An example of adhered eye parts. The pupil arc in red is mixed with the eyelid. The method cannot split the adhesion.**

Chapter 3

Method

3.1 Pipeline Overview and System Architecture

This chapter outlines the architecture of the proposed eye tracking system, which processes pure near-eye monocular event recordings and outputs ellipse parameters that characterize the pupil region within the field of view. The system operates on an updating-after-detection paradigm, where robust pupil detection ensures accuracy through segmentation and ellipse fitting, while template updating mechanisms optimize both tracking frequency and latency.

The eye tracking pipeline operates in two primary states: pupil detection and pupil updating, with control logic governing transitions between these modes (Figure 1.1). The system begins by partitioning the incoming event stream into slices of 5000 events, which are then accumulated to create dual-channel SITS frames as detailed in Section 3.2. These frames serve as input to the PupilUNet segmentation model, which extracts event coordinates corresponding to the pupil boundary. Subsequently, an elliptical template is fit to these coordinates to define the pupil region. The architectural details of PupilUNet and the ellipse fitting approach are presented in Section 3.3.

For PupilUNet model training, the EV-Eye dataset [67] is leveraged with augmented annotations through the augmentation strategy (Section 3.2). Using the provided frame-annotation pairs, a frame-based model is trained to generate pupil masks on reference frames where manual annotations are unavailable. By counting causal event slices backward from extended reference frame timestamps, corresponding SITS-mask samples are created for PupilUNet, establishing comprehensive train, validation, and test datasets for the SITS segmentation model.

Once successful pupil detection occurs, the system shifts to updating mode (Section 3.4). Here, the event slice size is reduced to 500 events, and candidate points are filtered in based on their proximity to the previous template. The optimized points-to-edge matching and template center updating algorithm then refines the template location. The system includes safeguards against abnormal fitting: when template scores fall below the threshold (indicating potential pupil occlusion or template distortion), the pipeline automatically returns to detection mode to establish a fresh template.

The complete system architecture, spanning from data preparation through

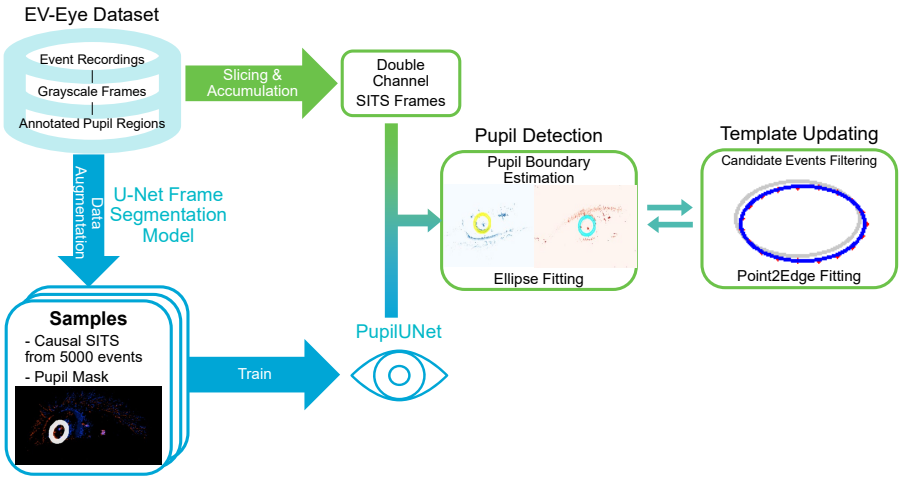


Figure 3.1: **Complete system architecture showing PupilUNet training pipeline (blue) and eye tracking pipeline (green). PupilUNet training pipeline: A U-Net frame-based model trained on 9011 frames with EV-Eye's original annotations generates extended annotations for additional frames, creating white pupil masks in *Samples*. Training samples consist of SITS-pupil mask pairs, where SITS inputs are generated by counting back 5000 events from reference timestamps and forming dual-channel frames (red: positive event channel, green: negative event channel). These samples train the PupilUNet segmentation model. Eye tracking pipeline: Event streams are sliced and accumulated into dual-channel SITS frames. PupilUNet predicts pupil boundary maps, yielding point coordinates that undergo ellipse fitting for final detection results. The system alternates between detection and updating modes based on template score examination. During *Template Updating*, candidate points (red dots) refine the template, with gray and blue ellipses representing previous and current templates, respectively.**

final tracking results, is illustrated in Figure 3.1. Despite the increased computational complexity of the segmentation-fitting approach compared to end-to-end regression methods, the enhanced explainability proves invaluable for system debugging and optimization. When tracking errors occur, each pipeline stage can be systematically traced from event frame quality to segmentation mask accuracy to ellipse fitting success, enabling targeted improvements. The robustness is also guaranteed because the segmentation model learns the shape and context of pupil regions, which enables the model to predict pupil regions even under challenging circumstances like partial pupil occlusion. The ellipse fitting further enhances robustness by minimizing possible segmentation outliers' distractions. Due to the fast updating mechanism, the average latency of eye tracking drops even if the segmentation-fitting detector takes a longer time. Section 3.5 addresses the integration of both tracking states and their control logic to form the complete tracking pipeline. Detailed implementation specifics

for each component and validations are provided in Chapter 4

3.2 Event Representation and Data Augmentation

A primary obstacle in developing robust segmentation models for event-based vision is the significant cost and effort associated with creating large and comprehensive annotated datasets that consist of frames and corresponding dense and precise labels. To overcome this limitation, frame-like event representation and an augmented dataset size are required. The data augmentation approach is designed to generate large-scale, high-quality, comprehensive training and validation datasets from the sparsely annotated EV-Eye dataset [67]. This process operates in two main stages: first, generating extensive new pupil mask annotations using a frame-based segmentation model, and second, converting corresponding event data into a specialized frame-like representation suitable for training the PupilUNet.

3.2.1 Frame-based Segmentation for Data Augmentation

The core of the data augmentation strategy is to leverage the limited provided ground truths of pupil annotation to generate a much larger set of annotations at timestamps when grayscale frames exist without annotations. The EV-Eye dataset provides 9,011 grayscale frames paired with elliptical pupil annotations. While valuable, this number is insufficient for training a stable and accurate segmentation network. The data augmentation approach uses these initial pairs to bootstrap the annotation process for other unannotated frames in the EV-Eye dataset.

In the field of image segmentation, one of the most influential deep-learning-based solutions is U-Net [56], which is a convolutional neural network architecture developed initially for biomedical image segmentation. The architecture features a distinctive U-shaped design with a contracting path (encoder) that captures context through successive convolutions and pooling operations, and an expansive path (decoder) that enables precise localization through upsampling and concatenation with high-resolution features from the contracting path. The key innovation of U-Net lies in its skip connections, which directly connect corresponding layers in the encoder and decoder paths, allowing the network to combine low-level spatial information with high-level semantic features for accurate pixel-wise predictions [37]. Since its introduction, U-Net has been widely adopted and adapted for various image segmentation tasks beyond biomedical applications, including satellite imagery analysis, autonomous driving, and general computer vision tasks [18, 68]. The architecture's effectiveness stems from its ability to work with relatively small datasets while achieving high segmentation accuracy, making it particularly valuable in domains where labeled data is scarce [56].

In this task, a classic frame-based U-Net model, featuring a standard encoder-decoder architecture with five levels of down-sampling and corresponding skip connections, is trained on 7208 frame-annotation pairs and validated on 1803 pairs, both from the 9011 ground truth annotation pairs. The architecture of this frame-based U-Net model is illustrated in Figure 3.2. This model learn

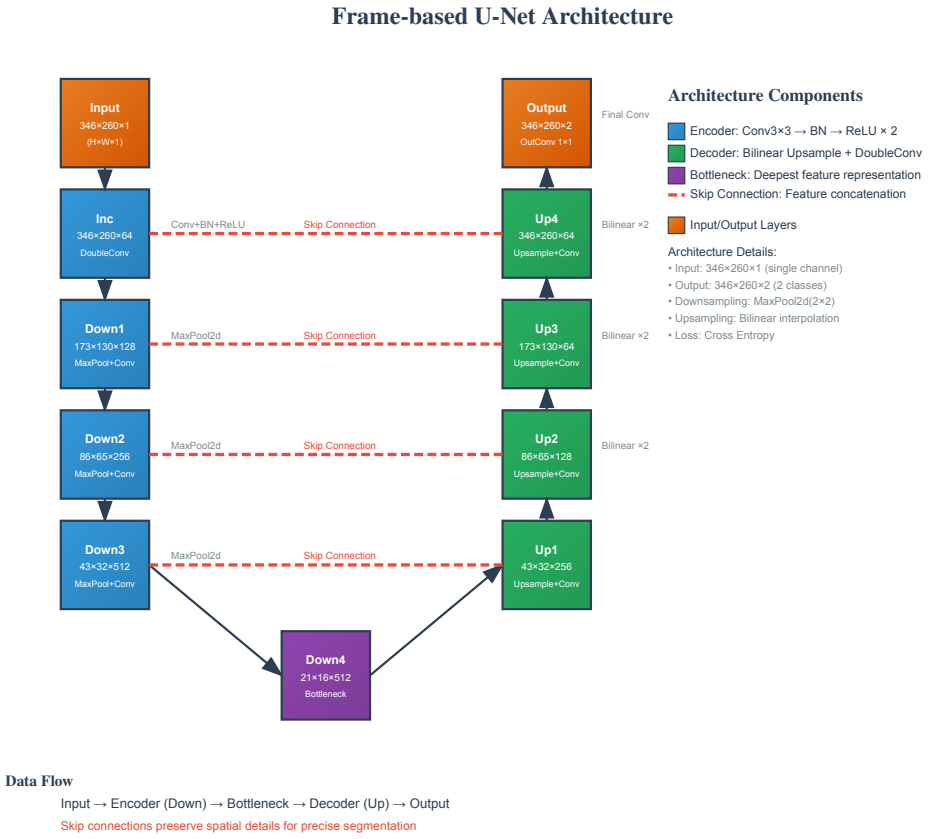


Figure 3.2: **A classic U-Net segmentation model to annotate pupil region. The architecture refers to FACET**[7]

.

to segment the pupil from grayscale images accurately. Once trained, this network is deployed as an auto-annotator on other frames within the dataset for which no ground-truth labels existed. This process can effectively expand the pool of annotations from a few thousand to over 60,000, creating a comprehensive set of different pupil masks, each with a precise reference timestamp for aligning frames with event slices. This automated annotation pipeline serves as the foundation for generating the training, validation and testing samples for event-based PupilUNet, as depicted in the blue part of Figure 3.1. The annotation masks are donut-shaped zones with 5-pixel thickness representing pupil boundaries, since events are mainly triggered by the movement of the boundary.

3.2.2 Speed-Invariant Time Surface (SITS) Accumulation

With a large collection of timestamped pupil masks, the next step is to create a corresponding event-based input representation for sample pairs. Rather than using raw event data, which is sparse and asynchronous, the PupilUNet segmentation task needs to accumulate events into a dense, frame-like tensor beforehand using the Speed-Invariant Time Surface (SITS) method introduced in Section 2.5. This representation is specifically designed to preserve critical temporal dynamics and event polarity, which are essential for understanding motion. The speed-invariant characteristic stabilizes the spatiotemporal features of the moving pupil at various speeds, which is crucial for the model to learn meaningfully.

For each generated pupil mask and its associated timestamp t_{label}, a causal window of the most recent N events (where N is empirically set to 5000) is extracted from the event stream. These events are then projected onto two separate 2D grids following SITS's algorithm, one for positive (ON) events and one for negative (OFF) events, matching the sensor's resolution of 260×346 pixels. The value at each pixel (x, y) in each channel is based on the recency of the last event at that location, calculated using an exponential decay function with regarding to the ΔN in (2.3), where λ is a decay constant empirically set to 0.001 for clear pupils, and ΔN is the relative index difference between the current event and the last recorded event at that exact pixel. This formulation makes the representation robust to variations in motion speed, as it depends on the order of events rather than their absolute timestamps.

The final result is a dual-channel SITS frame where one channel represents the speed-invariant time surface of ON events and the other represents the OFF events. This representation provides a rich, motion-aware input to the PupilUNet that encodes both spatial and temporal information from the event stream.

3.3 Segmentation Model and Pupil Detector

The core of the detection pipeline is a deep learning model designed for efficient, accurate pupil segmentation, followed by a robust ellipse fitting procedure. This section details the architecture of the dedicated PupilUNet, the specialized loss function used for its training, and the methodology for extracting precise ellipse parameters from the model's output.

3.3.1 PupilUNet Architecture

Built on U-Net, the PupilUNet is a lightweight segmentation network deeply customized for performance and efficiency in near-eye pupil tracking. The architecture is illustrated in Figure 3.3. A key innovation of PupilUNet is its truncated encoder, which utilizes MobileNet V3-Small [16] as its backbone. MobileNetV3-Small is a lightweight convolutional neural network architecture optimized for mobile and edge devices through neural architecture search, featuring squeeze-and-excitation blocks and hard-swish activations to achieve efficient inference with minimal computational overhead. Considering that pupil segmentation relies on identifying low-to-mid-level geometric features, such as contours and edges, rather than abstract, high-level semantic information like this is a human

eye, the architecture should be optimized to preserve these crucial spatial details while minimizing computational complexity. Thus, unlike conventional U-Net designs that use the full depth of a backbone network (e.g., Section 3.2.1), the encoder here is intentionally truncated, using only the shallow and middle-layer modules up to a 1/8 downsampling resolution. By discarding the deeper, computationally expensive layers, the model not only achieves a significant reduction in parameters and latency but, more importantly, it prevents the fine spatial information critical for precise contour detection from being diluted through excessive downsampling.

The decoder is engineered to complement this efficient encoder by integrating a parameter-free attention mechanism. Within each decoder block, skip connections from the encoder are enhanced using a Simple, Parameter-Free Attention Module (SimAM) [62]. Before fusing the shallow features from the encoder with the upsampled features from the decoder, it computes attention weights through spatial statistics, calculating the spatial mean (μ) and variance (v) across height and width dimensions. It then applies an energy function

$$e = \frac{d_{sq}}{4(v + \lambda)} + 0.5, \tag{3.1}$$

where d_{sq} represents squared deviations from the mean and λ is a hyperparameter. A sigmoid activation then generates attention weights that are element-wise multiplied with the input features, highlighting salient regions corresponding to pupil edges while suppressing irrelevant background noise. The PupilUNet decoder employs two decoder block instances with SimAM integration. The first one processes bottleneck output (40 channels, 1/8 resolution) with S2 skip features (24 channels, 1/4 resolution). The other one fuses D1's output (32 channels, 1/4 resolution) with S1 skip features (16 channels, 1/2 resolution). Each decoder block applies bilinear upsampling, handles spatial dimension mismatches through interpolation, applies SimAM attention to skip connections, concatenates features, and processes the fused representation through dual convolutional layers with batch normalization and dropout regularization. This is achieved without adding any trainable parameters and with negligible computational overhead, offering an extremely cost-effective method to improve feature fusion and sharpen boundary delineation. The final stage of the network consists of a lightweight head that produces a single-channel logit map representing the pupil segmentation.

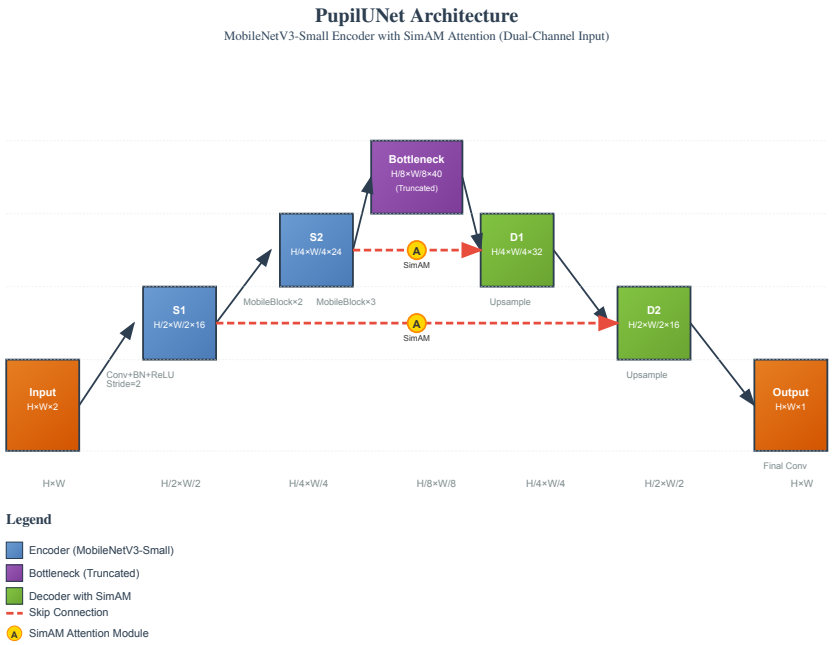


Figure 3.3: **PupilUNet Architecture Overview. The proposed PupilUNet employs a truncated MobileNetV3-Small encoder that extracts multi-scale features at three hierarchical levels: S1 ($H/2 \times W/2 \times 16$ channels), S2 ($H/4 \times W/4 \times 24$ channels), and a bottleneck layer ($H/8 \times W/8 \times 40$ channels). The encoder is intentionally truncated at 1/8 resolution to preserve low-to-mid-level geometric features crucial for contour segmentation. The decoder consists of two upsampling blocks (D1 and D2) that progressively restore spatial resolution through bilinear interpolation. Skip connections with parameter-free SimAM attention modules enhance feature fusion by selectively emphasizing relevant geometric information from encoder layers. The model accepts dual-channel input images and generates single-channel segmentation masks through a lightweight output head. This streamlined architecture balances computational efficiency with segmentation accuracy for pupil detection tasks.**

3.3.2 Confidence-Weighted Compound Loss Function

To train PupilUNet effectively, a sophisticated loss function was developed to improve the accuracy and robustness of pupil segmentation. A simple loss function is often insufficient, as it may struggle with the severe class imbalance between the small pupil region and the large background, or fail to enforce sharp boundary prediction. The single criterion also causes overfitting during training and validation. Therefore, a compound loss is formulated as a confidence-weighted sum of three distinct loss functions, each targeting a specific aspect of the segmentation task.

- **Dice Loss** is a segmentation loss function based on the Dice similarity coefficient that directly optimizes the overlap between predicted and ground truth regions [43], making it particularly effective for handling class imbalance and ensuring accurate regional overlap.
- **Focal Loss** [34] is a refinement of standard cross-entropy. It focuses the training process on hard-to-classify pixels, particularly those along the pupil boundary. It down-weights the loss for easily classified background pixels, forcing the model to concentrate on achieving higher precision at the edges.
- **Boundary Loss** [23] explicitly penalizes inaccuracies at the pupil boundary. It uses a distance map to apply a higher penalty to misclassified pixels that are further from the true boundary, directly encouraging the model to generate sharp and precise edges.

Furthermore, to enhance robustness against noisy or low-quality training samples generated by the data augmentation pipeline, a confidence weighting scheme is integrated into the final loss calculation. A sample confidence score is calculated for each SITS-mask pair based on the density of events within the pupil boundary region. This score allows the training process to weight high-quality samples more heavily, improving the model's robustness against ambiguous or noisy data, such as those generated during blinks or occlusions. The final loss $\mathcal{L}$ for each sample is scaled by this confidence score C, which is stated in (3.2):

$$\mathcal{L} = C \times (W_{Dice} \times \mathcal{L}_{Dice} + W_{Focal} \times \mathcal{L}_{Focal} + W_{Boundary} \times \mathcal{L}_{Boundary}), \quad (3.2)$$

where W_{Dice}, W_{Focal} and $W_{Boundary}$ are weights that balance the effect of each loss. This strategy allows the model to learn more from high-quality, reliable samples while reducing the influence of ambiguous data, leading to a more stable and accurate final model.

3.3.3 Pupil Detection from Segmentation Result

The output of the PupilUNet is a probability map, which must be translated into precise elliptical parameters for the following applications. This is accomplished through a multi-stage process designed to maximize accuracy by leveraging the raw event data.

First, the model's output logits are passed through a sigmoid function to generate a probability map, which is then binarized using an empirical threshold to create an initial pupil mask. However, instead of fitting an ellipse directly

to this mask, which may contain quantization errors, the system employs an event-based fitting strategy. The binary mask serves as a spatial filter to select event coordinates from the original input slice that fall within the predicted pupil boundary region. An ellipse is then fitted directly to this subset of high-precision event coordinates. This approach leverages the superior spatiotemporal resolution of the raw event data, using the segmentation mask as a robust guide to achieve a more accurate, sub-pixel fit.

Finally, a posterior filtering step is applied to ensure the plausibility of the detection. The parameters of the fitted ellipse are validated against physiological constraints, including realistic ranges for axis lengths and aspect ratios. Ellipses that are deemed implausible (e.g., excessively large, small, or elongated) are discarded. This final check enhances the robustness of the detector by rejecting erroneous fits that can occur under challenging conditions, ensuring that only reliable pupil detections are passed to subsequent stages of the pipeline.

3.4 Pupil Template Updating

To achieve the ultra-high tracking frequencies required for real-time applications, the system cannot rely solely on the computationally intensive deep learning detector for every estimation. Therefore, once an initial, high-confidence pupil template is established, the pipeline transitions to a lightweight and rapid updating mode, referring to EV-Eye's approach [67]. This mode is designed to incrementally refine the pupil's position using small, subsequent slices of events, achieving low latency and high speed. The process consists of candidate selection, iterative center refinement, and a quality assessment.

First, candidate events are selected from the incoming small event slice (e.g., 500 events) by filtering them through a donut-shaped Region of Interest (ROI) centered on the last known pupil elliptical template. This ringlike region, defined by inner and outer scaled ellipses of the template, efficiently isolates events most likely to originate from the moving pupil boundary, discarding irrelevant background noise and events from other eye parts. This selection step is critical for both efficiency and accuracy, as these events generated by moving the pupil boundary are significant to infer the tiny translational motion within a short time.

The core of the updating mechanism is an iterative point-to-edge matching algorithm that refines the template's center [67]. For a set of candidate points P, the algorithm computes the translation vector T that minimizes the sum of squared distances between the translated candidates and their closest corresponding points on the existing ellipse boundary, Q. This is expressed as:

$$\min_{T} E(T) = \min_{T} \frac{1}{N} \sum_{i=1}^{N} |q_i - (p_i + T)|^2 \tag{3.3}$$

This minimization is solved iteratively. In each iteration, the algorithm calculates the average displacement vector, ΔT, from the candidate points to their projections on the current template boundary. This mean displacement is then added to the total translation T:

$$T_{k+1} = T_k + \Delta \bar{T}_k \tag{3.4}$$

This process is repeated until the change in translation converges, at which point the final vector T is applied to the template's center. For stability, only the center of the ellipse is updated, while its shape and orientation are preserved from the last robust detection. This method provides a computationally inexpensive yet robust way to track motion between complete detection cycles.

Finally, a quantitative fit score is calculated to assess the quality of the update. This score measures the consensus between the new event data and the updated ellipse, effectively evaluating how well the events conform to the template's boundary. Since the point-to-edge algorithm maintains the template shape, the updated template may fail to fit the real pupil boundary, when the pupil moves far from the center of the FoV and its projection on the FoV changes dramatically during a saccade. This score then serves as a critical safeguard, enabling the system to be aware of tracking degradation or failure when the fit score is under the threshold. In this case, the detection mode should be called back to refresh the template's shape.

3.5 Thorough Integration and Control Logic

The complete eye tracking system architecture (i.e., the green part in Figure 3.1) is a bimodal pipeline that dynamically switches between the robust detection mode and the rapid updating mode to achieve an optimal balance of accuracy, frequency, and latency. The control logic governing this transition is fundamental to the system's performance and resilience.

The pipeline begins in detection mode. It accumulates a large slice of events (e.g., 5000) to construct a dense, high-quality SITS representation. This is fed into the PupilUNet to perform segmentation and ellipse fitting, as detailed in Section 3.3. The primary goal of this mode is to establish a highly accurate and reliable initial pupil template. The use of a larger event window ensures that the input to the neural network is sufficiently rich to overcome noise and produce a confident detection.

Upon a successful detection, the system immediately transitions into updating mode. The event slicer is dynamically reconfigured to process much smaller packets of events (e.g., 500). This drastic reduction in slice size is the key to enabling kilo-Hertz tracking frequencies and minimizing latency to the millisecond level. In this mode, the system applies the efficient template updating algorithm described in Section 3.4 to each incoming slice.

A critical component of the integration is the monitoring mechanism that triggers a transition from updating back to detection mode. After each updating cycle, the quality of the updated template is quantified using a fit score, which measures the mean absolute distance of candidate events from the ellipse boundary in a normalized coordinate space. This score, adapted from the method proposed in E-Gaze [32], is calculated as follows:

$$S_{\text{fit}} = 1 - \frac{1}{n} \sum_{i=1}^{n} \left| \|\mathbf{p}'_i\|_2 - 1 \right| \tag{3.5}$$

where n is the number of candidate events and $\mathbf{p}'_i$ is the coordinate of the i-th event in the normalized, axis-aligned space of the ellipse. This normalized coordinate is derived by translating the event to the ellipse's center, rotating it

into the ellipse's frame of reference, and scaling by the semi-axis lengths:

$$\mathbf{p}'_i = \begin{pmatrix} x'_i/w_p \\ y'_i/h_p \end{pmatrix} \quad \text{with} \quad \begin{pmatrix} x'_i \\ y'_i \end{pmatrix} = \begin{pmatrix} \cos\phi & \sin\phi \\ -\sin\phi & \cos\phi \end{pmatrix} \begin{pmatrix} x_i - x_p \\ y_i - y_p \end{pmatrix} \tag{3.6}$$

In these equations:

- S_{fit} is the resulting fit score, with a value of 1 indicating a perfect fit.
- n is the total number of candidate events.
- (x_i, y_i) are the original coordinates of the i-th event.
- (x_p, y_p) is the center of the ellipse template.
- (w_p, h_p) are the semi-axis lengths (width, height) of the ellipse.
- ϕ is the rotation angle of the ellipse in radians.
- (x'_i, y'_i) are the coordinates of the event after translation and rotation.

If this score falls below a predefined threshold, it signals a potential loss of tracking fidelity, which may be caused by pupil occlusion, rapid saccades towards the edge of FoV, or accumulated tracking drift. In response to this low-quality assessment, the control logic immediately reverts the pipeline to the robust detection mode. It resets the event slicer to the larger window size and attempts to re-detect the pupil from scratch using the PupilUNet detector.

This dual-state architecture combines the strengths of both approaches. It leverages the high accuracy and robustness of deep learning-based segmentation to anchor the tracking process, while exploiting the speed and efficiency of a lightweight iterative updater for low-latency, high-frequency refinement. The shifting logic ensures that the system can operate at maximum speed during reliable tracking and can recover from failures, resulting in a pipeline that is fast, accurate, and robust.

Chapter 4

Experiments and Evaluation

This chapter presents detailed implementations of the methods in Chapter 3 and a comprehensive evaluation of the proposed event-based eye tracking pipeline. The experiments and evaluation are detailed in four sections:

- Section 4.1: the data augmentation process was accomplished by training and assessing the frame-based segmentation model used for annotation expansion, followed by the preparation of the final event-annotation dataset for the PupilUNet model, including the data caching and sampling strategies.
- Section 4.2: the training, validation and performance profiling of the proposed PupilUNet model were conducted. After that, the overall pupil detector, including elliptical template generation, was tested.
- Section 4.3: the algorithm of pupil updating was implemented and optimized by vectorized operations.
- Section 4.4: the performance of the fully integrated detection-updating pipeline was evaluated on real-world event streams, assessing its accuracy, latency, and operational characteristics.

The computing device used in this thesis was equipped with an AMD Ryzen 5800H CPU and an NVIDIA RTX 3070 Laptop. The experiments were conducted in an environment with Python 3.11 and Pytorch 2.2.2+cu118. The EV-Eye dataset can be downloaded from
`https://1drv.ms/f/s!Ar4TcaawWPssqmu-0vJ45vYR3OHw`. The code implementation can be found in the repository
`https://github.com/SolarPlover/TUDelft_Thesis/tree/DL`.

4.1 Dataset Augmentation and Preparation

A large dataset size is necessary for a stable and robust segmentation model. The key module in the eye tracking system, PupilUNet, requires abundant densely annotated samples consisting of frame-like event representations and

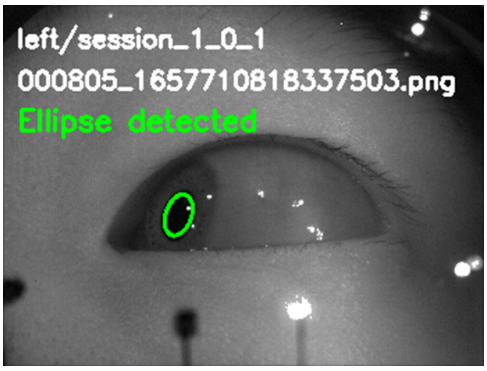


Figure 4.1: **Near-eye grayscale frame segmentation example. The green ellipse outlines the pupil region.**

pupil boundary label mask pairs to predict whether an event is related to the moving pupil boundary. Following the method in Section 3.2, a multi-stage data augmentation and preparation pipeline is thus established to generate a large-scale, high-quality dataset for training the PupilUNet.

4.1.1 Train and Validation of Frame-based Segmentation

The EV-Eye dataset provides 9011 ground truth pupil annotation pairs and many more unannotated near-eye grayscale frames, which can be leveraged to expand the number of reference pupil masks at corresponding timestamps. The foundation of the data augmentation strategy is a frame-based U-Net model, as described in Section 3.2.1, trained to generate new pupil mask annotations. This model was trained on the initial ground truth data provided by the EV-Eye dataset, which consisted of 9,011 frame samples labeled with binary masks of the same 346×260 resolution as the frames. The png masks contained highlighted elliptical pupil regions extracted from the hdf5 files in the `Data_davis_labelled_with_mask` folder. This dataset was then split into a training set of 7,208 samples and a validation set of 1,803 samples.

The frame-based segmentation model, a classic U-Net architecture with a standard five-level encoder-decoder structure and 17.3 million trainable parameters, was trained for 70 epochs using the PyTorch Lightning framework. The network had 1 input channel (grayscale frames) and 2 pixel-level segmentation classes (pupil region or not). An Adam optimizer with a learning rate of 2×10^{-3} and a weight decay of 1×10^{-5} was employed, and 16-bit automatic mixed precision (AMP) was used to accelerate the training. Validation was performed every five epochs to monitor the mean distance between predicted and reference pupil centers, saving the best-performing models. The training converged successfully, achieving a best validation mean distance of 0.6148 pixels at epoch 65. The model also demonstrated exceptional accuracy on other pixel-error thresholds, reaching 100% for P3 accuracy (P3: pixel error between predicted and reference pupil centers ≤ 3 pixels) and 86.4% for P1 accuracy (pixel error ≤ 1 pixel). The low mean distance error on the validation set confirmed that the model was sufficiently accurate and robust to serve as an "auto-annotator." Subsequently, this trained model was deployed to predict pupil masks on over

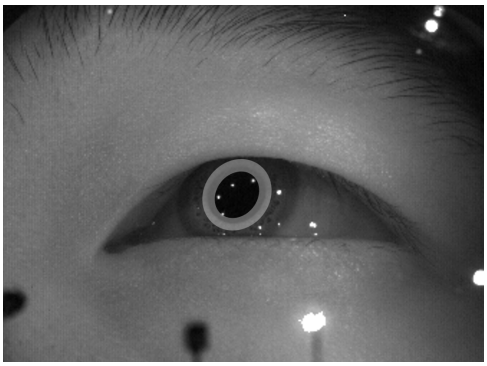


Figure 4.2: **Near-eye grayscale frame with dilated donut-shaped mask. The gray mask was then used as the label in the PupilUNet dataset.**

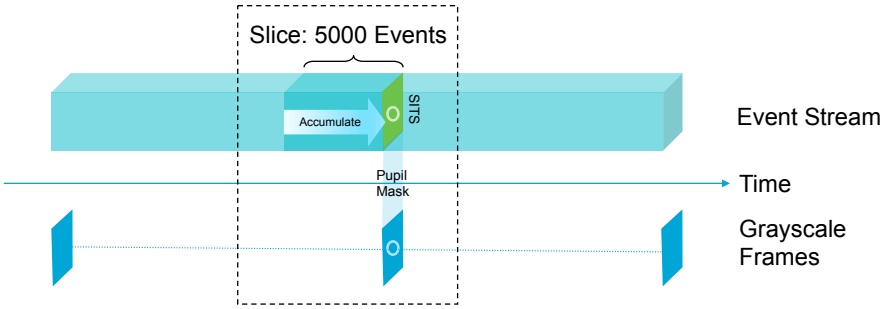


Figure 4.3: **The process to pair a pupil mask with the corresponding SITS to form a sample in the PupilUNet dataset. A white donut-shaped pupil boundary region was segmented by the frame-based segmentation model, serving as the label of the sample. The sample data was the dual-channel SITS accumulated from the causal slice of 5000 events preceding the corresponding timestamp.**

60,000 unannotated frames from the EV-Eye dataset, effectively expanding the pool of available annotations for the next stage. An example of frame-based segmentation results is visualized in Figure 4.1.

4.1.2 PupilUNet Dataset Samples

After the frame-based U-Net model successfully segmented pupil regions in unannotated frames, labels for the PupilUNet dataset were generated by dilating the contours of the elliptical pupil to create donut-shaped masks with 5-pixel thickness. The masks indicated the region where events were likely triggered by the moving pupil boundary, since event cameras are sensitive to edges of moving objects. An example of the dilated mask is demonstrated in Figure 4.2.

With an extensive set of timestamped pupil masks generated, the final dataset for training the PupilUNet model was prepared once event representations were processed following the method in Section 3.2.2. For each new annotation, a corresponding input sample was created by precisely locating its timestamp and extracting a causal window of the 5,000 most recent events preceding it, using

the `bisect_right` function for efficient searching. These events were accumulated into the dual-channel SITS representation with a decay factor $\lambda = 0.001$ in (2.3). The dual-channel SITS was then paired at each corresponding timestamp with the donut-shaped binary mask, where the white ring represents the pupil boundary. This design aligns with the nature of event data, as events are primarily generated by the motion of contours. The process is illustrated in Figure 4.3.

Samples were generated in the PupilUNet dataset, but they should not contribute equally during the training process. Some samples represented pupil occlusions or meaningless noise during fixations, which were not beneficial for PupilUNet to learn the constant features of the pupil region. To quantify the significance of samples, a confidence score for each sample was computed by measuring the density of events within the pupil donut-shaped mask and applying a square root function to normalize the score, making the subsequent training process more robust to noisy data. A visualization of samples from the prepared dataset is shown in Figure 4.4, demonstrating the SITS representation and the corresponding confidence scores for samples of varying quality.

4.1.3 PupilUNet Dataset Preparation

There are 48 users and 388 sessions with event recordings and frames in the EV-Eye dataset. To simplify the PupilUNet dataset, 193 sessions from 24 users were selected to go through the data augmentation process to expand pupil annotations. From the total pool of 193 sessions containing both events and augmented masks, a final dataset of 60,000 samples was randomly compiled and partitioned into 40,000 for PupilUNet training, 10,000 for validation, and 10,000 for testing.

To manage this large-scale dataset efficiently and ensure experimental reproducibility, a sophisticated, three-tier caching strategy was implemented:

- **Annotation Cache:** At the first level, a function scans the data directory once and caches the file paths of all corresponding event recordings and mask directories into a single pickle file. It intelligently checks file modification times to automatically rebuild the cache if the underlying data changes, reducing startup time from minutes to milliseconds on subsequent runs.

- **Persistent Events Cache:** At the second level, the `PersistentEventsCache` class processes the raw text-based event files (events.txt) into a compressed NumPy (.npz) format using `np.savez_compressed`. This binary format allows for significantly faster data loading by bypassing the expensive text parsing process. This approach is fundamental to achieving efficient training workflows.

- **Dataloader Cache:** At the third level, the `dataloader_with_cache` function saves the exact list of samples assigned to the train, validation, and test sets, guaranteeing reproducibility across different experiments.

Among the total $40,000 + 10,000 + 10,000$ samples in the PupilUNet dataset, sample confidences exhibited the distribution in Figure 4.5. Analyzing the confidence statistics, it is clear that most samples could contribute meaningfully

to the pupil feature learning. Some samples with low confidence would hardly impact the training process, as the PupilUNet would consider them as less significant samples and not learn from corrupted samples with heavy pupil occlusion or noise. The final outcome of the event-based segmentation was stable and robust following this strategy.

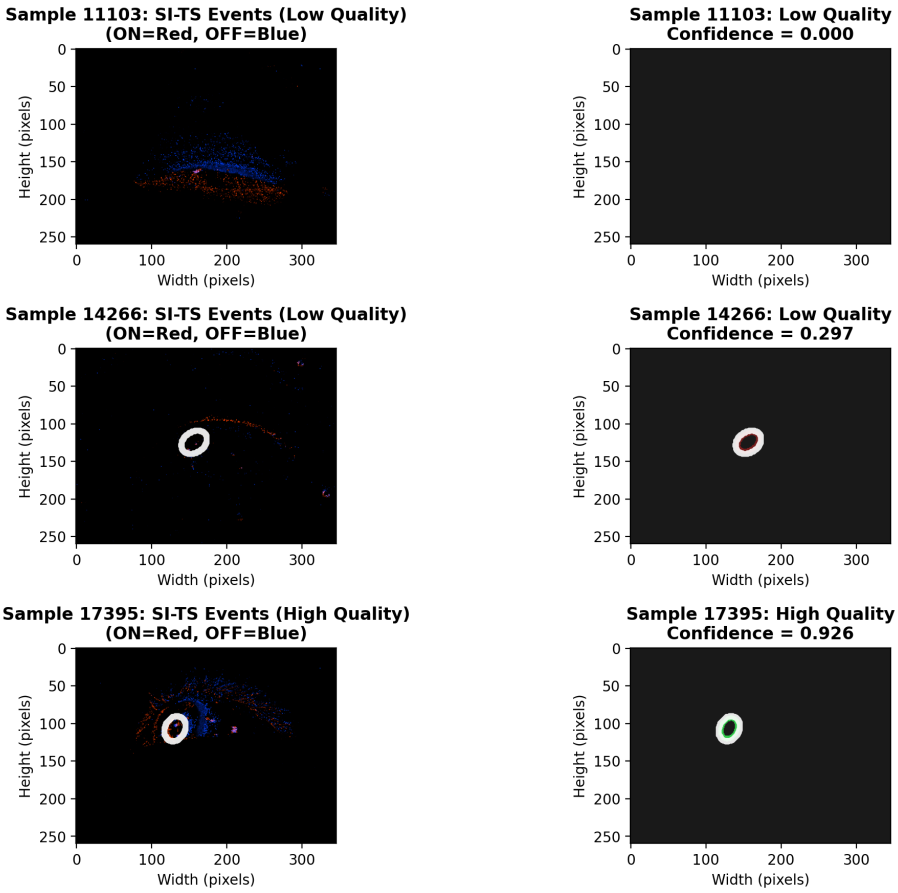


Figure 4.4: **Visualization of three representative samples from the dataset, showcasing low, medium, and high confidence scores. The left column displays the dual-channel SITS representation (ON events in red, OFF in blue) overlaid with the pupil boundary (white ring) for reference. The right column visualizes the pupil boundary masks from the frame-based segmentation model.**

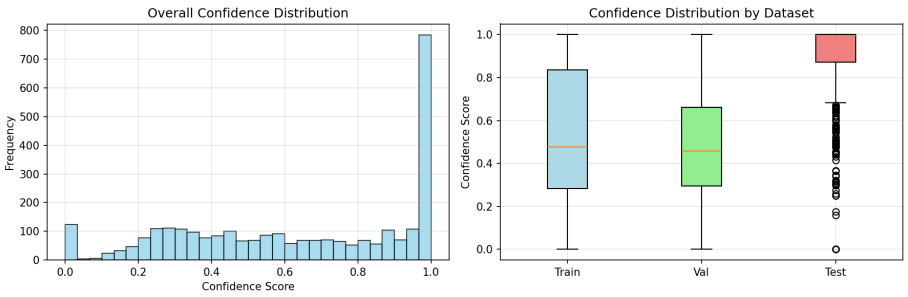


Figure 4.5: **The confidence score statistics of the PupilUNet dataset. Left Plot: The histogram shows the distribution of confidence scores across all samples. Most samples had relatively high confidence and could contribute to the learning. Right Plot: The box plot compares confidence distributions across train, validation, and test sets. The bottom edge of a box represents the 25% percentile (i.e., 25% of samples have confidence below this value). The top edge of a box represents the 75% percentile. The horizontal orange line inside a box is the median. The vertical lines show the range of normal data variation within $\pm 3\sigma$. The individual dots refer to outliers that fall outside the top and bottom bars of the vertical lines.**

4.2 Pupil Detection

4.2.1 PupilUNet Training and Validation

The PupilUNet model was built according to Section 3.3.1 using Pytorch. It was trained and validated using a combination of modern deep learning techniques to maximize performance, efficiency, and robustness. The program incorporated several key implementation details:

- **Model Architecture Implementation:** The PupilUNet class utilized pretrained MobileNetV3-Small as its encoder backbone. The input layer was modified to accept 2 channels corresponding to the bipolar SITS representation. The architecture was intentionally truncated, using only the initial blocks of the backbone up to a $\frac{1}{8}$ downsampling resolution to focus on geometric features. The `DecoderBlock` integrated SimAM specifically for skip connection enhancement. During forward propagation, the upsampled decoder feature was concatenated with the attention-enhanced skip connection. This selective application targeted shallow encoder features containing geometric and edge information crucial for pupil boundary detection.

- **Mixed Precision Training:** By leveraging `torch.cuda.amp.autocast` and `GradScaler`, the training process utilized tensor cores on NVIDIA GPUs for FP16 calculations. This approach significantly accelerated training speed and reduced memory consumption without compromising model accuracy.

- **Confidence-Weighted Compound Loss:** As introduced in Section 3.3.2, compound loss was used with weights of 0.375 for Dice Loss, 17.5 for Focal Loss, and 0.086 for Boundary Loss. The weights were determined to balance their numerical values first and then set their impact as approximately 4:3:3, respectively. The final loss was calculated by directly multiplying the per-sample compound loss by the sample's confidence score before averaging the batch, effectively forcing the model to prioritize high-quality data.

- **Dynamic Learning Rate and Early Stopping:** A `ReduceLROnPlateau` scheduler automatically adjusted the AdamW optimizer's learning rate (initially 1.0×10^{-3}) by monitoring the validation loss. An early stopping mechanism with a patience of three validation checks was implemented to prevent overfitting and terminate training when performance on the validation set ceased to improve.

The model was trained for a maximum of 50 epochs with a batch size of 192 to exploit the GPU. The model achieved its best validation loss of 0.1090 at epoch 24. Although training continued, the validation loss did not improve further over the subsequent three validation checks, triggering the early stopping mechanism. The final saved model was therefore the one from epoch 24, which demonstrated the best generalization capability on unseen data.

4.2.2 PupilUNet Profiling

An analysis of the final PupilUNet architecture revealed a highly efficient and lightweight design, making it well-suited for deployment on resource-constrained devices. The model contained only 0.177 million parameters and required just around 0.553 GFLOPs to infer on a dual-channel SITS frame of 346×260 resolution. This efficiency was a direct result of the truncated MobileNetV3-Small encoder, which focused on essential low-to-mid-level features, and the parameter-free SimAM attention mechanism. A breakdown of the computational cost showed that standard convolutional operations accounted for the majority of the workload, while the overhead from attention and other operations was negligible. This confirmed that the model's design successfully balanced high performance with low computational demand.

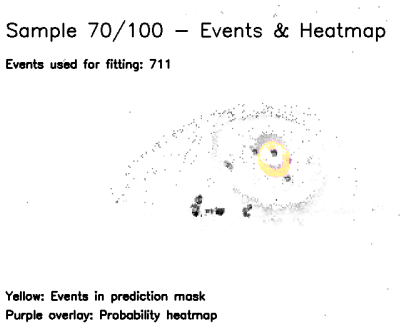
4.2.3 Testing on Pupil Detection

The performance of the trained PupilUNet detector was rigorously evaluated on the 10,000-sample test set from model inference to ellipse fitting. The evaluation pipeline prefigured the detection stage of the final system and proceeded as follows:

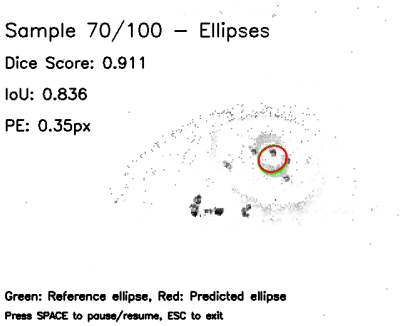
1. **Event Preprocessing**: A slice of 5,000 events was accumulated into a dual-channel SITS tensor following the same algorithm as that in training.
2. **Model Loading & Inference**: The PupilUNet model, set to `eval()` mode, performed inference without gradient calculation, dropout, or batch-norm processes to generate a probability map of pupil boundary.
3. **Event Filtering**: The map was binarized with a threshold of 0.9 to create a preliminary mask that was the most likely to cover the pupil boundary. The raw events that fell within the mask were triggered by the moving pupil boundary.
4. **Precise Ellipse Fitting:** The `fit_ellipse_from_events` function fitted an ellipse directly to the coordinates of the pupil boundary events. This function leveraged the high spatiotemporal precision of the original event data and used probability weighting via `np.percentile` to focus the `cv2.fitEllipse` algorithm on the most confident event points.
5. **Posterior Check:** The `posterior_ellipse_filtering` function validated the fitted ellipse against physiological constraints, including axis lengths and aspect ratio of the pupil, to ensure the results were plausible, discarding erroneous detections.

The detector returned the elliptical parameters of the pupil template, consisting of the ellipse center coordinate, axis lengths, and rotation degree. Figure 4.6 shows two examples of pupil detection results. The detector's performance on the real test set was evaluated using standard segmentation and detection metrics, as introduced in Table 4.1. The results from testing on 10,000 samples were also listed in Table 4.1.

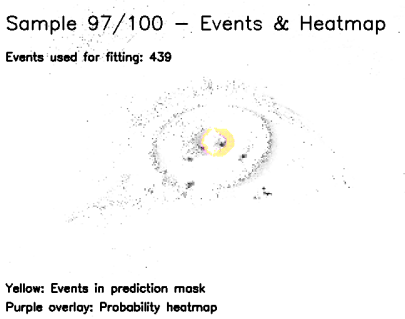
$$\text{Dice}(G, P) = \frac{2\,|G \cap P|}{|G| + |P|}. \tag{4.1}$$



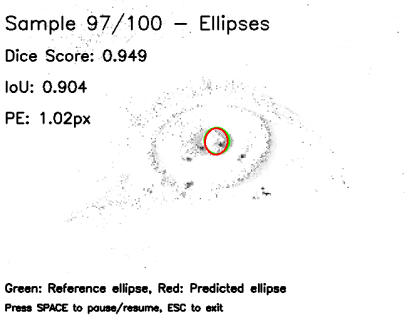
(a) Example 1: Events and Probability Map of Pupil Boundary



(b) Example 1: Reference and Predicted Ellipses



(c) Example 2: Events and Probability Map of Pupil Boundary



(d) Example 2: Reference and Predicted Ellipses

Figure 4.6: **Two examples visualize the key steps in pupil detection. The PupilUNet accepts SITS as input and predicts probability maps of pupil boundary in purple. Events within the binarized masks are dotted in yellow. The ellipses in red are fitted using the yellow event coordinates, compared with the reference elliptical pupil templates in green.**

$$\text{IoU}(G, P) \;=\; \frac{|G \cap P|}{|G \cup P|}. \tag{4.2}$$

$$PE = \sqrt{(x_p - x_g)^2 + (y_p - y_g)^2}. \tag{4.3}$$

The results demonstrated the detector's high accuracy, achieving an average pixel error of just 1.97 pixels and a P5 accuracy of 96.3%. This indicated that in over 96% of valid cases, the detected pupil center was within 5 pixels of the ground truth. The average total latency for an overall detection pipeline (preprocessing and inference) was approximately 80 ms, including detection time of 26.284 ms from SITS to ellipses. The slow preprocessing time included event stream slicing and accumulating in pure Python implementations. There are optimized libraries like dv_processing [20] that can compress the preprocessing time to milliseconds. While tens of milliseconds are still slow for real-time tracking on its own, it is perfectly acceptable for establishing a robust initial

Metrics	Description	Average Values
Dice Coefficient	It measures overlap between the reference pupil region (G) and predicted pupil elliptical template (P), calculation in (4.1).	0.863
Intersection over Union	IoU is the fraction of the combined area covered by both masks that is correctly predicted, (4.2). It penalizes both missed ground-truth pixels (false negatives) and extra predicted pixels (false positives) equally.	77.0%
Pixel Error	PE measures Euclidean distance in pixels between predicted pupil center (x_p, y_p) and reference center (x_g, y_g) as (4.3).	1.96
P3 P5 P10	Pd is the percentage of test samples whose predicted pupil center lies within d pixels of the reference center.	85.7% 96.3% 99.5%
Detection Time	The time includes PupilUNet inference time on dual-channel SITS, ellipse fitting and filtering time.	26.3 ms
Detection Success Rate	It is the percentage of successfully detected samples within all samples with references.	86.5%

Table 4.1: **Detector evaluation metrics and their introduction.**

template or for re-detection after tracking loss. The overall tracking frequency would not be encumbered too much because of the following fast updating.

4.3 Pupil Template Updating with Optimization

To achieve the ultra-low latency required for real-time tracking, the computationally intensive PupilUNet detector is complemented by a rapid template updating mechanism. The implementation of pupil template updating is a highly optimized version of the iterative point-to-edge matching algorithm described in Section 3.4. Key implementation details include:

- **Candidate Selection:** The `select_candidates_donut` function efficiently filters events. It transforms all event coordinates into the ellipse's local frame using a single matrix multiplication. It then calculates the normalized radial distance squared for each point, allowing it to filter events within the donut region (defined by `inner_scale`=0.8 and `outer_scale`=1.2) without resorting to computationally expensive square root operations.

- **Vectorized Update Algorithm:** The `template_update_fast` function is the core of the optimization. It replaces a slow, iterative Python loop with a fully vectorized approach. The `closest_points_on_ellipse_vectorized` function processes all candidate points simultaneously. It pre-computes rotation matrices and applies the Newton-Raphson method to entire NumPy arrays. An `active_newton_mask` is used to manage convergence for each point individually within the vectorized loop, ensuring efficiency.

- **Quality Assessment:** The `check_template` function implements the fit score (3.5) from E-Gaze. It normalizes all candidate event coordinates into the updated ellipse's axis-aligned space and calculates the mean absolute deviation from a unit circle. A score close to 1 indicates a good fit. In the final pipeline, a threshold of 0.95 is used to accept or reject an update.

This vectorized implementation reduced the time taken to update the template's position to the sub-millisecond level as low as 0.978 ms on average, a critical factor in enabling kilo-Hertz tracking frequencies.

4.4 Integrated Eye Tracking Pipeline and Evaluation

The final stage of the evaluation assessed the complete, integrated eye tracking system on raw data streams in the EV-Eye dataset with the dv_processing library [20] to accelerate the preprocessing of the event recording. The pipeline, composed of an event recording parsing program and a pupil detector/updater, demonstrated the cooperation between the detection and updating modes. Here are the details of the thorough integration:

- **System Composition:** The main loop in the pipeline uses the high-performance `dv.EventStreamSlicer` and `dv.SpeedInvariantTimeSurface` from the dv-processing library [20] to handle the demanding task of slicing and accumulating the event stream. This offloads the most time-consuming preprocessing operations from Python to optimized C++ code. It is claimed that the peak throughput of SITS accumulation can reach up to 36.8 MegaEvent/s on their system setup, while the prepossessing costs less than 1ms in my experiments.

- **Dynamic State shifting:** A boolean flag, `detected`, indicates the system's state between detection and updating. The main loop continuously checks this flag. If the pupil is lost (`detected` is False), it configures the slicer for large, 5,000-event packets for robust detection using `slicer.modifyNumberInterval`. Once a pupil is successfully detected (Figure 4.7(a)), the flag is set to True, and the slicer is immediately reconfigured for small, 500-event packets for rapid updating (Figure 4.7(b)). If an update fails the quality check of (3.5) (3.6), the flag is reset, and the system reverts to detection mode.

- **Callback Logic:** The `slicing_callback` function acts as the control hub. It receives event packets from the slicer and, based on the detected flag, directs the data to either the full detection pipeline

(a) A pupil was detected in the eye tracking pipeline.

(b) The pupil template was updated in the eye tracking pipeline.

Figure 4.7: **Two states in the integrated pipeline.**

Metric	Value
Operational Statistics	
Total Processed Slices	3,774
Detection / Update Ratio	38.6% / 61.4%
Latency Statistics	
Average Detection Latency	28.1 ms
Average Update Latency	1.04 ms
Accuracy vs. Ground Truth	
P5 Accuracy	85.2%
P10 Accuracy	96.3%

Table 4.2: **Integrated pipeline performance on a sample recording.**

(`get_pupil_each_frame`) or the lightweight update pipeline

(`select_candidates_donut`, `template_update_fast`, `check_template`).

The performance of the integrated pipeline on a sample recording (dvSave-2022_07_13_11_18_03.aedat4) is summarized in Table 4.2. The system processed 3,774 event slices, with the robust detector being invoked 38.6% of the time and the rapid updater 61.4% of the time. The average latency for a complete detection cycle was 28.10 ms. Most critically, the average latency for a template update was merely 1.04 ms, demonstrating the effectiveness of the architecture. Considering the eye tracking was dominated by updating, and the event slicing time was less than its computational latency, the peak tracking frequency was up to 961.5 Hz.

To evaluate the pipeline's end-to-end accuracy, the sequence of generated templates was compared to the manually annotated ground truth for that recording session. Since the timestamps of the generated templates and ground truth annotations do not align perfectly, the `match_annotations_to_templates` function estimates the pupil position at each ground truth timestamp by linearly interpolating between the two temporally closest templates generated by the pipeline. The PE was then calculated between this interpolated position and the ground truth.

The results show that the integrated system maintained high accuracy, with 85.2% of the estimations having a pixel error of 5 pixels or less, and 96.3% having an error of 10 pixels or less. This confirms that the proposed pipeline not only operates at very high frequencies due to its efficient updater but also preserves the high accuracy established by its robust deep learning-based detector.

Chapter 5

Discussion and Comparison

5.1 Result Discussion

The results presented in Chapter 4 validate the proposed eye tracking system with a critical and successful engineering trade-off. The discussion on the results includes analyzing the relationship between the performance of the detector, the efficiency of the updater, and the final performance of the integrated pipeline.

Considering the detection mode at first, the pupil detector equipped with the PupilUNet segmentation model achieved a P5 accuracy of 96.3% when tested on individual samples. The SITS representation proved highly effective at preprocessing the event stream, providing the network with a stable, normalized representation implying learnable pupil boundary features regardless of the pupil motion. The PupilUNet architecture itself, as a key contribution of this work with its truncated encoder and attention-augmented skip connections, was able to learn this representation effectively. The training was enabled by the real-data-driven augmentation framework, which provided the large-scale dataset necessary for robust learning. The average pixel error of 1.97 pixels demonstrates that this lightweight, segmentation-based approach can achieve competing precision when the input data is properly conditioned.

However, the 28.10 ms average detection latency makes it clear that relying solely on the PupilUNet for every slice would cap the system's tracking frequency at around 36 Hz, falling far short of the kilo-Hertz goal required for next-generation applications. Although latency could potentially be further reduced through inference optimizations such as TensorRT like that in [7], fresh detections for every 5000-event slice are unnecessary. The high temporal resolution of event cameras ensures minimal time intervals between slices, limiting pupil movement and allowing reliable region of interest determination from the previous template. This leads to the detector's intended role to serve as a robust, high-accuracy anchor and recovery mechanism, not as the primary real-time tracker.

This is where the performance of the rapid updating mode becomes critical. The measured average update latency of around 1 ms is significant for the pipeline's high-frequency capabilities. This millisecond-level performance was achieved through the optimization technique of a vectorized implementation, as detailed in Section 4.3, which made it possible to process entire batches

of candidate points simultaneously. The testing sample of an event recording in Section 4.4 confirmed the success of the dual-state pipeline, as the system spent most (61.4%) of its time in this ultra-fast mode. This figure shows that the pipeline is not constantly re-detecting. The high-quality templates provided by the PupilUNet, combined with an effective fit_score threshold, allow the system to maintain a stable track for extended periods using the computationally trivial updater.

Finally, the integrated pipeline's end-to-end accuracy reflects a balance between latency and precision. The measured reduction in P5 accuracy compared with the standalone detector arises from the design choice to prioritize continuous, high-frequency operation. Minor drift accumulates during rapid update phases before the control logic triggers a full re-detection. These brief intervals of deviation are the cost of achieving a peak frequency above 960 Hz while sustaining 85.2% accuracy within 5 pixels. Additional performance drop stems from the different evaluation protocols: the standalone detector is assessed on isolated samples with explicit references, whereas the integrated pipeline is tested on continuous event recordings with unevenly sampled ground truth and interpolation between annotations, introducing further error. These factors mean the integrated test understates the pipeline's true performance. Crucially, the segmentation-based approach offers explainability absent in end-to-end black-box regression, enabling post-analysis to localize failure causes to the event data, segmentation mask, or geometric fitting. Targeted improvements, including adaptive re-detection policies, refined interpolation and annotation, and enhanced segmentation, can narrow the accuracy gap without losing the substantial gains in real-time throughput.

5.2 Comparison against Competing Methods in Near-eye Event-based Eye Tracking

To better demonstrate the performance and efficiency of the proposed pipeline, it is compared against three representative near-eye event-based eye tracking methods: EV-Eye [67], FACET [7], and TennSt [49]. Each represents a distinct design paradigm: template updating with mixed modalities, elliptical parameter regressions, and regression-based deep neural networks, respectively. The primary metrics for the comparison are P5 accuracy, count of model parameters and GFLOPs, measuring the accuracy and efficiency of the methods. The inference time varies across different computing platforms, so model size and operation count are used for efficiency evaluation.

As introduced in Section 2.6, EV-Eye is one of the fundamental large-scale systems designed for near-eye tracking, employing a frame-event hybrid pipeline. The EV-Eye method is inherently dependent on frame input, which increases latency for image segmentation. In contrast, the proposed PupilUNet pipeline operates in a purely event-driven manner, eliminating the need for frames while still achieving stable template updates. Quantitatively, EV-Eye reports a P5 accuracy of around 99.91% with 17.27 M parameters and 40.11 GFLOPs of the detector, while the PupilUNet method achieves comparable accuracy with much lower computational cost on standard hardware.

FACET advances the field by introducing direct regression to estimate ellipt-

Methods	P5 Accuracy	Parameters	GFLOPs	Latency & Compute Cost
EV-Eye [67]	99.91%	17.27 M	40.11	High(Detection) Low(Updating)
FACET [7]	99.98%	3.92 M	3.44	Medium
TennSt [49]	96.77%	0.81 M	5.49	Medium
PupilUNet Pipeline	96.30%	0.18 M	0.55	Low

Table 5.1: **Comparison of event-based eye tracking methods.**

ical parameters from causal event volumes, outputting pupil shape parameters without additional fitting steps. It demonstrates high accuracy (P5 accuracy of 99.98%) and lower model complexity (3.92 M parameters) than EV-Eye's image segmentation network. However, FACET relies on a relatively heavy Feature Pyramid Network backbone and is optimized primarily for high-end GPUs with TensorRT support. The proposed PupilUNet pipeline, by contrast, uses a lightweight segmentation-plus-update strategy that runs efficiently on middle-end devices. While FACET achieves slightly lower pixel error, the PupilUNet system demonstrates better robustness in occlusion scenarios due to explicit segmentation and template fitting. It is more suitable for mobile or XR deployment where computational budgets are tight.

TennSt ranked top 3 in the [61] eye tracking challenge. It represents regression-based pupil center tracking, trained on the 3ET++ benchmark. By directly mapping event volumes to pupil center coordinates, it achieves competitive accuracy under clean conditions. Nevertheless, TennSt is vulnerable to centroid shifts when partial occlusions occur, since no geometric constraints are enforced. The PupilUNet system addresses this weakness by combining segmentation with explicit ellipse fitting, preserving shape continuity and resilience to occlusions. Although TennSt has as few as 0.81 M parameters and 5.49 GFLOPs for inference, my detector is even more lightweight. The lightweight updating mechanism further ensures kHz-level update rates with minimal overhead.

Table 5.1 provides a comprehensive comparison of the methods discussed above. The results highlight that the proposed pipeline offers a balanced trade-off between accuracy, robustness, and computational efficiency, positioning it as a practical candidate for eye tracking applications.

Chapter 6

Conclusion

This thesis presented an effective, efficient, purely event-based pupil tracking pipeline designed to meet the demanding requirements of next-generation eye tracking applications such as Extended Reality (XR). The research was motivated by the inherent limitations of traditional frame-based systems and existing event-based methods, which often struggle to simultaneously achieve high tracking frequency, low latency, robust accuracy, and computational efficiency suitable for resource-constrained mobile devices. The primary contribution of this work is an efficient, explainable, dual-state pupil detection-tracking framework that successfully navigates the critical trade-off between the robustness of deep learning-based detection and the speed of pupil template updating.

The development of this pipeline was supported by several key innovations. First, to overcome the scarcity of annotated event-based eye tracking data, a real-data-driven data augmentation framework was established. This involved training a frame-based segmentation model to generate a large-scale, high-quality dataset of pupil masks, which was then paired with Speed-Invariant Time Surface (SITS) representations for training the core event-based model. Second, a lightweight, attention-augmented PupilUNet architecture was designed. By utilizing a truncated MobileNetV3 Small encoder and a parameter-free attention mechanism, this model achieved robust pupil segmentation with minimal computational overhead, focusing specifically on the low-to-mid-level geometric features crucial for pupil boundary detection. Finally, this robust detector was integrated with a highly optimized, vectorized template updater. A dynamic control logic, based on a quantitative fit score, intelligently switches between the two modes, ensuring that the system can anchor its tracking with high accuracy and then maintain it at kilo-Hertz frequencies with minimal latency.

Experimental evaluation validated the effectiveness of this dual-state approach. The standalone PupilUNet detector demonstrated high accuracy, achieving a P5 accuracy of 96.3% on the test set. The rapid updater, on the other hand, achieved an average latency of approximately 1 ms. The fully integrated pipeline successfully combined these strengths, maintaining a P5 accuracy of 85.2% while operating at a peak frequency of over 960 Hz. This result confirms that the system's architecture makes an effective engineering trade-off, sacrificing a minor amount of peak accuracy for a massive gain in real-time operational speed and efficiency. The segmentation-based methodology also provides inherent explainability, allowing for easier debugging of tracking failures compared

to end-to-end regression models.

Despite its success, this work has identified several limitations that provide clear directions for future research. The primary limitation is the slight drop in accuracy in the integrated pipeline, which is an inherent consequence of the latency-accuracy trade-off. Furthermore, the initial implementation of SITS accumulation in Python proved to be a significant performance bottleneck, underscoring the necessity of optimized, low-level libraries for real-time preprocessing in event-based vision systems.

Future work will focus on three main areas. First, to ensure generalization and robustness across different datasets, the proposed pipeline will be migrated and tested on other event-based eye tracking datasets. Second, further performance optimizations will be explored. the PupilUNet detector could be accelerated using inference optimization frameworks like NVIDIA TensorRT, while the template updater could be implemented in a faster backend language to reduce latency further. Finally, the tracking algorithm itself can be enhanced by incorporating a Kalman filter into the updating stage. This would enable predictive tracking, leading to smoother template updates and improved resilience to brief occlusions or tracking drift, ultimately pushing the boundaries of what is possible for efficient and robust eye tracking on mobile and embedded systems.

Bibliography

[1] Amer Al-Rahayfeh and Miad Faezipour. Eye tracking and head movement detection: A state-of-art survey. *IEEE J. Transl. Eng. Health Med.*, 1:1–12, November 2013.

[2] Anastasios N. Angelopoulos, Julien N. P. Martel, Amit P. S. Kohli, Jörg Conradt, and Gordon Wetzstein. Event-based near-eye gaze tracking beyond 10,000 hz. *IEEE Trans. Vis. Comput. Graph.*, 27(5):2577–2586, May 2021.

[3] Apple. Introducing apple vision pro: Apple's first spatial computer. `https://www.apple.com/newsroom/2023/06/introducing-apple-vision-pro/`, 2023. Last accessed: Aug. 6, 2025.

[4] Pietro Bonazzi, Sizhen Bian, Giovanni Lippolis, Yawei Li, Sadique Sheik, and Michele Magno. Retina: Low-power eye tracking with event camera and spiking hardware. In *Proc. CVPRW*, Vancouver, Canada, June 2024. IEEE.

[5] Qinyu Chen, Zuowen Wang, Shih-Chii Liu, and Chang Gao. 3ET: Efficient event-based eye tracking using a change-based convlstm network. In *IEEE BioCAS*, pages 1–5, Toronto, Canada, October 2023. IEEE.

[6] 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), June 2023.

[7] 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. `https://arxiv.org/abs/2409.15584`, 2024. Last accessed: Aug. 7, 2025.

[8] Stefan Dowiasch, Peter Wolf, and Friederike Bremmer. Quantitative comparison of a mobile and a stationary video-based eye-tracker. *Behav. Res. Methods*, 52(2):667–680, June 2020.

[9] Huiyu Duan, Guangtao Zhai, Xiongkuo Min, Zhaohui Che, Yi Fang, Xiaokang Yang, Jesús Gutiérrez, and Patrick L. Le Callet. A dataset of eye movements for the children with autism spectrum disorder. In *Proc. Multimedia Syst.*, pages 255–260, Amherst, MA, USA, June 2019. ACM.

[10] David S. Dunn. Required accuracy of gaze tracking for varifocal displays. In *Proc. Virtual Reality 3D User Interfaces*, pages 1838–1842, Osaka, Japan, March 2019. IEEE.

[11] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proc. KDD*, Portland, OR, USA, August 2–4, 1996. AAAI Press.

[12] Guillermo Gallego, Tobi Delbruck, Garrick Orchard, Chiara Bartolozzi, Brian Taba, Andrea Censi, Stefan Leutenegger, Andrew Davison, Joerg Conradt, Kostas Daniilidis, and Davide Scaramuzza. Event-based vision. `https://www.ifi.uzh.ch/en/rpg/research/research_dvs.html`, 2023. Last accessed: Aug. 9, 2025.

[13] Guillermo Gallego, Tobi Delbruck, Garrick Orchard, Chiara Bartolozzi, Brian Taba, Andrea Censi, Stefan Leutenegger, Andrew J. Davison, Jörg Conradt, Kostas Daniilidis, and Davide Scaramuzza. Event-based vision: A survey. *IEEE Trans. Pattern Anal. Mach. Intell.*, 44(1), January 2022.

[14] Elias Daniel Guestrin and Moshe Eizenman. General theory of remote gaze estimation using the pupil center and corneal reflections. *IEEE Trans. Biomed. Eng.*, 53(6):1124–1133, June 2006.

[15] Ewald Hering. *The Theory of Binocular Vision*. Springer, New York, NY, USA, 1977.

[16] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, Quoc V. Le, and Hartwig Adam. Searching for MobileNetV3. `https://arxiv.org/abs/1905.02244`, 2019. Last accessed: Aug. 10, 2025.

[17] Khadija Iddrisu, Waseem Shariff, Noel E. O'Connor, Joseph Lemley, and Suzanne Little. Evaluating image-based face and eye tracking with event cameras. In *Proc. Comput. Vis.*, page 10357, Seoul, South Korea, October 2024. IEEE.

[18] Vladimir Iglovikov and Alexey Shvets. TernausNet: U-net with vgg11 encoder pre-trained on imagenet for image segmentation. `https://arxiv.org/abs/1801.05746`, 2018. Last accessed: Aug. 10, 2025.

[19] iniVation AG. Davis 346. `https://inivation.com/wp-content/uploads/2019/08/DAVIS346.pdf`, 2019. Last accessed: Aug. 9, 2025.

[20] iniVation AG. Dv-processing: Event accumulation. `https://dv-processing.inivation.com/master/accumulators.html`, 2022. Last accessed: Aug. 9, 2025.

[21] Yizhou Jiang, Wenwei Wang, Lei Yu, and Chu He. Eye tracking based on event camera and spiking neural network. *Electronics*, 13(14):2879, July 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, August 2017.

[23] Hoel Kervadec, Jihene Bouchtiba, Christian Desrosiers, Éric Granger, José Dolz, and Ismail Ben Ayed. Boundary loss for highly unbalanced segmentation. In *Proc. MIDL*, London, UK, July 8–10, 2019. PMLR.

[24] Jonghyun Kim, Michael Stengel, Andrew Majercik, Steven De Mello, Daniel Dunn, Sami Laine, Michael McGuire, and David Luebke. Nvgaze: An anatomically-informed dataset for low-latency, near-eye gaze estimation. In *Proc. CHI*, Glasgow, Scotland, UK, May 2019. ACM.

[25] Mingyu Kim, Jiwon Lee, Changyu Jeon, and Jinmo Kim. A study on interaction of gaze pointer-based user interface in mobile virtual reality environment. *Symmetry*, 9(9):189, September 2017.

[26] Oleg V. Komogortsev, Sampath Jayarathna, Cecilia R. Aragon, and Mechehoul Mahmoud. Biometric identification via an oculomotor plant mathematical model. In *Proc. ACM Symp. Eye-Tracking Res. Appl.*, pages 57–60, Austin, TX, USA, March 2010. ACM.

[27] Eileen Kowler. Eye movements: The past 25 years. *Vision Res.*, 51(13):1457–1483, July 2011.

[28] Tiffany C. K. Kwok, Peter Kiefer, Victor R. Schinazi, Benjamin Adams, and Martin Raubal. Gaze-guided narratives: Adapting audio guide content to gaze in virtual and real environments. In *Proc. ACM CHI Conf. Hum. Factors Comput. Syst.*, pages 1–12, Glasgow, Scotland, UK, May 2019. ACM.

[29] Mikko Kytö, Barrett Ens, Thammathip Piumsomboon, Gun A. Lee, and Mark Billinghurst. Pinpointing: Precise head- and eye-based target selection for augmented reality. In *Proc. CHI Conf. Hum. Factors Comput. Syst.*, pages 81:1–81:14, Montreal, QC, Canada, April 2018. ACM.

[30] Chih-Chuan Lai, Sheng-Wen Shih, and Yi-Ping Hung. Hybrid method for 3-d gaze tracking using glint and contour features. *IEEE Transactions on Circuits and Systems for Video Technology*, 25(1), January 2015.

[31] Kang Il Lee, Jung Ho Jeon, and Byung Cheol Song. Deep learning-based pupil center detection for fast and accurate eye tracking system. In *Proc. ECCV*, Glasgow, UK, August 2020. Springer.

[32] Nealson Li, Muya Chang, and Arijit Raychowdhury. E-Gaze: Gaze estimation with event camera. *IEEE Trans. Pattern Anal. Mach. Intell.*, 46(7):4796–4811, July 2024.

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

[34] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proc. ICCV*, Venice, Italy, October 22–29, 2017. IEEE.

[35] Min Liu and Tobias Delbrück. Adaptive time-slice block-matching optical flow algorithm for dynamic vision sensors. In *BMVC*, Newcastle upon Tyne, UK, September 3–6, 2018. BMVC.

[36] Dillon Lohr and Oleg V. Komogortsev. Eye know you too: Toward viable end-to-end eye movement biometrics for user authentication. *IEEE Trans. Inf. Forensics Secur.*, 17:3151–3164, January 2022.

[37] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proc. CVPR*, Boston, MA, USA, June 2015. IEEE.

[38] Marco Macanovic, Fabian Chersi, Félix Rutard, Sio-Hoi Ieng, and Ryad Benosman. When conventional machine learning meets neuromorphic engineering: Deep temporal networks (dtnets) a machine learning frawmework allowing to operate on events and frames and implantable on tensor flow like hardware. `https://arxiv.org/abs/1811.07672`, 2018. Last accessed: Aug. 9, 2025.

[39] Jacques Manderscheid, Amos Sironi, Nicolas Bourdis, Davide Migliore, and Vincent Lepetit. Speed invariant time surface for learning to detect corner points with event-based cameras. In *Proc. CVPR*, Long Beach, CA, USA, June 2019. IEEE/CVF.

[40] Maria Laura Mele and Stefano Federici. Gaze and eye-tracking solutions for psychological research. *Cogn. Process.*, 13(S1):261–265, August 2012.

[41] Mark A. Mento. This is how eye tracking technology works. `https://www.bitbrain.com/blog/eye-tracking-technology/`, 2020. Last accessed: Aug. 7, 2025.

[42] Clara Mestre, Josselin Gautier, and Jaume Pujol. Robust eye tracking based on multiple corneal reflections for clinical applications. *Journal of Biomedical Optics*, 23(3), March 2018.

[43] Fausto Milletari, Nassir Navab, and Seyed-Ahmad Ahmadi. V-Net: Fully convolutional neural networks for volumetric medical image segmentation. In *3DV*, Stanford, CA, USA, October 25–28, 2016. IEEE.

[44] Carlos H. Morimoto and Marcio R. M. Mimica. Eye gaze tracking techniques for interactive applications. *Comput. Vis. Image Underst.*, 98(1):4–24, April 2005.

[45] Jian nan Chi, Peng yi Zhang, Si yi Zheng, and Chuang Zhang. Key techniques of eye gaze tracking based on pupil corneal reflection. In *Proc. WRI Global Congress on Intelligent Systems*, volume 2, pages 133–138, Xiamen, China, May 2009. IEEE.

[46] Irmingard M. Neuhann, Barbara A. M. 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. *J. Refract. Surg.*, 26(1):17–27, January 2010.

[47] Zhenjiang Ni, Sio-Hoi Ieng, Christoph Posch, Stéphane Régnier, and Ryad Benosman. Visual tracking using neuromorphic asynchronous event-based cameras. *Neural Comput.*, 27(4):925–953, April 2015.

[48] Omar Oubari, Georgios Exarchakis, Gregor Lenz, Ryad Benosman, and Sio-Hoi Ieng. Efficient spatio-temporal feature clustering for large event-based datasets. *Neuromorphic Computing and Engineering*, 2(4), October 2022.

[49] Yan Ru Pei, Sasskia Brüers, Sébastien Crouzet, Douglas McLelland, and Olivier Coenen. A lightweight spatiotemporal network for online eye tracking with event camera. In *Proc. CVPR*, Seattle, WA, USA, June 16–22, 2024. IEEE.

[50] 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*, Charleston, SC, USA, March 2016. ACM Press.

[51] Martina Poletti and Michele Rucci. A compact field guide to the study of microsaccades: Challenges and functions. *Vision Res.*, 118:83–97, January 2016.

[52] Elena Pretegiani and Lance M. Optican. Eye movements in parkinson's disease and inherited parkinsonian syndromes. *Front. Neurol.*, 8:592, November 2017.

[53] Prophesee. Event-based vision for eye tracking. `https://www.prophesee.ai/event-based-vision-eye-tracking/`, 2025. Accessed: Aug. 7, 2025.

[54] Prophesee. World's fastest & lowest-energy eye-tracking. `https://www.prophesee.ai/event-based-vision-eye-tracking/`, 2025. Last accessed: Aug. 9, 2025.

[55] Somaiieh Rokhsaritalemi, Abolghasem Sadeghi-Niaraki, and Soo-Mi Choi. A review on mixed reality: Current trends, challenges and prospects. *Appl. Sci.*, 10(2), January 2020.

[56] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional networks for biomedical image segmentation. In *Proc. MICCAI*, Munich, Germany, October 2015. Springer.

[57] Argha Sen, Nuwan Sriyantha Bandara, Ila Gokarn, Thivya Kandappu, and Archan Misra. EyeTrAES: Fine-grained, low-latency eye tracking via adaptive event slicing. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 8(4):1–32, December 2024.

[58] Tobii AB. Tobii pro glasses 3: Real-world behavior. captured with precision. `https://www.tobii.com/products/eye-trackers/wearables/tobii-pro-glasses-3`, 2025. Last accessed: Aug. 7, 2025.

[59] Niilo V. Valtakari, Ignace T. C. Hooge, Charlotte Viktorsson, Pär Nyström, Terje Falck-Ytter, and Roy S. Hessels. Eye tracking in human interaction: Possibilities and limitations. *Behav. Res. Methods*, 53(4):1592–1608, August 2021.

[60] Kang Wang and Qiang Ji. Real time eye gaze tracking with 3d deformable eye-face model. In *Proc. ICCV*, Venice, Italy, October 2017. IEEE.

[61] Zuowen Wang, Chang Gao, Zongwei Wu, Marcos V. Conde, Radu Timofte, Shih-Chii Liu, Qinyu Chen, Zheng jun Zha, Wei Zhai, Han Han, Bohao Liao, Yuliang Wu, Zengyu Wan, Zhong Wang, Yang Cao, Ganchao Tan, Jinze Chen, Yan Ru Pei, Sasskia Brüers, Sébastien Crouzet, Douglas McLelland, Oliver Coenen, Baoheng Zhang, Yizhao Gao, Jingyuan Li, Hayden Kwok-Hay So, Philippe Bich, Chiara Boretti, Luciano Prono, Mircea Lică, David Dinucu-Jianu, Cătălin Grîu, Xiaopeng Lin, Hongwei Ren, Bojun Cheng, Xinan Zhang, Valentin Vial, Anthony Yezzi, and James Tsai. Event-based eye tracking. ais 2024 challenge survey. In *Proc. CVPRW.*, pages 5810–5825, Seattle, WA, USA, June 2024. IEEE.

[62] Lingxiao Yang, Ru-Yuan Zhang, Lida Li, and Xiaohua Xie. SimAM: A simple, parameter-free attention module for convolutional neural networks. In *Proc. MLR*, Virtual Event, July 18–24, 2021. PMLR.

[63] Yan Yang, Liyuan Pan, and Liu Liu. Event camera data pre-training. In *Proc. ICCV*, Paris, France, October 2023. IEEE.

[64] Niklas Zdarsky, Stefan Treue, and Moein Esghaei. A deep learning-based approach to video-based eye tracking for human psychophysics. *Front. Hum. Neurosci.*, 15, July 2021.

[65] Haiwei Zhang, Jiqing Zhang, Bo Dong, Pieter Peers, Wenwei Wu, Xiaopeng Wei, Felix Heide, and Xin Yang. In the blink of an eye: Event-based emotion recognition. In *Proc. Human Factors Comput. Syst.*, pages 1–11, Los Angeles, CA, USA, August 2023. ACM.

[66] Tongyu Zhang, Yiran Shen, Guangrong Zhao, Lin Wang, Xiaoming Chen, Lu Bai, and Yuanfeng Zhou. Swift-Eye: Towards anti-blink pupil tracking for precise and robust high-frequency near-eye movement analysis with event cameras. *IEEE Trans. Vis. Comput. Graph.*, 30(5), May 2024.

[67] 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. In *Proc. NeurIPS*, pages 62169–62182, New Orleans, LA, USA, December 2023. NeurIPS Foundation.

[68] Zongwei Zhou, Md Mahfuzur Rahman Siddiquee, Nima Tajbakhsh, and Jianming Liang. UNet++: A nested u-net architecture for medical image segmentation. In *DLMIA and ML-CDS*, Granada, Spain, September 20, 2018. Springer.

[69] Alex Zihao Zhu, Liangzhe Yuan, Kenneth Chaney, and Kostas Daniilidis. Unsupervised event-based optical flow using motion compensation. In *Proc. ECCV Workshops*, Munich, Germany, September 2018. Springer.