



Original article

E-BTS: A low-power Event-driven Blink Tracking System with hardware-software co-optimized design for real-time driver drowsiness detection

Jongin Choi, Eunjin Choi, Seonghyun Choi, Woojoo Lee *

Department of Intelligent Semiconductor Engineering, Chung-Ang University, Seoul, 06974, Republic of Korea

ARTICLE INFO

Keywords:

Eye blink detection
Dynamic vision sensor
DBSCAN
Driver drowsiness detection
Edge computing
Processor architecture
HW-SW co-design
FPGA prototype
Low-power design

ABSTRACT

Existing driver drowsiness detection methods based on RGB cameras and neural networks face critical limitations in variable lighting conditions and impose high computational burdens, rendering them unsuitable for real-time, low-power embedded applications. This paper presents an Event-driven Blink Tracking System (E-BTS) that leverages event-based images, which capture changes in light intensity rather than full-frame data, significantly reducing computational requirements. A key innovation of this work is the separation of blink tracking into two tasks: eye blink detection and eye region detection. Dedicated algorithms and hardware modules were developed for each task and integrated into the specialized hardware accelerator, *BlinkTrack*. By integrating this accelerator into the embedded processor, the developed E-BTS processor achieves real-time performance for blink tracking at 200 FPS. The proposed system was implemented and validated on the FPGA-prototyped E-BTS processor, demonstrating robust performance across diverse lighting conditions and scenarios. Experimental results highlight substantial performance improvements, including a 1275× speedup in blink detection and a 310–496× speedup in eye region detection, while the hardware contributes only 27.2% to the overall processor power consumption. Additionally, quantitative evaluation under varying lighting and noise conditions showed an average F1-score of approximately 86%, which is notably high considering the real-time operation on a low-power embedded platform without relying on neural networks. These results establish the E-BTS as a practical, efficient, and robust solution that effectively overcomes the limitations of conventional methods for real-time drowsiness detection in embedded systems.

1. Introduction

Driver drowsiness detection is a key feature of modern Advanced Driver Assistance Systems (ADAS), playing a crucial role in enhancing road safety and reducing the risk of accidents. Non-invasive, camera-based approaches utilizing low-cost sensors have been extensively studied for drowsiness detection [1], focusing on physiological signals such as eye blink rate [2], yawning [3], and head movements [4]. Among these, eye blinks are widely regarded as the most reliable signal, making them a primary focus in numerous studies [5–9].

While extensive research has been conducted on camera-based blink detection, achieving real-time, reliable performance under varying lighting and low-power constraints remains a significant challenge. Most drowsiness detection systems depend on RGB cameras and leverage advanced neural network models, such as CNNs and RNNs [10–12]. However, these approaches face two significant challenges. First, RGB cameras are highly sensitive to lighting conditions. For instance, under

low-light scenarios such as nighttime driving, tunnels, or when the driver's face is partially obscured (e.g., by glasses), RGB-based systems struggle to reliably detect the driver's eye state, potentially leading to failures in drowsiness detection. Second, as automotive Electronic Control Units (ECUs) become increasingly high-performance, issues such as heat dissipation and power consumption have emerged as critical concerns. These challenges undermine the reliability and efficiency of systems, particularly as computationally intensive neural network models require substantial memory and processing resources, making them impractical for power-constrained ECU environments.

To address these challenges, this study focuses on a Dynamic Vision Sensor (DVS)-based Event-driven Blink Tracking System (E-BTS). Unlike traditional RGB cameras, DVS records events asynchronously based on changes in light intensity exceeding a threshold. This unique operation grants DVS several advantages, including high dynamic range,

* Corresponding author.

E-mail address: space@cau.ac.kr (W. Lee).

<https://doi.org/10.1016/j.aej.2025.07.020>

Received 13 January 2025; Received in revised form 17 June 2025; Accepted 20 July 2025

Available online 5 August 2025

1110-0168/© 2025 The Authors. Published by Elsevier B.V. on behalf of Faculty of Engineering, Alexandria University. This is an open access article under the CC BY-NC-ND license (<http://creativecommons.org/licenses/by-nc-nd/4.0/>).

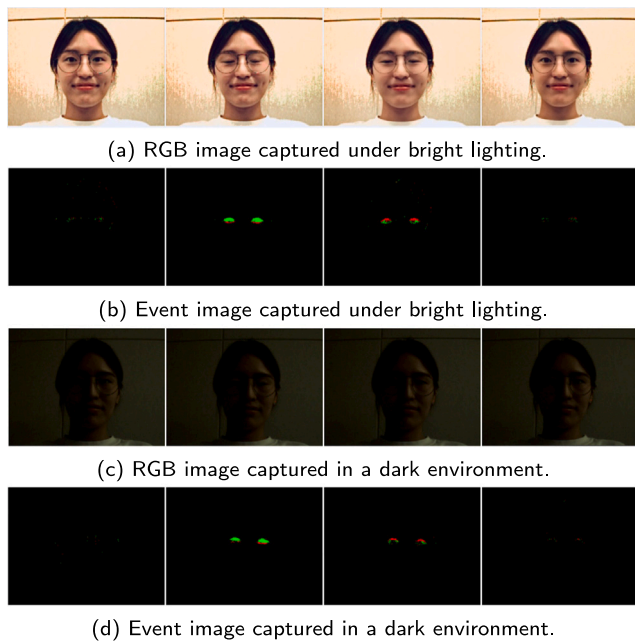


Fig. 1. Eye blink images captured using RGB and DVS cameras under bright and dark lighting conditions.

adaptive sampling rates, and low power consumption [13]. DVS operates reliably under varying lighting conditions, such as bright daylight, darkness, and tunnels, and excels in detecting the edge movement of fast-moving objects, making it particularly suitable for drowsiness detection. Fig. 1 compares blink detection data captured by an RGB camera and a DVS [14], illustrating DVS's capability to detect blinks consistently across different lighting conditions.

Leveraging DVS's strengths, recent studies have increasingly explored its application in event data analysis. However, most of these studies focus on using artificial neural networks (ANNs) to process event data without addressing the computational constraints of hardware [5,15,16]. While neural networks can effectively analyze event data and have shown strong performance across various real-life applications such as medical image analysis [17], traffic prediction [18], and object detection using mobile platforms [19], their high computational demands hinder implementation in low-power environments like E-BTS.

Indeed, as part of this research, we evaluated a representative open-source ANN accelerator — TVM's Versatile Tensor Accelerator (VTA) — which performs basic General Matrix Multiplication (GEMM) operations. Our synthesis results indicated that even this relatively simple ANN accelerator consumes more than 5× the resources of the core processor, and its substantial computational workload makes it unsuitable for real-time processing in low-power edge hardware scenarios. Notably, prior research has demonstrated that blinks can be successfully detected using the standard deviation (STD) of temporal event changes alone [7,8], highlighting the feasibility of blink detection without high-performance neural network models. Building on this insight, our study develops an E-BTS system optimized for low-power environments using a STD-based algorithm instead of relying on ANNs.

Although the STD-based algorithm requires significantly fewer computational resources than ANN-based methods, it still necessitates processing changes across all pixels. This can lead to considerable processing delays when executed on a single-core processor, complicating real-time operation and increasing power consumption. To overcome these limitations, this study introduces a dedicated hardware accelerator to enable parallel processing. By significantly reducing the computational burden of the STD-based algorithm, the accelerator facilitates real-time blink detection while maintaining low power consumption and

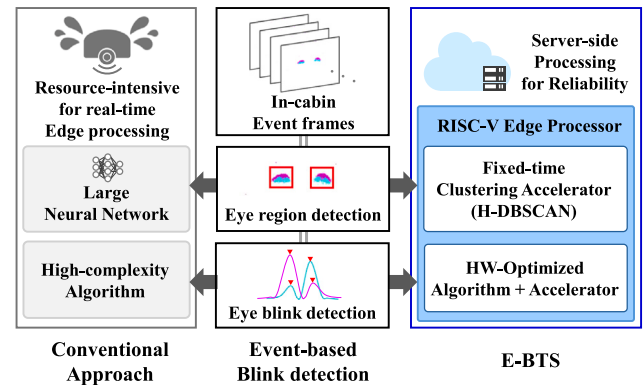


Fig. 2. Proposed E-BTS system incorporating the embedded processor and server integration.

high processing efficiency. Consequently, the E-BTS system can operate reliably in resource-constrained environments.

Another crucial challenge in blink detection is identifying the eye region. Since DVS detects motion events across the entire face, accurate blink detection requires isolating events originating from the eye region. This involves defining a bounding box (B-Box) around the eye region and applying the STD-based blink detection algorithm within this specific area. Eye region detection methods traditionally rely on pattern recognition in RGB images [20–22] or hybrid approaches combining RGB and event data with AI models [23–25]. However, these techniques often demand substantial memory resources and high computational power, making them impractical for low-power embedded systems. Recent attempts to use event-only data for eye region detection [5,8] also primarily depend on neural networks, which are similarly inefficient for embedded processors.

To address this issue, we propose a two-phase solution for eye region detection tailored to low-power embedded systems. First, under stable conditions such as highway driving where the driver's eye region position remains relatively constant, the embedded processor in E-BTS employs a non-neural algorithm, DBSCAN (Density-Based Spatial Clustering of Applications with Noise) algorithm [26], to detect the eye region in real time. Second, in scenarios involving rapid movement that challenge the embedded processor's detection capabilities, edge computing is utilized to perform AI-based eye region detection on a server, with the results returned to the processor.

Regarding the first phase, DBSCAN effectively handles noise and adapts to cluster size variations caused by changes in distance between the driver and the DVS, eliminating the need to assume a fixed eye size. However, conventional DBSCAN algorithms are still computationally intensive, making real-time processing on embedded processors difficult. Although the typical human blink rate is around 20 times per minute, each blink lasts approximately 100 ms on average and can be as brief as 30–50 ms depending on the driver's state of fatigue or health conditions. Reliable detection of these rapid blink events using DVS requires sufficient temporal resolution to clearly capture polarity transitions [27]. Nevertheless, executing conventional DBSCAN on an embedded processor operating at 50 MHz takes approximately 400–800 ms per frame, which fails to meet the required temporal resolution. To address this issue, we developed a hardware-friendly DBSCAN algorithm (*H-DBSCAN*) capable of performing eye region detection within a fixed processing time, along with dedicated hardware designed to accelerate its execution. Consequently, our proposed system generates eye region bounding boxes in just 1.26 ms per frame, comfortably achieving real-time processing at 200 FPS. This high temporal resolution ensures stable and robust blink detection, even under challenging conditions.

For the second phase, edge computing leverages server-based AI models to address scenarios where rapid movement complicates eye

Table 1
Key challenges and original contributions addressed by this work.

Challenges	Proposed solutions (Original achievements)
High computational complexity of existing RGB-based neural network methods	Hardware-friendly variance-based algorithm eliminating division and square-root operations
Difficulty of real-time blink detection due to pixel-level processing demands	Dedicated parallel-processing hardware accelerator (<i>BlinkTrack</i>) achieving real-time blink tracking at 200 FPS
Computational inefficiency of conventional DBSCAN algorithms for eye-region detection	Novel hardware-optimized DBSCAN (<i>H-DBSCAN</i>) algorithm with fixed-time complexity and hardware acceleration
Variability and unreliability under diverse lighting conditions using RGB cameras	Event-based (DVS) image processing leveraging high dynamic range for robust blink detection in varying lighting and noise conditions

Table 2
Comparison of previous E-BTS technologies and the proposed technology.

	Eye blink detection	Eye region detection	Detection range	Feasibility in low-power systems
[5]		HOG-based Random Forest model	Whole image	Not feasible
[6]		Waveclustering	Whole image	Challenging
[7]		Early temporal event model for STD-based blink detection	Whole image	Challenging
[8]	Modified [7]	GR-YOLO	ROI	Not feasible
This work	HW-optimized [7]	H-DBSCAN	ROI	Feasible

region detection on the embedded processor. Instead of transmitting resource-intensive RGB images, we minimize data transmission overhead by sending compact event images. Recent studies have demonstrated that AI models trained cross-modally on RGB and event images can accurately detect eye regions using actual event-only data [15,16]. By transmitting only event images, we significantly reduce transmission energy and latency, enabling efficient server-assisted eye region detection.

The challenges and corresponding solutions discussed above are summarized in Table 1, and Fig. 2 illustrates the proposed system, comprising DVS-based event-driven blink detection, DBSCAN-based eye region detection, and AI-supported eye region correction through edge computing. To implement this system, we developed the *BlinkTrack* hardware accelerator, integrating optimized hardware modules for the proposed blink detection and eye region detection algorithms. The accelerator, combined with a RISC-V core and other necessary IPs, forms the complete E-BTS processor. We prototyped the E-BTS processor on an FPGA and connected it to a server, completing the E-BTS system.

Our evaluation demonstrated that the proposed system operates effectively across diverse scenarios, enabling real-time blink tracking. Furthermore, synthesis of the entire processor using 45 nm technology node revealed that the dedicated E-BTS hardware contributes only 27.2% to overall processor power consumption, achieving significant energy efficiency. Specifically, the E-BTS achieved performance improvements of approximately 1275× for blink detection and 310× to 496× for eye region detection compared to existing methods. These results confirm that the proposed E-BTS system is a transformative and practical solution for low-power, real-time drowsiness detection in embedded environments.

2. Related work

Recent advancements in blink tracking systems have evolved from RGB-based approaches [10,20–22] and hybrid approaches combining RGB and event images [23–25,28] to techniques utilizing exclusively event data for blink detection.

Histogram of Oriented Gradients (HOG)-based approaches: Barua et al. [5] proposed extracting HOG features from event images for face detection using a random forest model. Although conventional in approach, this method extracts HOG features from the entire image instead of focusing on the Region of Interest (ROI) and relies heavily on

Table 3
Comparison between recent DBSCAN studies and H-DBSCAN.

	Dataset size	Execution time	Operating frequency
[30]	6.4K	81.6 μ s	100 MHz
[31]	19.5K	211.88 ms	377 MHz
[32]	768	0.8 ms	50 MHz
[33]	232	0.73 μ s	100 MHz
H-DBSCAN	90K	1.26 ms	50 MHz

neural networks, making it unsuitable for computationally constrained, low-power embedded systems and impractical for scenarios with sparse event density, such as highway driving.

Wavelet-based clustering approaches: Chen et al. [6] introduced WaveCluster [29], a grid-based clustering method utilizing wavelet transformations. Despite avoiding neural networks, this method requires computationally intensive pre-filtering and wavelet transformations over entire images, limiting its practical implementation in low-power hardware systems.

Standard deviation-based approaches: Lenz et al. [7] presented a blink detection algorithm leveraging the temporal signatures specific to event images, avoiding neural networks altogether. This method laid the groundwork for subsequent STD-based blink detection techniques [8]. However, these approaches compute blink patterns across overlapping grids over the entire image, resulting in significant computational overhead that is unsuitable for real-time embedded systems.

Hybrid GRU-YOLO approaches: While Ryan et al. [8] proposed the STD-based blink detection algorithm, they also developed a modular GR-YOLO network combining GRU and YOLO to predict bounding boxes (B-Box) for face and eye regions. Despite its modular nature and clear separation of tasks, the high computational cost remains a significant barrier for deployment on embedded processors. Nevertheless, the concept of partitioning blink tracking tasks inspired the current study.

Table 2 summarizes the distinctions between these prior works and our proposed study. Unlike previous approaches, we divide blink tracking into two distinct components — eye blink detection and eye region detection — and uniquely achieve low-power E-BTS functionality through software–hardware co-optimization for each component. This strategy significantly enhances practicality and efficiency, particularly in low-power embedded environments.

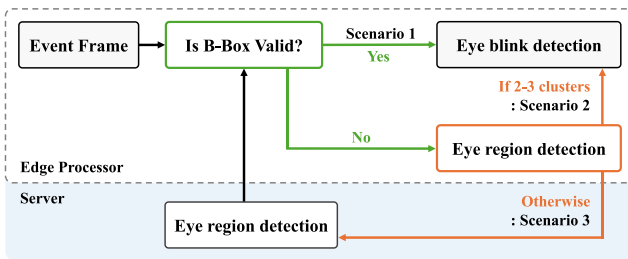


Fig. 3. Operation flowchart of the E-BTS.

DBSCAN-based clustering approaches: DBSCAN is recognized for detecting clusters of arbitrary shapes and efficiently managing noise. Table 3 reviews recent hardware implementations of DBSCAN accelerators across various domains. The traditional DBSCAN algorithm has higher computational complexity ($O(n^2)$ or $O(n \log n)$) compared to simpler grid-based clustering methods, prompting domain-specific optimizations.

Shimada et al. [30] optimized DBSCAN for grayscale images with Manhattan distance calculations. Scicluna and Bouganis [31] parallelized Manhattan distance calculations to cluster Harris Corner Detector features. Lee et al. [32] optimized DBSCAN for LiDAR sensor data by representing point clouds as fixed grids, and Borgna et al. [33] pre-sorted data along a primary axis to accelerate clustering in high-energy physics applications.

In contrast, our proposed H-DBSCAN algorithm uniquely addresses the requirements of low-power embedded systems by processing event data in fixed-size frames, providing consistent execution time, and enabling parallel clustering. Furthermore, H-DBSCAN supports adjustable neighborhood radius (ϵ), allowing flexibility beyond simple Manhattan distances. Table 3 demonstrates that H-DBSCAN achieves remarkably efficient real-time processing (1.26 ms per frame for 346×260 event images at 200 FPS) on a 50 MHz embedded processor, significantly outperforming traditional DBSCAN algorithms and ensuring suitability for resource-constrained environments.

3. E-BTS design and optimization

3.1. Operating scenarios of the proposed E-BTS

The proposed E-BTS processes event images at 200 FPS, handling event streams as frames every 5 ms in real-time. To manage computational resources efficiently, the system incorporates a frame-skipping mechanism. This mechanism evaluates each frame based on the upper and lower thresholds of the internal event count. Frames that are insufficient in information or require server-side computational support when considering the eye region from the previous frame are skipped, ensuring that only relevant frames are processed, thereby reducing unnecessary computational overhead.

Fig. 3 illustrates the operational flow of the E-BTS, which is categorized into three main scenarios. Each scenario corresponds to specific event densities or processing requirements and is explained in detail below.

- **Scenario 1:** This scenario occurs when the current eye region accurately aligns with the user's eye position, making additional eye region detection or B-Box updates unnecessary. The criterion for determining the appropriateness of the current B-Box is whether the event count within the B-Box exceeds a threshold ratio γ of the total event count in the event image frame. If the threshold is exceeded — indicating that most events in a single frame occur within the B-Box — the B-Box is considered appropriate. The system then proceeds to the *Transmit Variance* stage, as shown in Fig. 3, where blink detection is performed.

- **Scenario 2:** This scenario arises when the current B-Box no longer accurately aligns with the eye region due to the subject's movement, requiring a B-Box update. To address this, the embedded processor performs eye region detection based on the H-DBSCAN algorithm we developed. As a result, if the clusters detect the eyes (2 clusters) or both the eye and mouth (3 clusters), the process follows the same steps as in Scenario 1, namely eye blink detection.
- **Scenario 3:** This scenario occurs when excessive movement by the subject requires the B-Box's reliability to be ensured through communication with the server. If several frames are determined to have an inaccurate B-Box calculated by the embedded processor, the processor requests reliable eye detection from the server. The reliability of the processor's eye blink detection is not guaranteed until the B-Box is received from the server. During this time, all event image frames captured are temporarily stored in the memory buffer. Once the B-Box coordinates arrive from the server, the accumulated frames are processed quickly. If excessive movement occurs during communication with the server, requiring a new B-Box update, the past frames stored in the buffer are not transmitted. Instead, only the current frame, where the issue was detected, is sent to the server. This approach prevents the buffer from overflowing while storing image frames.

3.2. Proposed eye blink detection algorithm and its dedicated hardware accelerator

We have developed a methodology enabling the embedded processor to perform real-time eye region detection and eye blink detection for the proposed E-BTS scenarios 1, 2, and 3. This methodology was developed through a hardware-software co-design approach, which simultaneously considers the new algorithms for each detection task and their dedicated hardware implementation.

In particular, we have implemented this hardware as a single IP, BlinkTrack accelerator. The accelerator consists of a *Variance module*, which operates without the square root operation by using variance instead of the STD-based algorithm [7,8] for eye blink detection, an *H-DBSCAN module* that accelerates the eye region detection algorithm (H-DBSCAN), and a *Scenario Controller* that determines the scenario and selectively activates the appropriate modules. The overall architecture of the BlinkTrack accelerator is shown in Fig. 4. In the following subsections, we will provide a detailed explanation of the role of each module and the detection algorithms they implement.

3.2.1. Eye blink detection with the variance module

The Variance module is to calculate the vertical variance from a single event image frame. As shown on the left side of Fig. 4, the Variance module consists of two components: the *Thresholded Vector Generator* and the *Vertical Variance Generator*.

First, the Thresholded Vector Generator performs mean filtering and thresholding on the event image, converting it into positive and negative *thresholded maps* for variance calculation. This is a necessary step prior to eye region detection, aimed at reducing event noise in event images and clarifying cluster information. The thresholded map, generated in the form of a 32-bit thresholded vector, is stored in memory.

Typically, mean filtering involves summing all elements within the kernel and dividing by the number of elements. To reduce computational cost, we eliminated the division operation and instead designed lightweight hardware that thresholds partial sums based on pre-determined positive and negative values, which are scaled according to the fixed kernel size.

Next, the Vertical Variance Generator extracts the positive and negative vertical variance from a single frame. It reads the thresholded map stored in memory, masks the map based on the B-Box coordinates, and calculates the vertical variance in the eye region. More specifically, this hardware logic computes the variance for each column within the

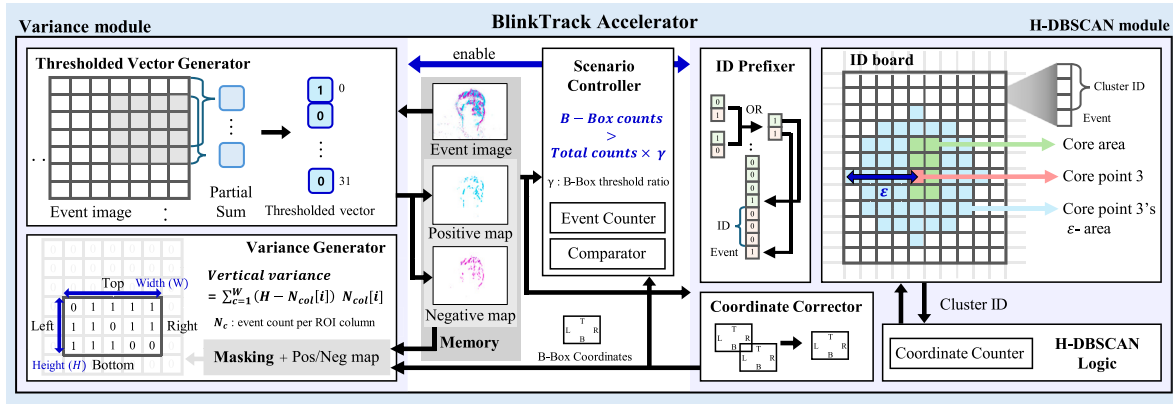


Fig. 4. Architecture of the proposed BlinkTrack accelerator, consisting of a Variance module for efficient blink detection without complex arithmetic, an H-DBSCAN module for fixed-time eye region clustering, and a Scenario Controller for managing operational scenarios in real-time.

B-Box, recording it as var_{col} , and then sums the var_{col} values across the width to produce the vertical variance var_{vert} . The i th var_{col} along the width of the B-Box can be expressed as follows:

$$var_{col}[i] = \frac{1}{h} \sum_{j=1}^h (x_j[i] - mean[i])^2, \quad (1)$$

where h represents the height of the B-Box, (i.e., the difference between the bottom and top of the B-Box), and $x_j[i]$ denotes the j th element in the i th column of the B-Box. The $mean[i]$ is the event count in the i th column ($N_{col}[i]$) divided by h , thus $mean[i] = \frac{N_{col}[i]}{h}$. By applying the fact that $x_j[i]$, as an element of the thresholded map, can only take values of 0 or 1, (1) can be simplified as follows:

$$var_{col}[i] = \frac{1}{h^3} [(h - N_{col}[i]) \cdot N_{col}[i]^2 + N_{col}[i] \cdot (h - N_{col}[i])^2].$$

Expanding and simplifying this further, it can be reduced to:

$$var_{col}[i] = \frac{1}{h^2} (h - N_{col}[i]) \cdot N_{col}[i]. \quad (2)$$

Next, by summing var_{col} across the width (w) of the B-Box, we can derive var_{vert} as follows:

$$var_{vert} = \sum_{i=1}^w (h - N_{col}[i]) \cdot N_{col}[i]. \quad (3)$$

In this process, the $\frac{1}{h^2}$ calculation from (2) is omitted, as our eye blink detection algorithm is peak-based and only requires the trend of increases and decreases across consecutive frames, rather than the absolute value of var_{vert} for each frame.

By following the steps from (1) to (3), we simplified the complex variance calculations, which involve multiple division operations, into a series of additions and multiplications. This allowed us to design lightweight hardware for processing.

3.2.2. Eye region detection with the H-DBSCAN module

The proposed eye region detection method leverages the DBSCAN clustering algorithm to group event data points around the eye into a single cohesive cluster. DBSCAN assigns cluster IDs by evaluating the density of data points within an ϵ radius. Points with at least $minpts$ neighbors are designated as core points, and all points within their ϵ radius are assigned the same cluster ID. Points that are assigned a cluster ID but are not core points are called border points, while points that are neither core points nor assigned a cluster ID are referred to as noise points. DBSCAN offers several advantages: (i) it can cluster arbitrary shapes effectively, (ii) it does not require the number of clusters or centroids to be predefined, and (iii) it can handle outliers, resulting in higher accuracy. In event images, the clusters of events around the eye region are irregular, and additional clusters can form due to movement in other areas like the mouth, making data prediction challenging. Therefore, DBSCAN is an adequate method for accurately

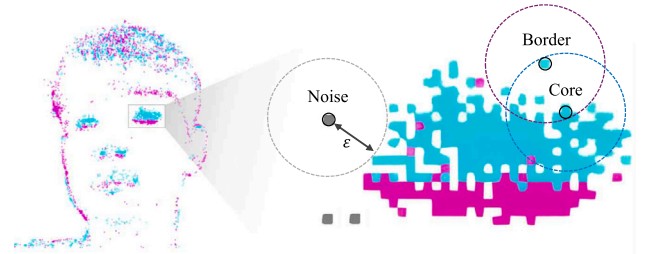


Fig. 5. Example eye region detection using the H-DBSCAN.

identifying eye positions. Fig. 5 shows an example result of applying DBSCAN to the event image, illustrating the clustering method used for the eye region.

However, as discussed in Section 2, conventional DBSCAN fundamentally requires pairwise distance calculations between each data point and its neighboring points within a given dimension (or frame), resulting in a computational complexity of $O(n^2)$. Utilizing spatial indexing structures such as KD-trees can reduce this complexity to $O(n \log n)$. When traditional DBSCAN is applied to event images, the varying number of data points per event can lead to inconsistent processing times for each frame. In the worst case, the processing time can become excessively long, making it challenging to optimize for real-time systems.

To address this issue, we propose a new H-DBSCAN algorithm along with its dedicated hardware module, referred to as the H-DBSCAN module, which is designed to perform clustering within a fixed time. As illustrated in Fig. 4, the proposed H-DBSCAN algorithm completes eye region detection by performing a single-pass windowing over the entire frame, considering local events within the core area. Consequently, unlike conventional DBSCAN, H-DBSCAN's processing time remains constant regardless of the number of events, making it highly suitable for real-time operation on low-power embedded processors. Both the hardware design and the operational details of the algorithm are described below.

As depicted on the right side of Fig. 4, the H-DBSCAN module consists of four key components: the ID Prefixer, ID Board, H-DBSCAN Logic for updating the ID Board, and a Coordinate Corrector. The ID Prefixer initializes the clustering process by preparing the thresholded map, which records the presence or absence of events as binary values (0 and 1), for further processing. Specifically, it transforms the thresholded map into the ID Board format required for the H-DBSCAN windowing operation.

As shown in Fig. 4, the thresholded map is split into positive and negative maps; however, the clustering process focuses solely on the

Algorithm 1: H-DBSCAN

```

1 H-DBSCAN:
2   if cluster detected then
3     for every element in  $\epsilon$ -area do
4       if core_id exists then
5         ID_update(core_id)
6       else if area_id exists then
7         ID_update(area_id)
8       else
9         ID_update(Cluster_Counter+1)
10        Cluster_Counter++
11 ID_update(cluster ID)
12   for every element in ID_board do
13     if ID_board.event && ID_board.id is 0 then
14       ID_board.id = cluster ID

```

presence of events, regardless of their polarity. The ID Prefixer reads both maps from memory, applies an OR operation to merge them, and assigns an *Event Bit* to indicate the presence of an event. Additionally, it prefixes each data point with a *Cluster ID Bit*, enabling the system to track cluster assignments during processing. The resulting ID Board represents each data point using an *Event Bit* and a *Cluster ID Bit*, effectively completing the preprocessing stage for clustering.

Next, the H-DBSCAN Logic, responsible for updating the ID board, is detailed in Algorithm 1. As shown in Fig. 4, the ID board includes a *Core area* for inspecting core points and an ϵ -area for evaluating their neighborhood. Similar to traditional DBSCAN, where a point with sufficient neighbors (*minpts*) is designated as a core point, in H-DBSCAN, if a point exists in the Core Area and the sum of points in the ϵ -Area exceeds *minpts*, both the core point and its neighboring points in the ϵ -area are assigned the same cluster ID.

This windowing approach has been extended to allow multiple core points to be inspected within a single window. Consequently, overlapping core points and their corresponding ϵ -areas are formed within the *Core area*, as illustrated in Fig. 4. This design aligns with a key DBSCAN principle: “If a core point exists within the epsilon region of another core point, the two clusters are merged into one”. By ensuring that the distances between core points in the *Core area* remain within ϵ , independent core points are treated as part of the same cluster, simplifying the logic and preventing the assignment of multiple IDs within a single window. Additionally, during the *ID_update* phase, if a pre-assigned cluster ID is detected within the ϵ -area, it is retained and the corresponding points are treated as part of the same cluster. This ensures consistency in cluster assignments while maintaining computational efficiency.

Finally, the Coordinate Corrector updates the B-Box coordinates as clustering progresses and operates after Algorithm 1 is complete. As the window moves without overlapping the *Core area*, the top-left region of the current window overlaps with the previous one. This indicates that the top-left region no longer needs to be updated. Based on this observation, the Coordinate Corrector monitors the top-left portion of the ID board. As mentioned, if a *Core ID* and another ID coexist within a single ϵ -area, they are treated as the same ID, and the B-Box coordinates (top, bottom, left, right boundaries) are updated to ensure precise tracking as the window progresses.

Meanwhile, the Scenario Controller, located between the Variance module and the H-DBSCAN module, plays the role of identifying Scenario 1 or 2. It does this by comparing whether the event count within the B-Box exceeds the threshold ratio γ of the total event count in the event image frame, based on the thresholded map output from the Variance module, and then sends an enabling signal to the H-DBSCAN module.

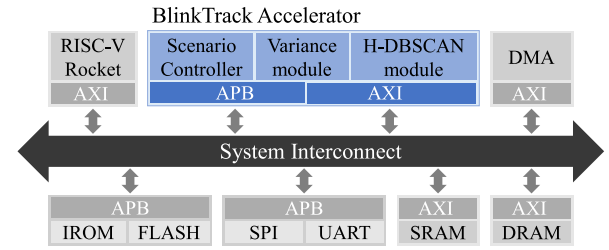


Fig. 6. Architecture of the proposed E-BTS processor, integrating the BlinkTrack accelerator, RISC-V Rocket core, memory, and external I/O interfaces, interconnected through μ NoC.

Table 4

FPGA resource consumption of the proposed processor.

		LUT	FF	BRAM
Processor		81 066	50 989	220
BlinkTrack accelerator	Variance	33 151	9598	8
	H-DBSCAN	14 816	5367	40
	Scenario controller	179	563	–
Proportion (%)		59.4	30.5	21.8

4. Implementation

4.1. Design and development of the E-BTS processor with the BlinkTrack accelerator

We designed an application-specific processor equipped with the BlinkTrack accelerator. Before integrating the BlinkTrack accelerator, we first designed a baseline processor using RISC-V eXpress (RVX), an EDA tool widely used for developing RISC-V-based processors [34–37]. This baseline processor operates at a 50 MHz clock frequency, utilizing the RISC-V Rocket [38] as the core. The system is equipped with 64 KB SRAM as the main memory, 128 MB DRAM for storing image data, Flash as non-volatile memory, IROM for the booting process, and standard external I/O interfaces such as UART and SPI. In Scenario 3, SPI is utilized for communication with the server via Bluetooth (BT 2.1). The system interconnect is based on μ NoC [39], a Network-on-Chip designed for embedded processors. We then integrated the BlinkTrack accelerator into the completed baseline processor using AXI and APB interfaces. Since BlinkTrack is implemented as an independent IP using Verilog with standard AXI and APB interfaces, it can be seamlessly integrated into various low-power hardware platforms such as ASICs, FPGAs, and SoCs, ensuring high scalability and portability across different embedded environments. Additionally, we added a DMA module to the accelerator to allow direct access to DVS event images from memory. Finally, Fig. 6 illustrates the architecture of the designed processor.

Both the baseline processor and the proposed processor with the BlinkTrack accelerator were prototyped using the Xilinx Kintex7 Genesys2 FPGA board. Table 4 reports the resource consumption from synthesizing the proposed processor on FPGA. The entire processor utilized 81,066 LUTs, 50,989 FFs, and 220 BRAMs, with the BlinkTrack accelerator consuming 59.4% of the LUTs and 30.5% of the FFs. The baseline processor was reported to consume 32,106 LUTs, 34,560 FFs, and 172 BRAMs.

Fig. 7 shows the demonstration of our implemented E-BTS. The FPGA prototyped processor is used for edge device, while a laptop serves as the server. The module used for BT communication between the prototype processor and the laptop server is the Pmod BT2 from Digilent [40]. The DVS employed is the DAVIS 346 [14], which requires a USB connection to the processor. However, since our prototype processor lacks a USB IP, we implemented a temporary solution where the images were first received by a PC and then transmitted to the

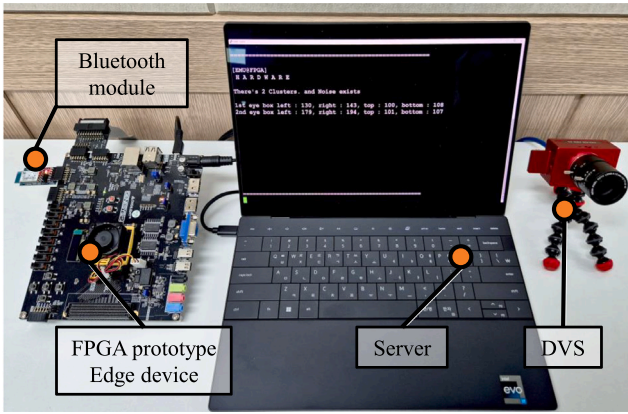


Fig. 7. Experimental setup of the E-BTS, featuring an FPGA-prototyped embedded processor (Genesys2 board) as the edge device and a laptop server connected via BT2.1.

Algorithm 2: Blink detection

Input: $right_p, left_p, right_n, left_n$
Output: blink

Parameters: α : overlap threshold, β : synchronization threshold

- 1 $Dur(p) := p.t_{end} - p.t_{start}$
- 2 $Overlap(p_1, p_2) := \min(p_1.t_{end}, p_2.t_{end}) - \max(p_1.t_{start}, p_2.t_{start})$
- 3 **Function** $Half_Blink(var_{major}, var_{minor})$:
- 4 $(major, minor) = Peak_Detection(var_{major}, var_{minor})$
- 5 **if** $major.apex > minor.apex$ **and** $Overlap(major, minor) > \alpha$
- 6 **if** $Dur(minor) > Dur(major)$ **then**
- 7 **return** $major$
- 8 **Function** $Sync(p_1, p_2)$:
- 9 **return** $Overlap(p_1, p_2) > \beta \cdot \min(Dur(p_1), Dur(p_2))$
- 10 **Function** $Blink_Detection()$:
- 11 $Close_R \leftarrow Half_Blink(right_n, right_p)$
- 12 $Open_R \leftarrow Half_Blink(right_p, right_n)$
- 13 $Close_L \leftarrow Half_Blink(left_n, left_p)$
- 14 $Open_L \leftarrow Half_Blink(left_p, left_n)$
- 15 **if** $Sync(Close_R, Close_L)$ **and** $Sync(Open_R, Open_L)$ **then**
- 16 **return** blink detected

embedded processor. The delay and power consumption resulting from this workaround were excluded from the experimental measurements (see Fig. 8).

4.2. Development of a blink detection application

We developed a blink detection application based on event images, leveraging real-time vertical variance values extracted by the BlinkTrack module. Unlike traditional RGB-camera-based methods [41] relying on eyelid-distance measurements, our proposed algorithm utilizes distinct temporal variance patterns observable specifically in DVS event data. During eye blinks, DVS event data exhibits clear and distinct variance patterns: negative-event variance predominantly appears during the eye-closing phase, while positive-event variance is dominant during the eye-opening phase, forming characteristic peaks for each phase, as seen in Fig. 8.

Algorithm 2 presents the pseudo-code of the proposed algorithm. The algorithm initially extracts the closing and opening phases independently using the $Half_Blink$ function. This function receives two variance sequences separated by event polarity and identifies major and minor peaks from each sequence, characterized by their apex, start time (t_{start}), and end time (t_{end}). Peak detection follows the method introduced in previous research [41]. Subsequently, the $Half_Blink$ function evaluates the temporal overlap between identified major and

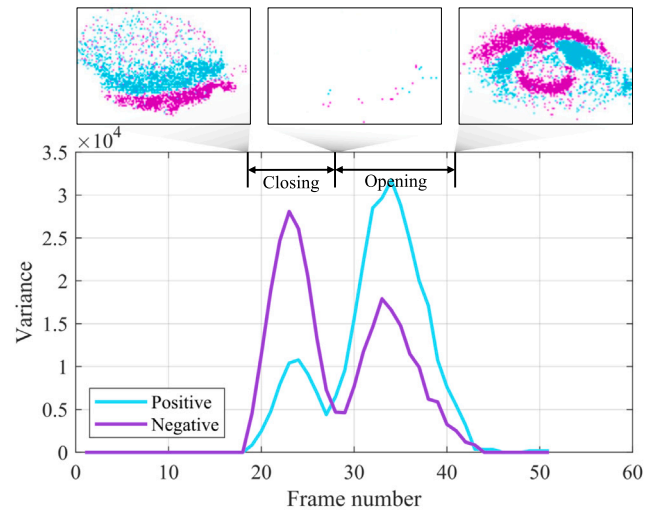


Fig. 8. Vertical variance graph extracted during the eye-blink, forming the basis for Algorithm 2.

minor peaks using the $Overlap$ function. If the apex of the major peak is greater than that of the minor peak, and the temporal overlap between these peaks ($Overlap(major, minor)$) exceeds the overlap threshold (α) relative to the duration of the minor peak ($Dur(minor)$), the event is confirmed as a valid closing or opening phase (cf. Line 6 – Line 7).

These closing and opening phases are independently detected for each eye, and subsequently, the $Sync$ function verifies temporal synchronization between detected half-blinks from both eyes. Specifically, if the temporal overlap between detected half-blinks exceeds the synchronization threshold (β), the event is recognized as a complete blink (cf. Line 9).

Threshold values (α, β) in the $Overlap$ and $Sync$ functions are typically set empirically based on experimental data, as no universal optimal thresholds exist. Through comprehensive experiments evaluating various threshold combinations, we found that higher threshold values significantly increased missed blink detections, drastically reducing overall detection performance. Conversely, thresholds around 0.1 consistently ensured robust performance without substantially impacting computational complexity or the algorithm's structural efficiency. Consequently, in our implementation, these thresholds are parameterized for application-specific flexibility, with default values set at 0.1 to optimally balance detection performance and computational efficiency. As detailed later in Section 5.2, our chosen default thresholds consistently achieved high detection reliability across diverse test conditions (F1-score around 86%).

5. Evaluation and verification

Through the previous sections, we demonstrated BlinkTrack's lightweight design and its ability to enable real-time processing for eye-blink and eye-region detections in the E-BTS. In this section, we report additional experimental results to further validate the performance of our proposed technology, particularly highlighting the advantages of the H-DBSCAN, the effectiveness of BlinkTrack across all operational scenarios of E-BTS, and the low power consumption of the BlinkTrack.

5.1. Real-time performance of the developed E-BTS processor

The E-BTS processor targets driver drowsiness detection with a goal of achieving 200 FPS, ensuring accurate eye blink detection even in environments with rapid motion. In other words, the developed embedded processor must either extract the vertical variance in E-BTS

Table 5

Execution time and speedup for E-BTS across three scenarios on baseline and proposed processors. *Note:* The baseline processor denotes the same 50 MHz RISC-V Rocket core executing equivalent blink detection and eye region detection algorithms without hardware acceleration or software optimizations.

	Scenario 1	Scenario 2	Scenario 3
Baseline (ms)	357.04	478.59–763.46	–
Proposed (ms)	0.28	1.54	148.84
Speedup	1275×	310–496×	–

Scenario 1 or complete the B-Box coordinate correction in Scenario 2 within 5 ms per frame.

We measured the performance for these scenarios on the FPGA prototype of the processor, and as reported in Table 5, the developed processor successfully met the target of 5 ms per frame. Specifically, the processor achieved vertical variance computation with an average execution time of 0.28 ms. This achievement is due to the support provided by the Variance module within the BlinkTrack accelerator, which simplifies computation by using only subtraction and multiplication with the B-Box height as a parameter, ensuring consistent execution time regardless of the number of events. As a result, the execution time remains stable across various cases with minimal deviation. Additionally, as reported in the table, executing the same task on the baseline processor without the BlinkTrack accelerator takes 357.04 ms, demonstrating that our proposed technology achieves an approximate 1275× performance improvement.

In Scenario 2, the Scenario Controller determines whether the current B-Box is inaccurate based on the output of the Thresholded Vector Generator and, if necessary, sends an activation signal to the H-DBSCAN module to update the B-Box. This process takes an average of 1.54 ms with minimal deviation. Unlike the execution time on the baseline processor (478.59–763.46 ms), the H-DBSCAN algorithm achieves stable processing time by relying solely on image size rather than the number of events in the image. This key characteristic distinguishes the H-DBSCAN algorithm from standard DBSCAN, resulting in up to approximately 496× speedup.

Scenario 3 of the E-BTS occurs when the embedded processor cannot autonomously correct the B-Box and must request assistance from the server. This scenario involves five key components: SPI communication time within the processor, the time to transmit the image data to the server via the BT 2.1 protocol, server processing time, the time to receive updated B-Box coordinates, and the final SPI communication time. Transmitting the 346×260 event image via SPI took 10.77 ms, while transmission via BT 2.1 required 128 ms. The server processed the event image and generated updated B-Box coordinates in 10 ms, and receiving two sets of four B-Box coordinates took 0.06 ms via BT 2.1 and 0.005 ms via SPI, resulting in a total execution time of 148.84 ms.

This duration roughly corresponds to the time for a single eye blink, meaning that real-time processing for that blink might be momentarily delayed. However, no data is lost, as approximately 30 frames are buffered during this time and quickly processed at 0.28 ms per frame once the updated B-Box is received. If a new B-Box update request to the server arises while processing buffered images, the system must decide between prioritizing real-time processing or risking data loss. In such cases, the system prioritizes real-time processing by transmitting the most recent image from the DVS for B-Box updates instead of relying on previously buffered images.

5.2. Quantitative evaluation of blink detection

In this subsection, we experimentally validate that the proposed E-BTS operates robustly under diverse driving conditions. Additionally, we quantitatively analyze how the proposed variance-based algorithm,

Table 6

Evaluation of E-BTS performance across four edge scenarios: Bright-Calm (B/C), Bright-Windy (B/W), Dark-Calm (D/C), and Dark-Windy (D/W).

Metric	(1) B/C	(2) B/W	(3) D/C	(4) D/W
Time (s)	86.2	67.8	80.5	50.3
Blinks	31	30	31	30
TP	26	27	29	25
FP	2	10	4	3
FN	5	3	2	5
Precision	83.9%	73.0%	87.9%	89.3%
Recall	92.9%	90.0%	93.5%	83.3%
F1	88.1%	80.6%	90.6%	86.2%

which is a hardware-friendly simplification of the conventional STD-based algorithm, enhances computational efficiency without incurring significant performance degradation. For the evaluation, the overlap and synchronization thresholds used in the Half_Blink and Sync functions were set to 0.1, chosen as the minimal threshold values capable of maintaining stable detection performance based on empirical analyses of various thresholds. The functional performance of E-BTS was assessed using the Precision, Recall, and F1-score metrics, defined as follows:

$$\text{Precision} = \frac{TP}{TP + FP}, \quad \text{Recall} = \frac{TP}{TP + FN},$$

$$\text{F1-score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}},$$

where TP (True Positive) denotes correctly detected blinks, FP (False Positive) represents non-blink events falsely identified as blinks, and FN (False Negative) indicates missed blink detections.

The dataset used in our evaluation was collected directly by our research team using a DAVIS346 event camera. Although this dataset is not publicly available, it was carefully designed to validate the real-time feasibility of the proposed system under diverse environmental conditions, combining varying lighting (bright/dark) and different background noise levels (calm/windy). Table 6 summarizes the evaluation results of Precision, Recall, and F1-score based on manually annotated blink data from this collected dataset. Our experiments confirm the high dynamic range capability of the DVS sensor, evidenced by negligible performance differences between bright and dark conditions. Moreover, the E-BTS achieved an F1-score exceeding 80% across all scenarios, averaging approximately 86%. These results demonstrate the feasibility of implementing real-time, hardware-based DVS blink detection, an achievement not previously addressed by conventional software-focused methods [42,43].

Furthermore, Table 7 presents a comparative analysis of F1-scores between the conventional STD-based method [7,8] and the proposed variance-based method optimized for hardware implementation. As elaborated in Section 3.2.1, the variance-based calculation substantially reduces computational complexity by eliminating square-root and division operations. This simplification is justified by the fact that STD is defined as the square root of variance, and square-root is a monotonic increasing function. Since our blink detection algorithm relies on detecting temporal peak patterns, removing the square-root operation only affects the scale of the values, not the relative patterns or peak positions themselves. Therefore, the core criteria for blink detection remain unaffected.

As a result, across the four dataset conditions, the variance-based and STD-based features yielded average F1-scores of 86.4% and 86.9%, respectively, with a marginal difference of less than 0.6%. Despite this minimal performance variation, the proposed variance-based approach significantly reduces hardware resource requirements by eliminating computationally intensive operations, specifically division and square-root calculations, thereby offering clear advantages for real-time implementation in edge computing environments.

Table 7

Comparison of average F1-scores between the proposed variance-based method and the conventional STD-based method under the same four edge scenarios as described in Table 6.

Feature	(1)	(2)	(3)	(4)	Avg
Variance	88.1%	80.6%	90.6%	86.2%	86.4%
STD	90.0%	80.6%	90.9%	86.3%	86.9%

Table 8

Power breakdown of the developed E-BTS processor.

	Module	Power (mW)
BlinkTrack accelerator	Variance	9.34
	H-DBSCAN	4.9
	Scenario controller	0.46
Processor components	Rocket	8.96
	System interconnect	15.39
	External peri.	4.12
	Normal peri.	9.67
	Special IP peri.	1.16

5.3. Performance and efficiency analysis with state-of-the-art DVS-based systems

To assess the low power consumption of the proposed BlinkTrack accelerator, we synthesized the embedded processor using a 45 nm PDK [44] with Synopsys Design Compiler [45]. As shown in Table 8, the power consumption of the main processor components was evaluated. The BlinkTrack accelerator accounts for approximately 27.2% of the total processor power consumption, which is only about 1.77 times greater than the power consumed by the Rocket core. This 27.2% power overhead is justified by the substantial performance improvements, including a remarkable 1275× speedup in blink detection and a 310–496× speedup in eye region detection, far exceeding what could be achieved by adding processor cores with equivalent power consumption. Consequently, our embedded processor is the first demonstrated hardware capable of real-time, event-based eye blink detection, effectively meeting the primary objectives of this work.

Recent advances in event-based vision sensors (DVS) have led to the emergence of various computational platforms, including specialized neuromorphic processors tailored for event-driven applications. These platforms typically aim to efficiently process event data in real-time applications such as AR/VR and wearable devices, often assuming stable or predefined ROI. To contextualize and highlight the unique contributions and advantages of our proposed BlinkTrack accelerator within the broader landscape of these state-of-the-art event-based platforms, we provide a detailed comparative analysis in this section.

For instance, Retina, proposed by Bonazzi et al. [46], employs a Spiking Neural Network (SNN) to regress pupil bounding-box coordinates. Stoffregen et al.'s Coded Differential Lighting (CDL) [47] utilizes dual LEDs to generate events through eye glints for 3D gaze estimation. Additionally, Zhang et al. introduced the Sub-manifold SCNN system [48], combining FPGA-accelerated SCNNs and CPU-based GRUs specifically tailored for wearable and AR scenarios.

However, the automotive in-cabin environment targeted by our study presents significantly different challenges due to dynamic and unpredictable ROI movements arising from driver head rotations, posture changes, and subtle movements. Unlike wearable-based neuromorphic systems that assume continuous event flows within a stable ROI, the in-cabin scenario involves sparse, temporally scattered events, with ROI locations frequently shifting or temporarily leaving the sensor's field of view. To address these unique challenges, our proposed E-BTS system integrates dedicated hardware-friendly algorithms for dynamic ROI tracking and efficient blink detection into a low-power embedded processor with custom hardware accelerators. Despite handling a broader and more unstable detection area, our system achieves real-time processing with latencies of 0.3–1.5 ms and power consumption of only 15.9 mW.

In comparison, Retina operates at a lower resolution (64×64 pixels) compared to our 346×260 pixels, requires periodic neuron resets (potentially causing discontinuous tracking), and achieves latencies of 5.6–8 ms with power consumption of 2.9–4.8 mW. CDL achieves approximately 1 ms latency via dual LED-induced glints but consumes around 35 mW, including additional overhead for LED integration. The Sub-manifold SCNN system achieves approximately 1 ms inference latency; however, high energy consumption per inference (~ 2 mJ) results in watt-level average power consumption at continuous 1 kHz operation.

In summary, the proposed E-BTS system uniquely implements a low-power, real-time blink detection hardware solution optimized specifically for automotive in-cabin scenarios without relying on wearable devices or additional hardware such as LEDs. By actively tracking dynamic eye regions using a single DVS sensor and performing computationally efficient variance-based detection, our system effectively balances robustness, latency, and power efficiency. This holistic hardware-algorithm co-design represents the first practical demonstration of real-time edge-level blink detection capability suitable for dynamically changing in-cabin environments, clearly distinguishing itself from prior state-of-the-art event-based platforms.

6. Discussion

The proposed E-BTS demonstrates robust performance and efficient computation tailored for real-time drowsiness detection in embedded automotive environments. However, several practical considerations must be addressed to enable large-scale deployment and ensure broader applicability in real-world scenarios.

First, while this study did not explicitly address security issues, securing data communication in edge–cloud architectures is critical for ensuring system integrity, particularly given the privacy-sensitive nature of driver monitoring data. Future implementations of E-BTS — or similar embedded vision systems — could incorporate lightweight symmetric encryption schemes, such as AES [49], combined with efficient elliptic curve cryptography (ECC) key-exchange protocols [50]. Such hybrid AES-ECC encryption methods have been shown to be feasible for resource-constrained embedded systems [35] and could effectively mitigate risks related to metadata leakage and data tampering. Proactively addressing these security concerns would significantly improve the robustness, privacy assurance, and trustworthiness of embedded driver-monitoring systems in practice.

Second, the communication protocol used in our prototype (BT 2.1) may not sufficiently satisfy the latency and reliability requirements in hybrid edge–cloud automotive environments — particularly under scenarios involving multiple vehicles or complex wireless topologies. Future extensions of this work should explore more recent low-power communication protocols, such as BLE 5.0 or ZigBee for short-range, low-latency communication, and LoRa for low-power long-range transmission. Detailed empirical studies examining latency, throughput, and packet loss under simultaneous multi-vehicle connections will inform the development of lightweight Quality of Service (QoS) strategies. Such adaptive approaches — potentially incorporating dynamic adjustment of transmission intervals or compression ratios — will be essential for maintaining consistent performance under variable network conditions and operational constraints.

Finally, given the stringent power constraints typically encountered in automotive embedded systems, comprehensive dynamic power management techniques should be considered. Future researchers may explore fine-grained clock gating methods, driven by scenario controllers, to activate computation modules such as H-DBSCAN or Variance blocks strictly on demand. Real-time monitoring of input throughput, when combined with Dynamic Voltage and Frequency Scaling (DVFS) [51], can further optimize the energy-performance trade-off. Leveraging vehicle-grade Power Management ICs (PMICs) [52] may also enable more advanced strategies — such as isolating accelerator IP

cores into independent power domains (i.e., power islands with precise V_{dd} control) — to maximize energy efficiency during system operation. These advanced power techniques are broadly applicable not only to the proposed E-BTS but also to other embedded automotive vision systems seeking aggressive reductions in energy consumption.

In summary, addressing these considerations — security, communication robustness, and dynamic power management — will be crucial for translating research-level innovations into practical and deployable automotive systems. The methodologies and directions outlined here provide a roadmap for extending this work beyond the specific context of blink detection, enabling broader adoption of event-driven, low-power embedded vision technologies.

7. Conclusion

This paper presented the Event-Driven Blink Tracking System (E-BTS), which leverages event-based images captured by a Dynamic Vision Sensor (DVS) to address the limitations of conventional RGB-based, neural-network-driven drowsiness detection methods. By overcoming sensitivity to lighting conditions and reducing computational complexity, E-BTS enables practical deployment in real-time, low-power embedded environments. The system comprises an embedded processor responsible for detecting eye regions and tracking blinks in real time, alongside a server that provides AI-based assistance under complex scenarios. A key contribution of this work is the design and implementation of a hardware-software co-optimized processor featuring the *BlinkTrack* accelerator. This specialized hardware module executes efficient algorithms for both blink detection and eye region detection, leveraging parallel processing to achieve high performance while minimizing power consumption. The E-BTS processor was prototyped on an FPGA platform and validated through extensive experiments. Evaluation results demonstrated stable and real-time performance across diverse lighting and motion conditions. Specifically, the proposed system achieved a 1275× speedup in blink detection and a 310–496× speedup in eye region detection, while the dedicated accelerator consumed only 27.2% of the processor's total power. Furthermore, quantitative tests under various lighting and noise conditions showed an average F1-score of approximately 86%, confirming the robustness and reliability of the proposed approach. Overall, the E-BTS system establishes a scalable, efficient, and reliable solution for real-time drowsiness detection in embedded automotive systems and similar power-constrained environments.

CRedit authorship contribution statement

Jongin Choi: Writing – original draft, Validation, Software, Methodology, Conceptualization. **Eunjin Choi:** Writing – original draft, Visualization, Validation, Software, Formal analysis. **Seonghyun Choi:** Writing – original draft, Visualization, Validation. **Woojoo Lee:** Writing – review & editing, Writing – original draft, Supervision, Funding acquisition, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported in part by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. RS-2023-00277060, Development of open edge AI SoC hardware and software platform), in part by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. RS-2024-00345668), and in part by the Chung-Ang University Graduate Research Scholarship Grants in 2024.

References

- [1] A.A. Aljohani, Real-time driver distraction recognition: A hybrid genetic deep network based approach, *Alex. Eng. J.* 66 (2023) 377–389.
- [2] V. Skaramagkas, G. Giannakakis, E. Ktistakis, D. Manousos, I. Karatzanis, N.S. Tachos, E. Tripoliti, K. Marias, D.I. Fotiadis, M. Tsiknakis, Review of eye tracking metrics involved in emotional and cognitive processes, *IEEE Rev. Biomed. Eng.* 16 (2023) 260–277.
- [3] S. Kaplan, M.A. Guvensan, A.G. Yavuz, Y. Karalurt, Driver behavior analysis for safe driving: A survey, *IEEE Trans. Intell. Transp. Syst.* 16 (6) (2015) 3017–3032.
- [4] A. Tawari, S. Martin, M.M. Trivedi, Continuous head movement estimator for driver assistance: Issues, algorithms, and On-Road evaluations, *IEEE Trans. Intell. Transp. Syst.* 15 (2) (2014) 818–830.
- [5] S. Barua, Y. Miyatani, A. Veeraraghavan, Direct face detection and video reconstruction from event cameras, in: 2016 IEEE Winter Conference on Applications of Computer Vision, WACV, 2016, pp. 1–9.
- [6] G. Chen, L. Hong, J. Dong, P. Liu, J. Conradt, A. Knoll, EDD: Event-Based drowsiness driving detection through facial motion analysis with neuromorphic vision sensor, *IEEE Sens. J.* 20 (11) (2020) 6170–6181.
- [7] G. Lenz, S.-H. Ieng, R. Benosman, Event-Based face detection and tracking using the dynamics of Eye Blinks, *Front. Neurosci.* 14 (2020).
- [8] C. Ryan, B. O'Sullivan, A. Elrasad, A. Cahill, J. Lemley, P. Kieley, C. Posch, E. Perot, Real-time face & eye tracking and blink detection using event cameras, *Neural Netw.* 141 (2021) 87–97.
- [9] F.M. Nazarov, M.M. Xamidov, Eye state classification method for detecting physiological deviations in drivers based on CNN algorithm, in: International Russian Automation Conference (RusAutoCon), 2024, pp. 802–807.
- [10] S. Rajamohana, E. Radhika, S. Priya, S. Sangeetha, Driver drowsiness detection system using hybrid approach of convolutional neural network and bidirectional long short term memory (CNN_BILSTM), *Mater. Today: Proc.* 45 (2021) 2897–2901, International Conference on Advances in Materials Research.
- [11] S. Hong, Y. Kim, T. Park, “Blinks in the dark”: Blink estimation with domain adversarial training (BEAT) network, *IEEE Trans. Consum. Electron.* 69 (3) (2023) 581–593.
- [12] T.R. Mahesh, M.S. Ram, N.S.S. Ram, A. Gowtham, T.V.N. Swamy, Real-Time eye blinking for password authentication, in: F.P. García Márquez (Ed.), International Conference on Intelligent Emerging Methods of Artificial Intelligence & Cloud Computing, Springer International Publishing, Cham, 2022, pp. 428–434.
- [13] G. Gallego, T. Delbrück, G. Orchard, C. Bartolozzi, B. Taba, A. Censi, S. Leutenegger, A.J. Davison, J. Conradt, K. Daniilidis, D. Scaramuzza, Event-Based vision: A survey, *IEEE Trans. Pattern Anal. Mach. Intell.* 44 (1) (2022) 154–180.
- [14] inivation, DAVIS346, 2025, <https://inivation.com/wp-content/uploads/2019/08/DAVIS346.pdf> (Accessed 12 January 2025).
- [15] D. Kang, L. Ma, Real-time eye tracking for bare and sunglasses-wearing faces for augmented reality 3D Head-Up displays, *IEEE Access* 9 (2021) 125508–125522.
- [16] D. Kang, D. Kang, Event Camera-Based pupil localization: Facilitating training with Event-Style translation of RGB faces, *IEEE Access* 11 (2023) 142304–142316.
- [17] W.H. Bangyal, R. Shakir, A. Ashraf, Z.U. Qayyum, N.U. Rehman, Automatic detection of diabetic retinopathy from fundus images using machine learning based approaches, in: International Multitopic Conference, INMIC, 2023, pp. 1–6.
- [18] M.A. Amin, J.-U.-R. Chughtai, W. Ahmad, W.H. Bangyal, I. Ul Haq, Trajectory data mining and trip travel time prediction on specific roads, in: International Conference on Engineering & Computing Technologies, ICECT, 2024, pp. 1–8.
- [19] M. Khalid, A. Ashraf, W.H. Bangyal, M. Iqbal, An android application for unwanted vehicle detection and counting, in: International Conference on Human-Centered Cognitive Systems, HCCS, 2023, pp. 1–6.
- [20] P. Viola, M. Jones, Rapid object detection using a boosted cascade of simple features, in: Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001, Vol. 1, 2001, 1–1.
- [21] N. Dalal, B. Triggs, Histograms of oriented gradients for human detection, in: 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR'05, Vol. 1, 2005, pp. 886–893, vol. 1.
- [22] R. Ranjan, V.M. Patel, R. Chellappa, A deep pyramid deformable part model for face detection, in: 2015 IEEE 7th International Conference on Biometrics Theory, Applications and Systems, BTAS, 2015, pp. 1–8.
- [23] A.N. Angelopoulos, J.N. Martel, A.P. Kohli, J. Conradt, G. Wetzstein, Event-based near-eye gaze tracking beyond 10,000 Hz, *IEEE Trans. Vis. Comput. Graphics* 27 (5) (2021) 2577–2586.
- [24] Y. Feng, N. Goulding-Hotta, A. Khan, H. Reyserhove, Y. Zhu, Real-Time gaze tracking with Event-Driven eye segmentation, in: IEEE Conference on Virtual Reality and 3D User Interfaces, VR, 2022, pp. 399–408.
- [25] G. Zhao, Y. Yang, J. Liu, N. Chen, Y. Shen, H. Wen, G. Lan, EV-Eye: rethinking high-frequency eye tracking through the lenses of event cameras, in: Proceedings of the 37th International Conference on Neural Information Processing Systems, NIPS '23, Curran Associates Inc., Red Hook, NY, USA, 2024.
- [26] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, et al., A density-based algorithm for discovering clusters in large spatial databases with noise, in: Kdd, Vol. 96, 1996, pp. 226–231.

- [27] A. Picot, A. Caplier, S. Charbonnier, Comparison between EOG and high frame rate camera for drowsiness detection, in: Workshop on Applications of Computer Vision, WACV, 2009, pp. 1–6.
- [28] K. Iddrisu, W. Shariff, P. Corcoran, N.E. O'Connor, J. Lemley, S. Little, Event Camera-Based eye motion analysis: A survey, *IEEE Access* 12 (2024) 136783–136804.
- [29] L. Chen, T. Yu, R. Chirkova, WaveCluster with differential privacy, in: Proceedings of the 24th ACM International on Conference on Information and Knowledge Management, CIKM '15, Association for Computing Machinery, New York, NY, USA, 2015, pp. 1011–1020.
- [30] A. Shimada, H. Zhu, T. Shibata, A VLSI DBSCAN processor composed as an array of micro agents having self-growing interconnects, in: 2013 IEEE International Symposium on Circuits and Systems, ISCAS, 2013, pp. 2062–2065.
- [31] N. Scicluna, C.S. Bouganis, ARC 2014: A multidimensional FPGA-Based parallel DBSCAN architecture, 9, (1) Association for Computing Machinery, New York, NY, USA, 2015,
- [32] S. Lee, S. An, J. Kim, H. Namkung, J. Park, R. Kim, S.E. Lee, Grid-Based DBSCAN clustering accelerator for LiDAR's point cloud, *Electron.* 13 (17) (2024).
- [33] L. Borgna, M. Barbone, J. Cao, A. Rose, A. Tapper, R. Bainbridge, W. Luk, Accelerating the DBSCAN clustering algorithm for low-latency primary vertex reconstruction at the HL-LHC, in: Proceedings of the 20th International Workshop on Advanced Computing and Analysis Techniques in Physics Research (ACAT 2022), Villa Romanazzi Carducci, Bari, Italy, 2022.
- [34] K. Han, S. Lee, K.-I. Oh, Y. Bae, H. Jang, J.-J. Lee, W. Lee, M. Pedram, Developing TEI-Aware Ultralow-Power SoC platforms for IoT end nodes, *IEEE Internet Things J.* 8 (6) (2021) 4642–4656.
- [35] E. Choi, J. Park, K. Han, W. Lee, AESware: Developing AES-enabled low-power multicore processors leveraging open RISC-V cores with a shared lightweight AES accelerator, *Eng. Sci. Technol. Int. J.* 60 (2024) 101894.
- [36] J. Park, K. Han, E. Choi, J.-J. Lee, K. Lee, W. Lee, M. Pedram, Designing Low-Power RISC-V multicore processors with a shared lightweight floating point unit for IoT endnodes, *IEEE Trans. Circuits Syst. I. Regul. Pap.* 71 (9) (2024) 4106–4119.
- [37] K. Lee, S. Jeon, K. Lee, W. Lee, M. Pedram, Radar-PIM: Developing IoT processors utilizing Processing-in-Memory architecture for Ultrawideband-Radar-Based respiration detection, *IEEE Internet Things J.* 12 (1) (2025) 515–530.
- [38] Rocket, Rocket-risc-v, 2025, <https://github.com/chipsalliance/rocket-chip>. (Accessed 12 January 2025).
- [39] K. Han, S. Lee, J.-J. Lee, W. Lee, M. Pedram, TIP: A temperature effect Inversion-Aware Ultra-Low power System-on-Chip platform, in: 2019 IEEE/ACM International Symposium on Low Power Electronics and Design, ISLPED, 2019, pp. 1–6.
- [40] Digilent, PmodBT2, 2025, <https://digilent.com/shop/pmod-bt2-bluetooth-interface/>. (Accessed 12 January 2025).
- [41] M.H. Baccour, F. Driewer, E. Kasneci, W. Rosenstiel, Camera-Based eye blink detection algorithm for assessing driver drowsiness, in: 2019 IEEE Intelligent Vehicles Symposium, IV, 2019, pp. 987–993.
- [42] R. Daza, D. DeAlcala, A. Morales, R. Tolosana, R. Cobos, J. Fierrez, ALEBk: Feasibility study of attention level estimation via blink detection applied to e-learning, 2021, [arXiv:2112.09165](https://arxiv.org/abs/2112.09165) URL <https://arxiv.org/abs/2112.09165>.
- [43] W. Zeng, Y. Xiao, S. Wei, J. Gan, X. Zhang, Z. Cao, Z. Fang, J.T. Zhou, Real-Time Multi-Person eyeblink detection in the wild for untrimmed video, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR, 2023, pp. 13854–13863.
- [44] NCSU, FreePDK45, 2025, <https://eda.ncsu.edu/freepdk/freepdk45>. (Accessed 12 January 2025).
- [45] Synopsys, Design compiler, 2025, <https://www.synopsys.com/implementation-and-signoff/rtl-synthesis-test/dc-ultra.html>. (Accessed 12 January 2025).
- [46] P. Bonazzi, S. Bian, G. Lippolis, Y. Li, S. Sheik, M. Magno, Retina : Low-power eye tracking with event camera and spiking hardware, in: 2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, CVPRW, 2024, pp. 5684–5692, <http://dx.doi.org/10.1109/CVPRW63382.2024.00577>.
- [47] T. Stoffregen, H. Daraei, C. Robinson, A. Fix, Event-Based kilohertz eye tracking using coded differential lighting, in: 2022 IEEE/CVF Winter Conference on Applications of Computer Vision, WACV, 2022, pp. 3937–3945, <http://dx.doi.org/10.1109/WACV51458.2022.00399>.
- [48] B. Zhang, Y. Gao, J. Li, H.K.-H. So, Co-designing a Sub-millisecond latency Event-based eye tracking system with submanifold sparse CNN, 2024, [arXiv:2404.14279](https://arxiv.org/abs/2404.14279) URL <https://arxiv.org/abs/2404.14279>.
- [49] S.S. Dhanda, B. Singh, P. Jindal, V. Kumar, S.K. Gupta, AES-8: A lightweight AES for resource-constrained IoT devices, *Trans. Emerg. Telecommun. Technol.* 36 (3) (2025) e70094.
- [50] L.-Y. Yeh, P.-J. Chen, C.-C. Pai, T.-T. Liu, An Energy-Efficient Dual-Field elliptic curve cryptography processor for Internet of Things applications, *IEEE Trans. Circuits Syst. II: Express Briefs* 67 (9) (2020) 1614–1618.
- [51] W. Lee, Y. Wang, M. Pedram, Optimizing a reconfigurable power distribution network in a multicore platform, *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* 34 (7) (2015) 1110–1123.
- [52] Intel Automotive, Vehicle platform power management standard proposal, 2025, <https://inivation.com/wp-content/uploads/2019/08/DAVIS346.pdf>. (Accessed 12 January 2025).