

Article

Low-Latency Machine Vision Based on a Neuromorphic Vision Sensor

Paul K. J. Park ^{1,2} , Junseok Kim ¹, Juhyun Ko ¹ and Yeoungjin Chang ^{2,*}

¹ Samsung Electronics, Hwaseong 18448, Gyeonggi-do, Republic of Korea; paulpark@gachon.ac.kr (P.K.J.P.); junseok7.kim@samsung.com (J.K.); juhyun03.ko@samsung.com (J.K.)

² Department of Semiconductor Display, Gachon University, Seongnam 13120, Gyeonggi-do, Republic of Korea

* Correspondence: yjchang@gachon.ac.kr

Abstract

Low-latency visual perception is essential for interactive machine vision on edge AI devices, but conventional frame-based image sensors impose frame period delays and generate dense image data that increase memory bandwidth and processing latency. Although Dynamic Vision Sensors (DVSs) are known to provide low latency, sparse output, and high dynamic range, these sensor-level properties do not automatically translate into practical application-level latency reduction on resource-constrained edge platforms. This paper presents a latency-driven sensing algorithm co-design approach for DVS-based low-latency machine vision. The main objective is to connect DVS sensor-level characteristics, event representations, task-dependent processing flows, and measured response times on mobile application processors. We first analyze latency requirements for three representative edge AI applications (i.e., person detection, gesture recognition, and Simultaneous Localization and Mapping (SLAM)), which correspond to different latency regimes and processing structures. We then describe the DVS operating principle, pixel-level event latency, and readout latency, showing how asynchronous event generation reduces sensing delay and suppresses redundant static background information before algorithmic processing. In contrast to prior event camera studies that mainly optimize a single task or a specific event representation, this work evaluates three task-specific event processing systems on mobile processors. Person detection achieves 92 ms processing latency on Exynos 7570, gesture recognition based on event-driven 4-DoF motion estimation achieves 20 ms latency on Exynos 5422, and SLAM achieves 15.9 ms latency on Snapdragon 845. These results satisfy the practical latency targets of the corresponding applications and demonstrate that DVS-based sensing can provide not only sensor-level speed advantages but also system-level latency benefits for AIoT, mobile, robotics, and AR/VR machine vision systems.



Academic Editors: Dong Zhang and Dah-Jye Lee

Received: 29 May 2026

Revised: 22 June 2026

Accepted: 25 June 2026

Published: 27 June 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

Keywords: Dynamic Vision Sensor; event-based vision; edge AI; neuromorphic; person detection; gesture recognition; simultaneous localization and mapping; address event representation; CMOS image sensor

1. Introduction

Low-latency visual perception has become an essential requirement for intelligent edge devices, mobile platforms, smart home systems, robotics, and human-machine interfaces. As visual sensing is increasingly deployed in always-on and interactive environments, a vision system is no longer evaluated only by recognition accuracy. It must also respond

within a timescale that is compatible with human perception, physical motion, and closed-loop control. In practical edge AI systems, latency is accumulated from multiple stages, including image acquisition, sensor readout, data transfer, preprocessing, neural network inference, postprocessing, and application-level decision-making. Even when the algorithm itself is optimized, the total response time can still be dominated by sensor exposure, frame readout, memory movement, or the processing of redundant image data. Therefore, reducing latency requires a joint consideration of the sensing device, data representation, and processing algorithm.

Conventional CMOS Image Sensors (CISs) acquire dense image frames at fixed temporal intervals. This frame-based acquisition model is highly effective for high-quality imaging, biometric recognition, and detailed scene understanding because it provides spatially complete intensity information. However, for interactive machine vision, fixed-frame acquisition introduces two fundamental limitations. First, the sensor output is synchronized to the frame period rather than to the timing of the physical event. For example, at 30 fps and 60 fps, the frame intervals are 33.3 ms and 16.7 ms, respectively, before additional delays from exposure, readout, image signal processing, and inference are considered. Second, the entire image frame is read out and processed even when only a small region of the scene changes. In many AIoT (Artificial Intelligence of Things) and smart home applications, such as person detection, gesture interaction, or motion-triggered sensing, the static background contains little task-relevant information but still consumes bandwidth, memory, and computation. This mismatch between dense frame acquisition and sparse scene dynamics becomes a major obstacle to low-latency and low-power operation on resource-constrained edge platforms.

Neuromorphic vision sensors, also known as event cameras or Dynamic Vision Sensors (DVSs), provide an alternative sensing paradigm that directly addresses these limitations. Inspired by biological retinas, a DVS does not output full image frames at fixed intervals. Instead, each pixel independently detects local changes in log intensity and asynchronously generates an event when the temporal contrast exceeds a threshold. Each event contains the pixel address, timestamp, and polarity, indicating where and when a local brightness increase or decrease occurred. Because events are generated only in response to changes, the sensor output is sparse in static scenes and temporally dense only where motion or illumination variation occurs. This event-driven operation provides several advantages over frame-based imaging, including microsecond-level temporal resolution, low latency, high dynamic range, reduced motion blur, and potentially lower data bandwidth for sparse dynamic scenes [1–4]. These properties make DVSs particularly suitable for machine vision tasks in which motion and temporal contrast are more important than full-frame texture.

Recent event camera studies have demonstrated the potential of this sensing paradigm in several latency-critical applications. Gehrig and Scaramuzza proposed a low-latency automotive perception system that combines RGB frames and event streams, using the rich appearance information of standard images together with the high temporal resolution and sparsity of events to reduce perceptual and computational latency [5]. Their work showed that event cameras can help overcome the bandwidth–latency trade-off of frame-based automotive vision, particularly by producing high-rate detections between image frames. Verma et al. focused on asynchronous object detection and argued that converting irregular event streams into dense tensors can diminish the inherent advantages of event-based sensing. They therefore proposed a spatiotemporal multigraph representation using spatial and motion vector graphs, improving event-based object detection accuracy and speed on automotive datasets [6]. Wang et al. addressed low-latency event-based visual object tracking through a slow–fast tracking paradigm, in which a high-precision slow tracker and a lightweight fast tracker are combined to adapt to different resource and

latency requirements [7]. In gesture recognition, Bhattacharyya et al. presented Helios, an extremely low-power event-based gesture recognition system for always-on smart eyewear, achieving real-time micro-gesture recognition using a compact event camera and an embedded processor [8]. In event-based SLAM and visual odometry, Niu et al. proposed ESVO2, a stereo event visual–inertial odometry system that reduces mapping complexity and improves pose tracking by using event local dynamics and IMU pre-integration [9].

These studies show that event-based vision has matured beyond sensor demonstrations and is now being actively explored for object detection, tracking, gesture recognition, automotive perception, and visual odometry. However, most existing works are primarily organized around a single task, a specific event representation, or a particular network architecture. For example, automotive detection studies focus on event–frame hybrid detection or asynchronous graph-based object detection; tracking studies focus on model-level latency reduction; gesture recognition studies optimize ultra-low-power wearable interaction; and event-based visual odometry studies emphasize mapping accuracy, robustness, or state-estimation performance. These directions are important, but they leave a broader system-level question insufficiently addressed: how can the intrinsic low-latency and sparse output properties of DVSs be translated into practical application-level latency reduction across different edge AI machine vision tasks and different mobile processor constraints?

This paper addresses that question by presenting a latency-driven sensing algorithm co-design approach for DVS-based low-latency machine vision. The novelty of this work is not the DVS principle itself, since low latency, sparsity, and high dynamic range are already established properties of event cameras. Rather, the contribution lies in connecting the sensor-level event generation mechanism, readout latency, event representation, task-specific processing pipeline, and measured mobile processor latency within a unified system-level framework. This perspective is important because DVS sensing should not be treated as a drop-in replacement for CIS imaging. Instead, the event representation and processing algorithm must be matched to the latency requirement and computational structure of each target application.

In this work, we focus on three representative applications: person detection, gesture recognition, and SLAM. These tasks were selected because they represent different latency regimes and different processing structures in interactive edge AI systems. Person detection is a discrete recognition task that should respond within approximately 100 ms for smart home or AIoT feedback. Gesture recognition requires lower latency because it is directly coupled to user motion and touchless control. SLAM requires the strictest latency because pose estimation is part of a closed-loop perception and control system for AR/VR, robotics, and mobile navigation. By evaluating these three tasks together, this work goes beyond single-task event camera demonstrations and clarifies how DVS-based sensing can be adapted to different application-level latency budgets.

The main contributions of this paper are summarized as follows: First, we formulate a task-dependent latency framework for interactive edge AI machine vision. Instead of only describing the general advantages of DVSs, we relate application-level latency requirements to sensing delay, event readout, event representation, algorithmic processing, and mobile processor constraints. This framework is used to derive practical latency targets for person detection, gesture recognition, and SLAM. Second, we analyze how the DVS pixel-level event generation mechanism and readout architecture contribute to the total sensing delay. By relating temporal contrast, event threshold, illumination-dependent pixel response, event rate-dependent readout latency, and task-specific event representation to downstream algorithmic requirements, we clarify how the intrinsic properties of DVS can be converted into system-level latency reduction. Third, we implement three task-specific event processing pipelines rather than relying on a single generic event representation.

Person detection uses an event–frame-based detection pipeline, gesture recognition uses event-driven 4-DoF motion estimation, and SLAM uses event-based feature extraction, map-point projection, feature matching, and 3D–2D pose estimation. This task-specific design shows that the optimal event processing strategy depends on the latency regime and information structure of the target application. Fourth, we experimentally validate the proposed sensing algorithm co-design on mobile application processors. The measured processing latencies are 92 ms for person detection on Exynos 7570, 20 ms for gesture recognition on Exynos 5422, and 15.9 ms for SLAM on Snapdragon 845. These results demonstrate that DVS-based event sensing can satisfy practical latency requirements on resource-constrained edge platforms and provide application-level benefits beyond the well-known sensor-level advantages of DVS.

The remainder of this paper is organized as follows: Section 2 discusses latency requirements for representative machine vision tasks and establishes practical latency targets for person detection, gesture recognition, and SLAM. Section 3 describes the operating principle, pixel architecture, and readout latency of the DVS. Section 4 presents the proposed low-latency machine vision applications and evaluates their processing latency on mobile edge platforms. Section 5 discusses the implications, limitations, and future directions of DVS-based low-latency machine vision.

2. Latency Requirements

2.1. Literature Review

In interactive edge AI systems, latency is commonly defined as the elapsed time between a physical change in the environment (or a user action) and the system’s corresponding output (decision, control signal, or visual feedback). For human-in-the-loop applications, latency requirements are ultimately constrained by human perception and sensorimotor response. The foundational studies by Miller [10] and Nielsen [11] identified three qualitatively distinct response time thresholds: 0.1 s for the perception of instantaneous causality, 1.0 s for maintaining uninterrupted cognitive flow, and 10 s for sustaining user attention. These thresholds have been reviewed and discussed in later HCI research by Attig et al. [12] and remain the de facto engineering benchmarks for interactive system design.

From a neurophysiology perspective, the speed at which the human visual system can categorize complex natural scenes provides an empirical lower bound on perception latency. Thorpe et al. [13] demonstrated that the visual processing required to determine whether a briefly flashed natural image contains an animal can be completed within 150 ms, as evidenced by differential event-related potentials emerging at this timescale. This finding implies that feedforward visual processing in biological systems operates at sub-150 ms speeds even for complex categorization tasks, establishing a neuroscience-grounded reference point for machine vision latency targets.

In addition, Shneiderman [14] reviewed the broader literature on computer response times and concluded that a 2 s threshold is appropriate for many online conversational tasks, while direct manipulation interfaces demand responses well below 100 ms to feel natural [10]. Therefore, machine vision tasks intended for real-time interaction should target response times below 100 ms whenever possible, and at minimum remain well below the latency range where users perceive lag or experience degraded control.

2.2. Object Detection

Object (or person) detection in smart home and IoT environments often triggers actions that are immediately noticeable to users, such as wake-up display activation, adaptive lighting, or safety alerts. For such feedback-driven tasks, the system should respond fast

enough to maintain the illusion of immediacy. The widely cited 0.1 s instantaneous bound in interactive systems [10,12] provides a practical engineering target for end-to-end detection latency, encompassing both sensor acquisition and algorithmic processing.

From a neurophysiology perspective, rapid visual categorization in humans can be achieved within 150 ms under demanding conditions, indicating that fast recognition in biological vision occurs on a sub-150 ms timescale [13,15]. It has also been reported that selective motor responses to complex visual stimuli can be initiated within 250–300 ms, with differential neural processing emerging as early as 150 ms post-stimulus [13]. These findings establish a neuroscience-motivated ceiling for interactive detection latency: if a machine vision system is to match the responsiveness of biological vision, it must complete the full sensing-to-decision processing within approximately 100–150 ms.

However, on resource-constrained AIoT edge platforms lacking dedicated neural accelerators, detection latency may exceed 100 ms even with aggressively optimized models, making the sensor-side latency budget all the more critical [16,17]. Taken together, these algorithmic and perceptual considerations motivate an ~100 ms latency target for interactive object/person detection (both sensor and processing combined).

2.3. Gesture Recognition

Gesture recognition is typically used for direct manipulation (touchless UI, command triggering, or continuous control), where humans are particularly sensitive to delays between their physical action and the corresponding visual or haptic feedback. Mitra and Acharya [18] categorized gesture-based interaction into two paradigms: static posture recognition (hand shape classification) and dynamic gesture recognition (spatiotemporal motion trajectory analysis). For dynamic gestures involving continuous control, such as swipe, pinch, or wave motions, the temporal fidelity of the feedback loop directly determines the perceived quality of the interaction.

For user interfaces, ~100 ms remains an upper bound for instantaneous interaction [10,12]; beyond this threshold, lag becomes perceptible and can degrade control performance. At the lower end, human vision can discriminate temporal structure on the order of a few tens of milliseconds in several psychophysical paradigms (e.g., temporal position/ordering acuity of around 20–30 ms) [19]. This suggests that responsive gesture interaction benefits from latencies closer to tens of milliseconds, especially when continuous feedback is desired. Köpüklü et al. [20] demonstrated real-time hand gesture detection with a two-stage 3D-CNN architecture achieving latencies below 30 ms on GPU platforms, while Molchanov et al. [21] showed that recurrent 3D convolutional networks can classify dynamic hand gestures online with sub-frame latency.

For event camera-based gesture recognition, the sparse and asynchronous nature of the event stream enables further latency reduction. Recent work on neuromorphic gesture recognition has demonstrated end-to-end latencies as low as 60 ms for micro-gesture classification on ultra-low-power edge hardware [8], confirming that event-driven sensing is well suited for latency-critical gesture applications. In the broader context of touchless interaction for smart home and IoT environments, gesture interfaces must also operate robustly under varying lighting conditions and at extended working distances, which favors the high dynamic range and motion-centric output of DVSs [22]. Accordingly, we set a practical 10–100 ms requirement window for gesture recognition, where ~100 ms represents the instantaneous interaction limit and the lower bound reflects the range where humans can exploit fine temporal structure for smooth, continuous feedback.

2.4. SLAM

For SLAM and visual-inertial tracking, latency is tightly coupled to closed-loop stability and, in AR/VR applications, user comfort. Motion-to-photon (MTP) latency—the elapsed time from a user’s physical head movement to the corresponding visual update on the display—is a well-established contributor to cybersickness symptoms including nausea, disorientation, and oculomotor discomfort [23]. Stauffert et al. [24] conducted a comprehensive review of MTP latency effects and confirmed that latency jitter, in addition to absolute latency magnitude, significantly degrades user comfort and task performance in immersive environments.

Commonly used design targets emphasize approximately 20 ms or lower MTP latency threshold as the requirement for a comfortable VR experience [24,25]. Elbamby et al. [25] analyzed the end-to-end latency budget for untethered VR systems and showed that achieving this target requires coordinated optimization across sensing, computation, and communication subsystems, with the visual tracking component typically allocated a budget of 10–15 ms. Because MTP latency is a system-level sum (sensing, estimation, rendering, display), the tracking/pose estimation component must operate within a smaller internal budget to leave headroom for downstream rendering and display refresh stages.

From an algorithmic perspective, state-of-the-art visual SLAM systems such as ORB-SLAM [26] operate at 20–30 ms per frame on desktop hardware, while lightweight visual-inertial odometry methods could achieve sub-10 ms tracking latency [27]. Cadena et al. [28] provided a comprehensive survey of SLAM research and identified real-time operation, robustness, and resource efficiency as the three principal challenges for deploying SLAM on edge platforms. For event camera-based SLAM, the asynchronous and high-temporal-resolution event stream offers the potential to decouple pose estimation from the fixed-rate frame schedule of conventional cameras, enabling continuous tracking updates at rates exceeding 1 kHz and correspondingly reducing the effective sensing latency. This motivates the <20 ms target for SLAM latency used in our task requirements, where the sensing component is expected to consume only a fraction of the total MTP budget.

2.5. Evaluation Criteria

The latency requirements discussed in Sections 2.1–2.4 are used as the evaluation basis for the experimental validation in Section 4. In this work, latency is interpreted as a system-level budget consisting of sensor-side delay and processor-side application delay. The sensor-side delay ($L_{sensing}$) includes the pixel-level event generation latency and event readout latency, which are analyzed in Section 3. The processor-side application delay ($L_{processing}$) includes event accumulation or representation, feature extraction, inference, motion estimation, feature matching, pose estimation, and task-level decision-making, depending on the target application. Therefore, the total response latency can be conceptually expressed as

- Total response latency: $L_{total} = L_{sensing} + L_{processing}$

In the experimental evaluation, we report the measured processor-side application latency on mobile application processors, while the DVS sensor-side latency is analyzed separately based on the pixel response and readout characteristics. This separation allows us to verify whether each DVS-based processing pipeline satisfies the practical latency budget required by its target application.

Table 1 summarizes how the latency requirements derived in Section 2 are connected to the experimental design and evaluation criteria used in Section 4. Person detection is evaluated against an approximately 100 ms response target because it is used for discrete AIoT or smart home feedback. Gesture recognition is evaluated against a 10–100 ms latency window, with tens-of-millisecond latency preferred for responsive touchless interaction.

SLAM is evaluated against a stricter sub-20 ms target because visual tracking and pose estimation must operate within the motion-to-photon or closed-loop control budget of AR/VR, robotics, and mobile navigation systems.

Table 1. Task-dependent latency targets and evaluation criteria used in this work.

Application	Practical Latency Target	Evaluation Metric	Experimental Platform	Evaluation Criterion
Person detection	≤ 100 ms	Processing latency from DVS event representation to detection output	Exynos 7570	The measured latency should be within the interactive AIoT response target
Gesture recognition	10–100 ms, preferably tens of milliseconds	Processing latency of event-driven 4-DoF motion estimation	Exynos 5422	The measured latency should support responsive touchless interaction
SLAM	< 20 ms	Processing latency of event-based feature extraction, matching, and pose estimation	Snapdragon 845	The measured latency should fit within the tracking budget for closed-loop perception

3. Neuromorphic Vision Sensor

3.1. Operating Principle

A DVS is an activity-driven image sensor that converts local temporal changes in brightness into asynchronous events. Unlike a conventional CIS, which acquires full image frames at a fixed exposure and readout rate, a DVS does not periodically sample the entire pixel array. Instead, each pixel independently monitors the change in logarithmic light intensity and generates an event only when the change exceeds a predefined temporal contrast threshold [1,2,29]. Therefore, the output of a DVS is not a sequence of dense intensity images but a stream of sparse events represented by

$$e_k = (x_k, y_k, t_k, p_k) \quad (1)$$

where (x_k, y_k) is the pixel address, t_k is the timestamp, and p_k is the polarity indicating whether the local brightness has increased or decreased. The event generation condition can be expressed as

$$\Delta L(x, y, t) = L(x, y, t) - L(x, y, t_{ref}) \geq C_{ON} \quad (2)$$

for an ON event, and

$$\Delta L(x, y, t) = L(x, y, t) - L(x, y, t_{ref}) \leq -C_{OFF} \quad (3)$$

for an OFF event, where $L = \log(I)$ is the logarithmic intensity, I is the photocurrent or incident light intensity, t_{ref} is the time of the previous event or reset at the same pixel, and C_{ON} and C_{OFF} are the positive and negative contrast thresholds, respectively. After an event is generated, the pixel updates its reference level and continues monitoring subsequent brightness changes. This local reset-and-detect operation enables each pixel to operate asynchronously and independently from the global frame timing.

This operating principle provides two important advantages for low-latency machine vision, but its system-level benefit appears only when the event stream is matched to the downstream algorithm. First, the sensor does not need to wait for the next frame boundary to report a visual change. When motion or contrast variation occurs in the scene, the corresponding pixels can immediately generate events. Therefore, the earliest availability of visual information is determined mainly by the pixel response and event

readout latency, rather than by the frame period of the image sensor. Second, the DVS suppresses redundant information from static regions. Since no events are generated from unchanged background areas, the amount of sensor output is naturally proportional to scene activity. This reduces the number of pixels, features, proposals, or candidate matches that must be processed by downstream algorithms. For person detection, this reduces the amount of background texture that the detection network must analyze. For gesture recognition, it allows the algorithm to operate directly on motion-induced events without dense hand segmentation or optical flow estimation. For SLAM, it concentrates computation on recently active edge structures that are useful for feature tracking and pose estimation. Thus, the DVS contributes to low latency not only by reducing sensor acquisition delay, but also by changing the computational structure of the machine vision pipeline.

Figure 1 compares the outputs of a conventional CIS and a DVS in an indoor environment. The CIS image in Figure 1a contains dense spatial information over the entire field of view, including static background structures such as walls, furniture, and illumination patterns. This type of frame is useful for high-quality visual recognition, but it also requires the full image to be transferred and processed even when only a small part of the scene is relevant to the task. In contrast, the DVS output in Figure 1b emphasizes only the moving contours and temporal contrast regions generated by the person's motion. Static background components are largely suppressed because they do not produce temporal brightness changes. As a result, the moving person can be isolated with a much smaller amount of visual data, allowing simple event accumulation, clustering, or lightweight feature extraction to provide a useful input representation for machine vision.

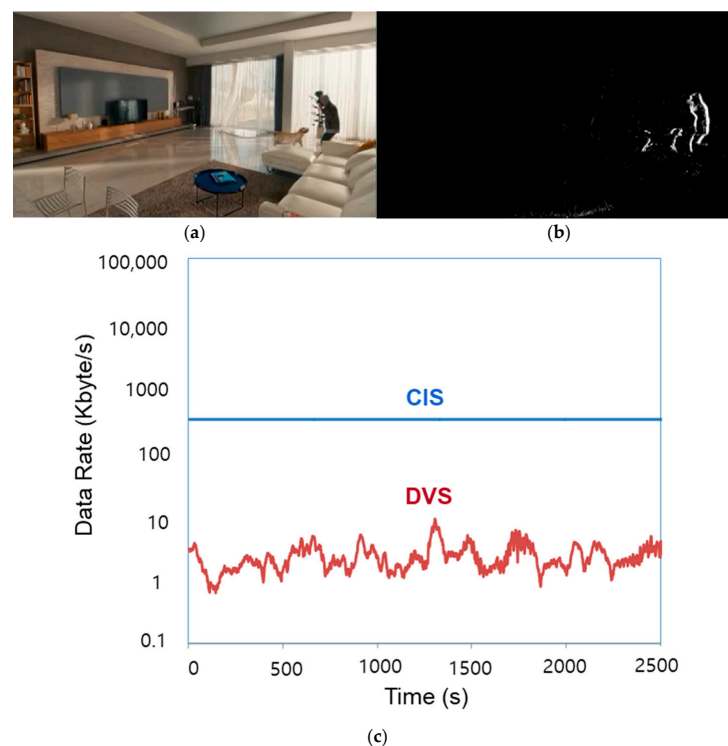


Figure 1. Comparison of frame-based and event-based sensing in an indoor machine vision scenario. (a) Conventional CIS output captured as a dense image frame. The frame includes both the moving person and static background information. (b) DVS output generated from asynchronous ON/OFF events. The event representation highlights motion-dependent temporal contrast regions while suppressing static background content. (c) Data rate comparison between CIS and DVS during repeated human motion in a home environment. The CIS produces frame data at a nearly fixed rate determined by the imaging configuration, whereas the DVS data rate varies with scene activity and decreases when the scene is static.

Figure 1c further illustrates the difference in data generation behavior between CIS and DVS. In a frame-based CIS, the data rate is primarily determined by the frame rate, resolution, bit depth, and interface format. Therefore, the sensor continues to produce a relatively constant amount of data regardless of whether the scene is static or dynamic. In a DVS, by contrast, the data rate is activity-dependent. When the person moves, events are generated along the moving body contours and the DVS data rate increases. When the scene becomes static, the event rate decreases significantly because unchanged pixels do not continuously transmit intensity values. This activity-dependent data generation is a key reason why DVS-based sensing can reduce both sensing delay and processing burden in motion-centric applications such as person detection, gesture recognition, and SLAM.

It should be noted that a DVS does not replace a CIS for all imaging tasks. A CIS provides dense texture, color, and absolute intensity information, which remain important for high-resolution image understanding and appearance-based recognition. The advantage of a DVS is most evident in latency-critical and motion-centric tasks, where temporal changes carry the most relevant information. Therefore, in this work, the DVS is used as a low-latency sensing front-end for interactive machine vision rather than as a general-purpose image-quality sensor.

3.2. Pixel Latency

Figure 2 illustrates the pixel-level mechanism enabling event generation. The pixel architecture typically consists of three fundamental stages: the logarithmic photoreceptor, the differencing circuit (change amplifier), and the comparative threshold stage.

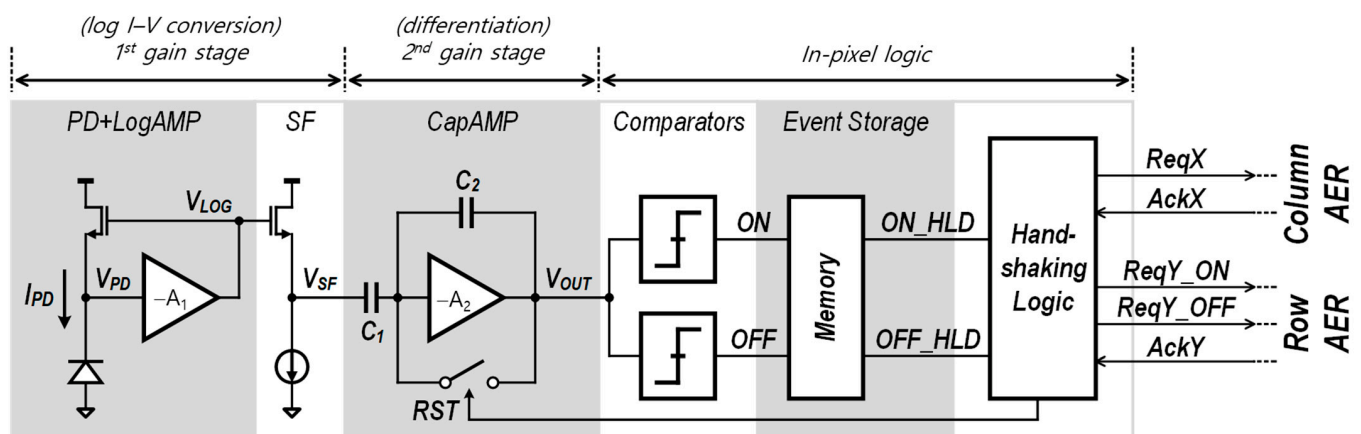


Figure 2. Pixel-level architecture and event generation mechanism of the DVS. Each pixel converts incident light into a photocurrent and logarithmically detects temporal intensity changes. When the change in log intensity exceeds the preset ON or OFF contrast threshold, the pixel asynchronously generates an event request containing address, timestamp, and polarity information.

1. **Logarithmic photoreceptor stage:** This front-end stage is responsible for sensing the continuous photocurrent I_{PD} generated by the photodiode. It utilizes the subthreshold characteristics of MOS transistors to convert the photocurrent into a voltage V_{LOG} that is proportional to the logarithm of the light intensity. This logarithmic compression allows the sensor to handle a wide dynamic range (DR), often exceeding 100 dB, ensuring robust operation in environments ranging from dim indoor lighting to bright sunlight.
2. **Differencing circuit (change amplifier):** This stage acts as a high-pass filter that removes the DC component (static background) and amplifies only the temporal changes in V_{LOG} . By using a capacitive-feedback amplifier, the circuit stores the initial voltage level and monitors the deviation ΔV_{LOG} . This mechanism is the key to DVS's

data sparsity, as it suppresses redundant information from static scenes and highlights only moving edges or intensity fluctuations.

3. Comparative threshold stage: The amplified change signal is compared against two preset global thresholds: C_{ON} for positive contrast (brightening) and C_{OFF} for negative contrast (darkening). When the signal exceeds either threshold, the pixel asynchronously generates an event request to the peripheral arbitration logic.

The primary advantage of DVS is its sub-millisecond sensing latency, which is significantly faster than the 16–33 ms frame integration time of standard 30–60 fps CIS cameras. In DVS, the pixel latency L_{pixel} —the time interval between a physical change in light and the generation of an electronic event—is not determined by a clock but by the internal dynamics of the pixel circuit.

The speed of the pixel is primarily limited by the bandwidth of the logarithmic photoreceptor stage. In the subthreshold region, the relationship between the gate-to-source voltage and the drain current (I_{PD}) of the feedback transistor is expressed as follows [2,29,30]:

$$V_{LOG} = V_0 + \frac{U_T}{\kappa} \ln \left(\frac{I_{PD}}{I_0} \right) \quad (4)$$

where U_T is the thermal voltage (~ 26 mV at room temperature), κ is the subthreshold slope factor, and I_0 is the reference current. Since the node voltage V_{LOG} must change to reflect a change in I_{PD} , the node's parasitic capacitance C_p must be charged or discharged by the photocurrent itself. The small-signal transconductance (g_m) of the subthreshold transistor is given by

$$g_m = \frac{\partial I_{PD}}{\partial V_{gs}} = \frac{\kappa I_{PD}}{U_T} \quad (5)$$

The time constant τ of the photoreceptor node, which determines the response speed to a change in light, is defined by the ratio of the capacitance to the transconductance:

$$\tau \approx \frac{C_p}{g_m} = \frac{C_p U_T}{\kappa I_{PD}} \quad (6)$$

This equation shows that the pixel's temporal resolution is inherently tied to the available light. In high-illumination conditions (high I_{PD}), τ becomes very small, leading to high-speed sensing. Conversely, τ increases under low-illumination conditions, which explains the observed latency degradation in dim environments. The pixel latency L_{pixel} can be modeled as the time required for the voltage change to reach the threshold (C_{ON} , C_{OFF}) after a contrast change of $\Delta I/I$. Considering the amplification factor of the differencing circuit, the latency is approximately

$$L_{pixel} \approx n\tau = n \left(\frac{C_p U_T}{\kappa I_{PD}} \right) \quad (7)$$

where n is a coefficient related to the contrast sensitivity and the signal-to-noise ratio requirements of the comparator. Commercial and prototype DVSs typically exhibit sub-millisecond latency under adequate illumination, and reported or specified pixel latency can reach tens to hundreds of microseconds under high illuminance around or above 1 klux.

The fundamental difference between DVS and CIS lies in the latency–bandwidth trade-off. Conventional CIS pixels integrate photons over a fixed exposure time (T_{exp}), which acts as a low-pass filter and introduces a mandatory delay before any readout can occur.

- CIS sensing latency: $L_{CIS} = T_{exp} + T_{readout} \approx 1/(2f) + T_{readout}$;
- DVS sensing latency: $L_{DVS} = L_{pixel} + L_{readout}$.

In DVS, L_{pixel} is determined by the time required for the logarithmic photoreceptor and differencing circuit to convert a physical brightness change into a threshold-crossing voltage signal, while $L_{readout}$ is determined by the event arbitration and output interface. Unlike CIS latency, which is bounded by the exposure and frame readout period, L_{pixel} is governed by local circuit dynamics and by the optical input itself. A larger temporal contrast produces faster threshold crossing, while a smaller contrast requires a longer integration time before an event is generated. Similarly, a lower event threshold can reduce the time required for event generation, although it may also increase background activity and noise events. Illumination is another important factor: under higher illumination, the photocurrent increases, the photoreceptor time constant decreases, and the pixel can respond more rapidly; under low illumination, the reduced photocurrent increases the time constant and can increase pixel latency. This relationship between temporal contrast, event threshold, illumination, and pixel latency is important for downstream algorithm design. For latency-critical tasks, the event representation should be selected so that the algorithm can exploit the earliest reliable events without waiting for a full image frame. In person detection, events may be accumulated over a short temporal window to form an event image compatible with a lightweight detector. In gesture recognition, short-window event clusters or motion vectors can be processed directly because the task depends primarily on hand motion rather than dense texture. In SLAM, recent edge events can be used for feature extraction, map-point matching, and pose estimation. Therefore, the DVS pixel is not only a faster sensing element; it also defines the temporal granularity at which the downstream algorithm can begin processing. This sensor-to-algorithm coupling is the basis for converting the intrinsic low-latency property of DVS into application-level latency reduction.

3.3. Readout Latency

In a DVS, once a pixel generates an event, the event must traverse a shared communication channel to reach the output bus. Because multiple pixels may generate events concurrently, a readout arbitration or scheduling mechanism is required to serialize event transmission without data loss. The design of this readout architecture directly determines the relationship between the instantaneous event rate and the readout latency and, therefore, has a first-order impact on the worst-case sensor-level latency budget [31].

Figure 3 compares two representative readout architectures for DVS event transmission: (a) word-serial Address Event Representation (AER) with row and column arbitration, and (b) sequential column-scanning readout. In the AER architecture (Figure 3a), each pixel that has generated an event asserts a request signal to a shared row arbiter. The arbiter grants access to one requesting row at a time, after which a column arbiter selects the active pixel within the granted row. The selected pixel's address (row and column indices) is placed on the output bus along with the timestamp and polarity, completing one event transaction. This process repeats for each pending event in a first-come-first-served or priority-based order. The key characteristic of AER arbitration is that the readout delay per event is approximately constant (determined by the handshaking cycle time of the arbiter), but the total readout latency experienced by any given event depends on the number of events queued ahead of it—i.e., the readout delay scales with the instantaneous event rate.

In the sequential column-scanning architecture (Figure 3b), the readout controller scans through the columns of each row in a fixed sequence, polling each column for pending events. When a pending event is found, it is read out; otherwise, the scanner advances to the next column. This approach trades arbitration complexity for deterministic scan timing: the readout is bounded by the scanning schedule (one complete scan covers all columns within a row period), which can reduce worst-case contention latency under burst event

conditions because no single event can be indefinitely delayed by arbitration starvation. However, the scanning introduces a fixed latency component proportional to the scan period, even when the event rate is low. In contrast, the AER architecture incurs minimal latency under sparse conditions (few events to arbitrate) but can exhibit queuing delays under high event rate bursts.

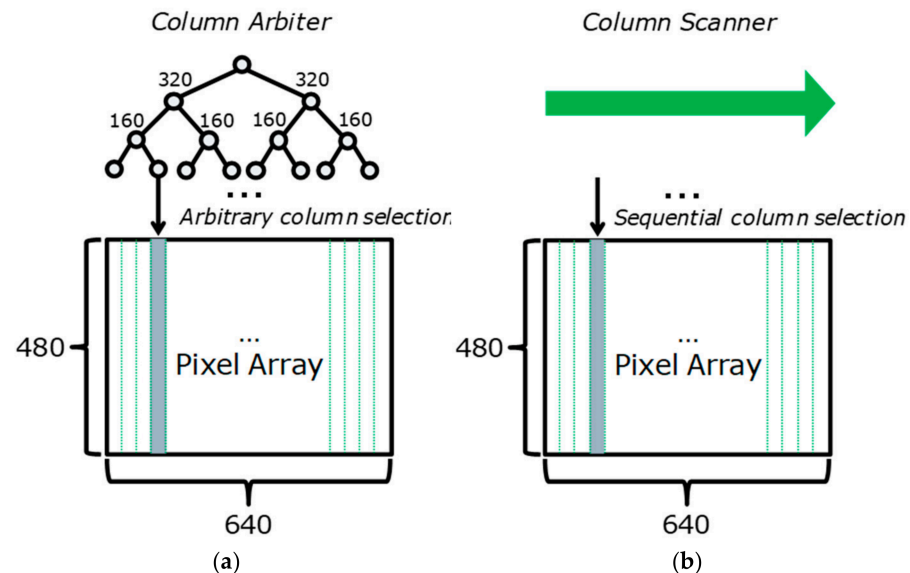


Figure 3. Comparison of two DVS event readout schemes. (a) Arbitration-based address-event readout, in which active pixels asynchronously request access to the output bus and are transmitted after arbitration. This scheme provides low latency for sparse event streams but can experience queuing delays during burst events. (b) Sequential scanning or column-selection readout, in which event information is read according to a predefined scan order. This scheme introduces a fixed scan-dependent delay but provides bounded latency under dense event activity. In both (a,b), the green dashed lines represent individual columns within the pixel array, the ellipsis indicates omitted columns, and the gray area highlights the currently selected column. In (b), the green arrow indicates the direction of the sequential scan.

Figure 4 presents a quantitative comparison of the readout latency for the two architectures at 640×480 resolution as a function of the instantaneous event rate. The AER latency was estimated from the event service time and the instantaneous number of queued events, whereas the sequential column-selection latency was estimated from the scan period. The calculation assumes a VGA array and the maximum output event bandwidth summarized in Table 2. For the word-serial AER scheme (Figure 4, blue curve), the readout latency increases approximately linearly with the event rate because each additional event occupies one arbitration cycle on the shared bus. At low event rates (e.g., ≤ 1 Meps (million events per second)), the AER latency is negligible (sub-microsecond per event), making it highly efficient for the sparse event streams typical of indoor smart home scenarios. However, as the event rate approaches the bus bandwidth limit (e.g., 300 Meps for the sensor in Table 2), queuing delays accumulate and the per-event readout latency increases. For the sequential column-scanning scheme (Figure 4, red curve), the readout latency is bounded by the scan period and remains relatively flat across a wide range of event rates, but exhibits a non-zero floor even at very low event rates due to the fixed scanning overhead.

The practical implication of this comparison is that the choice of readout architecture should be matched to the expected event rate distribution of the target application. For always-on smart home sensing, where the scene is predominantly static with occasional bursts of motion, AER arbitration is well suited because it delivers negligible latency during idle periods while tolerating moderate burst rates. For applications with sustained high

event rates (e.g., high-speed robotics or automotive scenarios with many moving objects), sequential scanning may offer more predictable worst-case latency guarantees. Importantly, for both architectures, the readout latency can be kept below 0.3 ms even under worst-case event rate conditions at VGA resolution [31], confirming that the digital readout contribution to the total sensor latency is small compared with the pixel-level photoreceptor response time analyzed in Section 3.2.

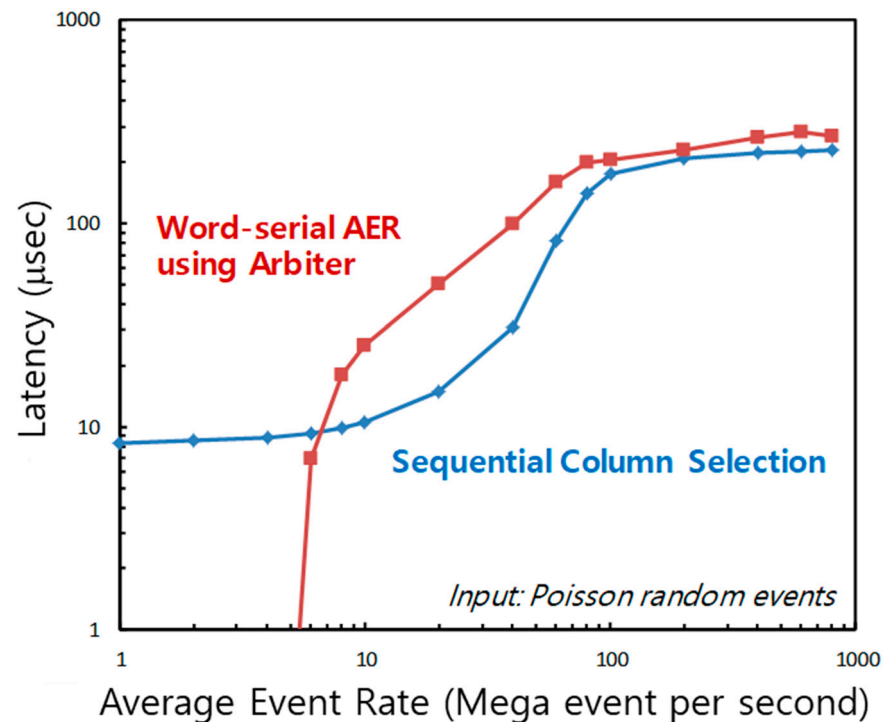


Figure 4. Calculated readout latency of word-serial AER and sequential column-selection readout using the event bandwidth and readout parameters of the VGA DVS in [31]. The AER readout latency increases with instantaneous event activity because events are arbitrated and transmitted sequentially, whereas sequential column selection provides scan-schedule-bounded latency. In the evaluated condition, the readout latency remains below 0.3 ms.

Table 2. Key specifications of the DVS used in this work.

Specifications	Unit	DVS
Optical Format	inch	1/2.5
Resolution		640 × 480
Pixel Pitch	μm	9
Chip Size	μm ²	8000 × 5800
Voltage	V	2.8 (Analog), 1.2 (Digital)
Operating Illumination Range	lux	3 to 100,000
Maximum Equivalent Frame Rate	FPS	>2000
Interface		MIPI, Parallel
Maximum Data Bandwidth	Mbps	2500
Power Consumption	mW	65

The readout architecture also affects how sensor-level latency is translated into algorithm-level latency. In sparse scenes, where only a small number of pixels generate events, arbitration-based AER readout provides very low delay because events can be transmitted almost immediately after threshold crossing. This behavior is suitable for always-on person detection and gesture recognition, where the scene is often static and only a moving person or hand generates events. In contrast, when the event rate becomes high because

of fast camera motion, dense scene edges, or multiple moving objects, queuing delay may increase in an AER readout scheme. In such cases, a bounded latency scanning or column-selection architecture can provide more predictable worst-case behavior, which is useful for SLAM or robotics applications that require stable tracking latency. Therefore, the readout architecture should be considered together with the expected event rate distribution and the downstream algorithmic workload. The practical latency of a DVS-based machine vision system is not determined solely by the pixel response time; it is determined by the combined effect of pixel latency, readout latency, event accumulation or representation latency, and the processing complexity of the target algorithm.

Table 2 summarizes the key specifications of the DVS sensor used in this work. The sensor features a 1/2.5-inch optical format with a native resolution of 640×480 pixels at a $9 \mu\text{m}$ pixel pitch, providing a spatial resolution comparable to standard VGA-class CIS devices while maintaining the compact die size ($8000 \times 5800 \mu\text{m}$) required for integration into mobile and IoT edge platforms. The sensor operates at 2.8 V (analog) and 1.2 V (digital) supply voltages, achieving a total power consumption of 65 mW—well within the power budget of battery-operated smart home devices and wearable AR/VR headsets.

The dynamic range specification of >100 dB (equivalent to a scene illuminance range from 3 lux to over 100 klux) is a direct consequence of the logarithmic photoreceptor compression described in Section 3.2, Equation (4). This wide dynamic range eliminates the need for auto-exposure control or multi-frame HDR synthesis, both of which introduce additional latency and computational overhead in CIS-based systems. The maximum effective frame rate exceeds 2000 equivalent frames per second, reflecting the high temporal resolution enabled by the asynchronous pixel operation. In practice, the temporal resolution is limited not by a global frame clock but by the pixel-level event latency and digital readout bandwidth. The sensor provides a MIPI interface with a maximum data bandwidth of 2500 Mbps, which is sufficient to sustain event output rates of up to 300 Meps without bus congestion.

Taken together, Sections 3.1–3.3 show how DVS sensor characteristics are connected to downstream processing latency. The temporal contrast threshold determines when a pixel generates an event, the illumination-dependent photoreceptor response determines how quickly the threshold can be reached, and the readout architecture determines how quickly the generated event can be delivered to the processor. Once delivered, the sparse and motion-selective event stream reduces the size and complexity of the algorithmic input. In this work, these sensor-level characteristics are mapped to three application-level processing pipelines: event-frame-based person detection, event-driven 4-DoF gesture motion estimation, and event-based SLAM. This mapping clarifies how the intrinsic properties of DVS can be converted into system-level latency reduction on mobile edge processors.

3.4. Latency Decomposition

In this work, sensing latency (L_{sensing}) denotes the latency required for the DVS to detect a physical brightness change and deliver the corresponding event to the processor. It includes the pixel-level event generation latency (L_{pixel}) and the event readout latency (L_{readout}). Here, L_{sensing} is not directly included in the measured processing latency values reported in Section 4. Instead, it is separately defined and analyzed in Section 3 through the pixel response model and readout latency analysis.

Processing latency ($L_{\text{processing}}$) denotes the latency required after DVS events are available to the processing pipeline. It includes both event representation latency ($L_{\text{representation}}$) and algorithm latency ($L_{\text{algorithm}}$):

- Processing latency: $L_{\text{processing}} = L_{\text{representation}} + L_{\text{algorithm}}$

$L_{representation}$ is the time required to convert or organize the asynchronous event stream into the input representation used by the downstream algorithm. Depending on the application, this representation may be an event image, an event cluster, a motion vector, a time surface, or an event-based feature set. $L_{algorithm}$ is the computation time required by the machine vision algorithm itself, such as CNN-based detection, event-driven gesture motion estimation, feature extraction, feature matching, map-point projection, or pose estimation. In practical implementation, $L_{representation}$ and $L_{algorithm}$ do not necessarily increase the processing latency as two fully sequential terms. The event representation stage and the algorithmic processing stage can be implemented as pipelined or parallel threads on the application processor. In this structure, one thread continuously receives and organizes incoming events into the task-specific representation, while another thread performs the downstream algorithm using the most recently available representation. Therefore, a significant portion of the event representation latency can be overlapped with algorithm execution. This overlap is particularly important for high-speed motion-centric applications such as gesture recognition and SLAM. In these applications, the event representation can be generated by accumulating events over a short temporal window, typically within 5 ms in our implementation, because the relevant information is concentrated in recent motion-induced events rather than in dense spatial texture. As a result, the representation update is short compared with the downstream algorithmic computation and can be largely hidden by the pipelined implementation. Therefore, the effective processing latency becomes dominated mainly by $L_{algorithm}$, rather than by the event representation stage.

This pipelined processing structure was used in all three application demonstrations in this work. For person detection, event accumulation and event image update were performed as a representation stage, while the CNN-based detection pipeline [32] operated on the updated event representation. For gesture recognition, event clustering or motion vector representation was updated continuously while the 4-DoF motion estimation algorithm processed the current event information. For SLAM, event-based feature representation and feature update were performed in parallel with feature matching, map-point projection, and pose estimation. Therefore, the measured processing latencies reported in Section 4—92 ms for person detection, 20 ms for gesture recognition, and 15.9 ms for SLAM—represent the practical processing latencies of the implemented pipelined systems, in which event representation and algorithmic processing are overlapped as much as possible and the remaining latency is mainly determined by the algorithmic computation.

This decomposition clarifies that the latency benefit of DVS-based machine vision originates from two different levels. At the sensing level, the DVS reduces latency by removing the fixed-frame period delay of conventional CIS cameras and by delivering local brightness changes asynchronously through pixel-level event generation and event readout. At the processing level, the sparse and motion-selective event stream reduces the amount of input data that must be represented and processed by downstream algorithms. Thus, the latency reduction discussed in this work should be understood as the combined result of separately analyzed sensing latency and experimentally measured processing latency.

4. Low-Latency Machine Vision

The experimental evaluation in this section follows the latency definition introduced in Section 3.4. The reported latency values correspond to processing latency, which includes event representation and algorithmic processing on the target mobile application processor. The sensing latency, consisting of pixel-level event generation latency and readout latency, is analyzed separately in Section 3 and is not merged into the measured processing latency values reported here. Therefore, the results in this section evaluate how efficiently each DVS-based event stream can be represented and processed for the target machine vision

task after event sensing and readout. In all three applications, event representation and algorithmic processing were implemented using a pipelined or parallel-thread structure on the mobile application processor, so that the representation update was overlapped with downstream computation as much as possible. Accordingly, the reported processing latency mainly reflects the effective application-processing latency dominated by the algorithmic computation rather than a simple serial sum of event representation and algorithm latency.

4.1. Person Detection

Person detection is a representative discrete recognition task for AIoT and smart home systems because it can trigger immediately noticeable actions such as display wake-up, adaptive lighting, occupancy sensing, or safety notification. In this work, person detection is used as the first representative application for evaluating whether DVS-based machine vision can satisfy the approximately 100 ms latency target defined in Section 2. The purpose of this subsection is not to propose a new person detection algorithm, but to validate the processing latency of a previously established DVS-based detection pipeline on a resource-constrained mobile application processor.

For person detection, we adopted the DVS-based event representation and detection configuration reported in our previous event-based edge AI machine vision study [33]. In that study, the DVS event stream was converted into an event-derived image representation that preserves motion-induced human contours while suppressing static background regions. The event-derived image was then processed by a Faster R-CNN-style detection pipeline using a lightweight CNN backbone and a region-proposal-based detection head. The detailed dataset configuration, annotation procedure, training setup, and detection performance evaluation are reported in [33]. In the present manuscript, this previously validated detection module is used as a representative person detection pipeline for system-level latency validation.

The adopted detection framework is based on Faster R-CNN, originally proposed by Ren et al. [32]. Faster R-CNN is a two-stage object detection architecture comprising a shared convolutional backbone for feature extraction, a Region Proposal Network (RPN) for generating candidate object regions, and a detection head that performs bounding box regression and classification on the proposed regions. The key innovation of Faster R-CNN over its predecessors, R-CNN and Fast R-CNN, is the introduction of the RPN, a fully convolutional network that shares features with the detection network and generates region proposals with relatively low computational cost by using anchor boxes of multiple scales and aspect ratios. This architecture provides a favorable balance between detection accuracy and computational efficiency, making it suitable for adaptation to DVS-based edge deployment.

Figure 5 illustrates the architecture of the DVS-adapted person detection algorithm used in this work. The algorithm consists of three stages: (i) event-to-image conversion, in which the asynchronous DVS event stream is accumulated over a short temporal window to generate a two-dimensional event-derived image encoding the spatial distribution of recent brightness changes; (ii) feature extraction through a lightweight convolutional backbone that produces a multi-scale feature map from the event-derived image; and (iii) the RPN and detection head, which generate candidate person regions and predict bounding boxes with classification scores.

A key advantage of using DVS input rather than conventional CIS frames for person detection is the reduction in computational complexity of the detection pipeline. This advantage arises from two complementary mechanisms. First, because the DVS event-derived image inherently suppresses static background and preserves primarily motion-related contours, the input contains much less irrelevant texture and background clutter than

a conventional RGB frame. This natural foreground–background separation simplifies feature extraction: the backbone network does not need to learn to suppress complex static background patterns, and fewer unnecessary proposals are generated by the RPN. Second, the sparse and edge-like nature of DVS event-derived images makes the detection pipeline more amenable to network optimization techniques such as architectural reduction, channel pruning, and mixed-bit quantization. As reported in [33], combining DVS event encoding with task-specific network optimizations achieved an $11\times$ speed-up for human detection and pose estimation on edge hardware while maintaining practical detection performance.

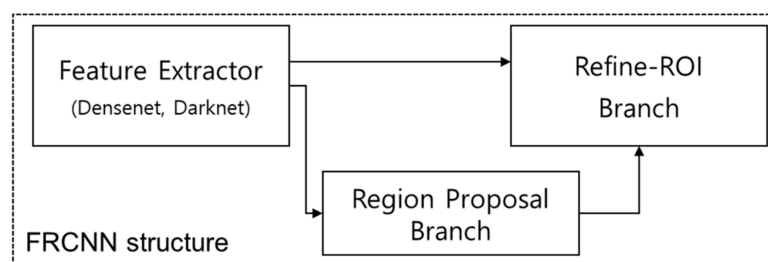


Figure 5. DVS-based person detection procedure based on the Faster R-CNN architecture [32,34]. The asynchronous DVS event stream is converted into a frame-like event representation and processed by a lightweight CNN backbone. A Region Proposal Network (RPN) generates candidate person regions, and the detection head performs region-wise classification and bounding box regression. Because the DVS input suppresses static background and emphasizes motion-induced human contours, the number of unnecessary proposals and the feature extraction cost can be reduced for low-latency edge inference.

Figure 6 presents representative person detection results obtained from the DVS-based Faster R-CNN-style method operating on an Exynos 7570 mobile application processor. The Exynos 7570 is a quad-core Cortex-A53 processor clocked at 1.4 GHz, representative of the low-power application processors found in entry-level smartphones and standalone AIoT edge devices. The detection results shown in Figure 6 demonstrate that the system successfully detects and localizes persons in DVS event-derived images, producing bounding boxes around moving subjects while suppressing static background information. The measured processing latency was 92 ms, which satisfies the approximately 100 ms latency target established in Section 2 for interactive person detection.



Figure 6. DVS-based person detection result in an indoor environment. The event-derived image emphasizes the moving person’s contour while suppressing static background information. Using the Faster R-CNN-style detector with a lightweight backbone, the proposed implementation achieves 92 ms processing latency on an Exynos 7570 mobile processor. The red rectangular boxes indicate the bounding boxes predicted by the detection network to localize the persons.

Table 3 summarizes the person detection configuration used in this work. The detailed detection algorithm, dataset construction, training procedure, and performance evaluation

are reported in [33]. Briefly, the referenced study evaluated the DVS-based detection pipeline in indoor AIoT/home surveillance scenarios containing human cases, non-human moving objects, and background dynamics. The reported detection performance showed recall greater than 95% and a false acceptance rate lower than 2% under the evaluated indoor conditions [33]. Therefore, in the present work, the person detection module is treated as a previously validated DVS-based detection pipeline, and the focus is placed on whether its processing latency satisfies the task-dependent latency requirement on a mobile edge processor.

Table 3. Summary of the person detection configuration used in this work.

Item	Description
Detection module	DVS-based event-derived image + Faster R-CNN-style detector
Target scenario	Indoor smart home/AIoT person detection
Input representation	Event-derived human contours
Dataset composition	Human cases and non-human moving objects
Evaluation protocol	Held-out test-set evaluation, as reported in [33]
Detection performance	Recall > 95%, false acceptance rate < 2%
Network optimization	11× speed-up for human detection
Measured latency	92 ms processing latency on Exynos 7570
Latency target	Approximately 100 ms for interactive AIoT feedback

Although higher-end mobile processors with dedicated NPUs or GPU acceleration could further reduce inference time, the Exynos 7570 result is significant because it demonstrates feasibility on a low-power quad-core ARM processor. This indicates that DVS-based person detection can satisfy the perceptual instantaneity threshold for interactive AIoT feedback even on resource-constrained edge hardware. Therefore, the person detection experiment should be understood as a system-level latency validation using a previously established DVS-based detection module, rather than as a new benchmark-level object detection study.

4.2. Gesture Recognition

Gesture recognition is a representative low-latency machine vision task because it is directly coupled to user action. In touchless human–machine interfaces, a user expects the system to respond immediately to hand motion, such as swipe, rotation, or directional movement. If the visual processing delay becomes large, the feedback no longer appears to be causally linked to the user’s gesture, resulting in degraded usability. Therefore, gesture recognition requires not only accurate motion interpretation but also a short and stable processing latency.

In this work, gesture recognition was implemented using the event-based 4-DoF motion estimation method proposed by Lee et al. [35]. The purpose of this subsection is not to propose a new gesture recognition algorithm, but to validate whether a previously established DVS-based motion estimation pipeline can satisfy the latency requirement of touchless interaction on a mobile application processor. The detailed algorithmic derivation and full experimental configuration are reported in [35], while the present work summarizes the necessary implementation details and evaluates the measured processing latency within the system-level latency framework defined in Section 3.4.

The processing pipeline consists of four stages. First, incoming DVS events are accumulated over a short temporal window to generate a compact event representation of recent hand motion. In our implementation, high-speed gesture processing uses recent events accumulated within a short window, typically within 5 ms, so that the representation reflects the most recent motion without waiting for a full image frame. Second, motion-induced event clusters corresponding to the moving hand contour are extracted while static background regions are naturally suppressed by the DVS sensing mechanism.

Third, the dominant 4-DoF motion parameters are estimated from the spatial and temporal distribution of the events. Fourth, the estimated motion parameters are mapped to gesture commands using decision thresholds on translation direction and rotation angle. In practical implementation, event representation and 4-DoF motion estimation are executed using a pipelined processing structure on the mobile application processor. One thread continuously updates the short-window event representation, while the motion estimation and gesture decision thread operates on the most recently available representation. Because the event representation window is short and the representation update is lightweight compared with the motion estimation computation, the effective processing latency is dominated mainly by the algorithmic motion estimation and decision stage rather than by event accumulation.

Figure 7 illustrates the DVS-based gesture recognition demonstration on a smartphone platform. When a user moves the hand in front of the sensor, the DVS does not acquire dense image frames. Instead, events are generated only along moving hand contours and high-contrast edge regions. These asynchronous events are accumulated over a short temporal interval and used to estimate the dominant hand motion. The estimated 4-DoF motion parameters are then mapped to gesture commands, such as directional movement or rotation-based interaction. Because the system directly analyzes motion-induced events, it does not require full-frame hand segmentation, dense optical flow computation, or repeated background subtraction. The computational advantage of the DVS-based method can be understood by comparing it with a conventional frame-based gesture recognition technique. In a CIS-based approach, each input frame contains both the hand and the static background. The algorithm must first process the full image, remove background regions, detect or segment the hand, estimate motion between frames, and finally classify the gesture. These steps require memory access and computation over the entire image area, even when only a small portion of the scene contains useful hand motion. In contrast, a DVS performs the first stage of motion filtering at the sensor level. Since static regions do not generate events, the downstream algorithm receives a sparse stream of motion-relevant data. Therefore, the number of pixels or features that must be processed is greatly reduced.

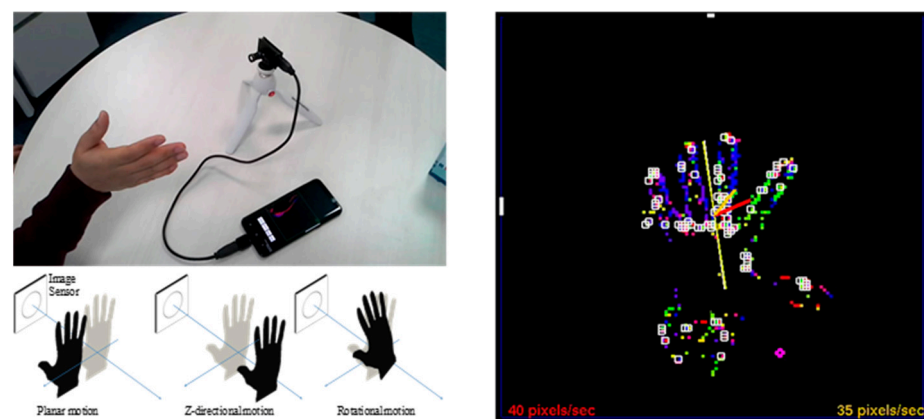


Figure 7. Event-driven gesture recognition using DVS-based 4-DoF motion estimation on a smartphone platform. The DVS generates sparse asynchronous events along moving hand contours, and the gesture recognition algorithm estimates dominant 4-DoF motion components from informative edge events. Because the method processes only motion-induced events rather than dense image frames, it reduces full-frame preprocessing, background subtraction, and dense optical flow computation. The measured processing latency is 20 ms on an Exynos 5422 mobile processor. In the visualization, the white boxes highlight the extracted motion-induced event clusters along the moving hand contour. The colored lines represent the estimated dominant 4-DoF motion vectors (such as translation and rotation directions), and the various colors of the events indicate their directions within the short accumulation window.

DVS also provides several practical advantages for gesture recognition. First, because the sensor reports events with very high temporal resolution, fast hand motion can be captured without waiting for the next frame boundary. Second, motion blur is largely avoided because events are generated at the moment of brightness change rather than after an exposure period. Third, static background clutter is suppressed, which improves robustness in indoor environments where furniture, walls, and illumination patterns may otherwise interfere with hand detection. Fourth, the amount of generated data naturally decreases when the hand is not moving, enabling low-power always-on operation for standby gesture interfaces.

Table 4 summarizes the implementation and evaluation configuration of the gesture recognition experiment. The detailed gesture dataset and full recognition performance evaluation are reported in [35,36]. In the present work, we use this previously validated 4-DoF motion estimation pipeline as a representative high-speed interaction task and report its processing latency on a mobile edge platform.

Table 4. Summary of the DVS-based gesture recognition configuration used in this work.

Item	Description
Gesture recognition module	DVS-based 4-DoF motion estimation and gesture decision
Motion parameters	2D translation and in-plane rotation
Input representation	Short-window event representation from recent hand motion events
Event representation window	Within 5 ms
Gesture decision method	Threshold-based mapping of 4-DoF motion parameters to gesture commands
Gesture classes	6 classes (up, down, left, right, clockwise, counterclockwise)
Dataset scale	6 subjects, 132 samples per class [36]
Recognition performance	97% success rate
Measured latency	20 ms on Exynos 5422
Latency target	10–100 ms for responsive gesture interaction

4.3. SLAM

SLAM is a key function for mobile and interactive machine vision systems because it simultaneously estimates the camera pose and reconstructs a map of the surrounding environment. In applications such as AR/VR, autonomous robots, drones, and mobile navigation, SLAM is not an offline perception task but a real-time closed-loop process. The estimated pose must be updated with low latency so that downstream modules, such as rendering, navigation, stabilization, or control, can respond to the current device motion rather than to a delayed camera frame. For this reason, SLAM requires a stricter latency budget than person detection or gesture recognition. In this work, the DVS-based SLAM implementation follows the low-latency interactive sensing framework reported by Park et al. [34]. In that work, the event-driven input reduces the amount of visual data that must be processed by the SLAM algorithm, while preserving the edge and motion information needed for camera tracking.

Figure 8 shows the overall structure of the DVS-based SLAM system. The DVS first generates asynchronous events from scene edges that move relative to the sensor. These events are accumulated over a short temporal window to form an event image or event-based edge map. Unlike a conventional CIS frame, this event image does not contain dense texture and static background intensity values. Instead, it mainly contains the contours and edge-like structures that have produced recent temporal contrast. These edge structures are well-suited for feature extraction and tracking because they correspond to high-gradient regions in the scene. The event image is then processed by the SLAM front-end to extract features, match them with projected map points, and estimate the current camera pose. The computational advantage of DVS-based SLAM can be understood by comparing it with

a conventional frame-based visual SLAM method. In a CIS-based system, every incoming image frame contains all pixels, including static background, low-texture surfaces, repeated patterns, and illumination components that may not contribute useful pose information. Feature extraction, descriptor computation, matching, and outlier rejection are therefore performed on dense images or on features detected from the full frame. In contrast, the DVS produces a sparse motion- and edge-centric representation. In SLAM, however, camera ego-motion causes static scene edges to generate events, and these motion-induced edge events provide useful information for pose tracking.

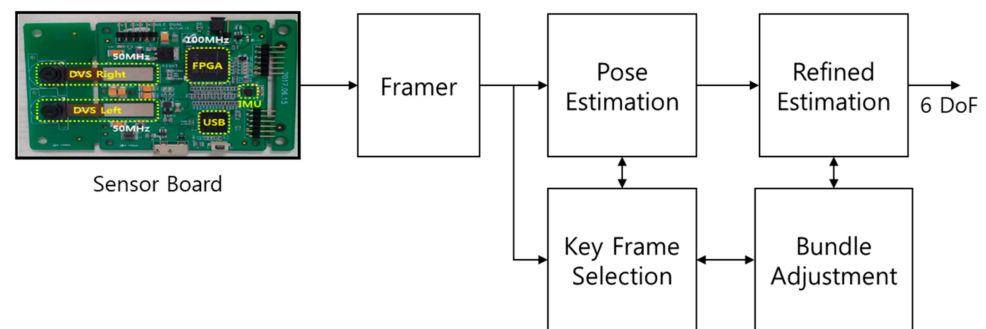


Figure 8. Structure of the DVS-based SLAM system. The DVS generates asynchronous events from temporal brightness changes caused by camera or scene motion. Events are accumulated over a short temporal window to form an event image, from which edge-like features are extracted for pose tracking and map update. Because the event image suppresses static background intensity and preserves motion-induced structures, the SLAM front-end can reduce full-frame feature extraction and operate with low latency on a mobile processor. The arrows indicate the data flow and processing sequence between the system modules.

Figure 9 illustrates the algorithm flow. The event stream is first converted into an event image or active event feature representation. Feature extraction is then performed on the event representation, followed by map-point projection, feature matching, and 3D–2D PnP (Perspective-n-Point)-based pose estimation. Keyframes are inserted when the number or spatial distribution of tracked landmarks becomes insufficient, and map points are refined through bundle adjustment. This processing structure allows the SLAM front-end to operate on active event-induced structures rather than dense intensity frames.

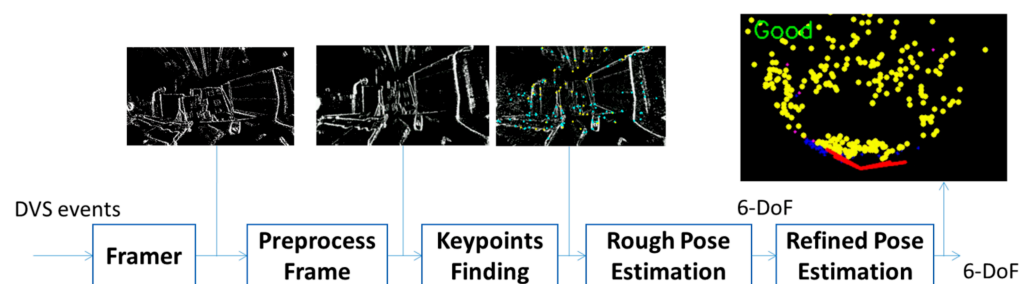


Figure 9. Algorithm flow of the DVS-based SLAM method. Incoming events are accumulated into an event image, followed by feature extraction, map-point projection, feature matching, and 3D–2D PnP pose estimation. Keyframes are inserted when the number or distribution of tracked landmarks becomes insufficient, and map points are refined through bundle adjustment. The implemented algorithm achieves 15.9 ms processing latency on a Snapdragon 845 mobile processor. In the visual output, the colored dots in the 2D frames (e.g., cyan and yellow) represent the extracted keypoints and matched features. In the 3D map view, the colored points (e.g., yellow) denote the reconstructed 3D map points, and the colored lines (e.g., red and blue) indicate the estimated pose direction and camera trajectory.

To evaluate localization accuracy and robustness, we constructed a self-collected office SLAM dataset under a 300-lux indoor illumination condition. The test trajectory had a total travel distance of 20 m, and the sensor platform was moved at a relatively high speed of 1–3 m/s. This condition was selected to evaluate a practical fast-motion scenario in which conventional frame-based SLAM may suffer from frame period delay, motion blur, and reduced tracking stability. The proposed DVS SLAM was compared with a CIS-based SLAM baseline implemented using ARCore on a mobile phone. Table 5 summarizes the localization accuracy and drift evaluation. Four metrics were used: last position error, AAPE (average absolute position error), relative drift error, and relative translation error. The last position error evaluates the final position deviation at the end of the trajectory, while AAPE evaluates the average absolute position error over the trajectory. Relative drift error and relative translation error normalize the corresponding position error values by the total travel distance, providing scale-independent measures of trajectory drift.

Table 5. Localization accuracy and drift comparison between DVS SLAM and CIS SLAM. The experiment was conducted in a 300-lux office environment over a 20 m trajectory with a motion speed of 1–3 m/s. The CIS SLAM baseline was measured using ARCore on a mobile phone.

Evaluation Metric	DVS SLAM (Average)	DVS SLAM (Median)	CIS SLAM (Average)	CIS SLAM (Median)
Last Position Error (cm)	3.2	2.7	130	125
AAPE (cm)	2.7	2.3	45.4	39.8
Relative Drift Error (%)	0.28	0.24	13.61	12.6
Relative Translation Error (%)	0.25	0.16	4.96	4.02

The DVS SLAM showed substantially lower localization error than the CIS-based ARCore SLAM baseline under the evaluated fast-motion office condition. The average last position error was 3.2 cm for DVS SLAM, whereas the CIS SLAM baseline showed 130 cm. Similarly, the average AAPE was 2.7 cm for DVS SLAM and 45.4 cm for CIS SLAM. The relative drift error was also significantly reduced from 13.61% in CIS SLAM to 0.28% in DVS SLAM, and the relative translation error was reduced from 4.96% to 0.25%. The median values show the same trend, indicating that the improvement was not limited to a single outlier case.

These results indicate that the DVS-based SLAM pipeline provides not only low processing latency but also practical localization accuracy under fast-motion indoor conditions. The improvement can be attributed to the event-driven sensing mechanism and sparse event-based processing. Unlike CIS-based SLAM, which depends on frame period image acquisition and can be degraded by motion blur during fast motion, the DVS reports asynchronous brightness changes with high temporal resolution. Therefore, active edge structures can be captured and processed with reduced temporal delay, allowing feature tracking and pose estimation to remain stable during high-speed motion.

The measured processing latency of the implemented DVS SLAM pipeline was 15.9 ms on a Snapdragon 845 mobile processor. This latency includes event representation, feature extraction, map-point projection, feature matching, PnP-based pose estimation, keyframe handling, and map refinement, following the processing latency definition in Section 3.4. The sensing latency, including pixel-level event generation and event readout, is analyzed separately in Section 3. Since the DVS sensing latency is maintained in the sub-millisecond to millisecond range, the practical latency bottleneck of the implemented SLAM system is primarily the processing pipeline. Therefore, the 15.9 ms result demonstrates that DVS SLAM can satisfy the sub-20 ms tracking latency target while maintaining low trajectory error under the evaluated fast-motion condition.

The latency and accuracy improvements should be interpreted as the result of sensing algorithm co-design rather than as the result of the sensor alone. At the sensing level, the DVS removes the fixed-frame period delay and suppresses static redundant information. At the processing level, the SLAM front-end operates on compact event representations and active features instead of dense frames. This combination reduces the number of pixels, features, and candidate matches that must be processed, thereby reducing memory access and computation on the mobile processor. The quantitative comparison with ARCore-based CIS SLAM confirms that the proposed DVS-based system provides practical advantages in both processing latency and localization stability for mobile edge SLAM applications.

5. Discussion and Conclusions

The three application demonstrations were evaluated on different mobile application processors: Exynos 7570 for person detection, Exynos 5422 for gesture recognition, and Snapdragon 845 for SLAM. Therefore, the measured latency values should not be interpreted as a direct processor-normalized comparison among the three tasks. Instead, the platform selection was application-driven: each processor was selected to reflect the expected deployment condition and computational budget of the corresponding target application.

Person detection is mainly intended for always-on AIoT and edge devices, such as smart home sensors, occupancy detectors, and low-cost surveillance nodes. In such devices, cost, power consumption, and hardware simplicity are often more important than maximum computing performance. Therefore, we selected Exynos 7570, a low-power quad-core Cortex-A53 mobile processor, as a representative resource-constrained edge platform. Demonstrating 92 ms processing latency on this processor shows that DVS-based person detection can satisfy the approximately 100 ms interaction target even under economical and low-performance edge hardware conditions.

Gesture recognition is primarily intended for mobile and smart device interfaces, where moderate computing capability is available and the system must provide responsive user interaction. For this reason, Exynos 5422 was selected as a mid-range mobile processor for evaluating the event-driven gesture recognition pipeline. The measured 20 ms processing latency demonstrates that the DVS-based 4-DoF gesture processing method can satisfy the tens-of-millisecond responsiveness required for touchless mobile interaction.

SLAM is intended for computationally demanding applications such as AR/VR, robotics, and mobile navigation, where real-time tracking, pose estimation, keyframe handling, and map refinement must be performed within a strict latency budget. In such applications, the use of a higher-performance mobile processor is practical and consistent with the hardware configuration of modern AR/VR devices. Therefore, Snapdragon 845 was selected as a representative high-end mobile processor for evaluating DVS-based SLAM. The measured 15.9 ms processing latency demonstrates that the proposed DVS-based SLAM pipeline can satisfy the sub-20 ms tracking latency target on a processor class suitable for advanced AR/VR and mobile tracking systems.

Accordingly, the reported latency results are evaluated against task-specific latency targets rather than directly against one another as shown in Table 6. The 92 ms person detection latency is compared with the approximately 100 ms AIoT response target, the 20 ms gesture recognition latency is compared with the 10–100 ms touchless interaction target, and the 15.9 ms SLAM latency is compared with the sub-20 ms closed-loop tracking target. This interpretation is consistent with the system-level objective of this work: to validate whether DVS-based sensing and task-specific event processing can satisfy practical latency requirements under realistic application-dependent processor constraints.

Table 6. Application-driven processor selection and interpretation of latency results.

Application	Processor	Reason for Selection	Latency Target	Measured Processing Latency
Person detection	Exynos 7570 (Low-cost edge/AIoT-class processor)	Person detection is expected to run on economical, low-power, and resource-constrained edge devices	~100 ms	92 ms
Gesture recognition	Exynos 5422 (Mid-range mobile processor)	Gesture interfaces are mainly used on mobile or smart device platforms requiring responsive user interaction	10–100 ms	20 ms
SLAM	Snapdragon 845 (High-end mobile/AR-VR-class processor)	SLAM is used in computationally demanding AR/VR, robotics, and mobile tracking systems that can employ higher-performance processors	<20 ms	15.9 ms

Although DVS-based machine vision has general limitations related to the absence of color and texture information, weak response to completely static objects, and the relative immaturity of event-based algorithms compared with conventional RGB-based vision, these issues did not materially degrade the three application results reported in this work. This is because the evaluated tasks were selected to exploit the motion-centric characteristics of DVS sensing. Person detection used motion-induced human contours rather than fine color or texture cues. Gesture recognition relied on dynamic hand motion rather than static appearance. SLAM used active edge structures and map/keyframe information rather than dense color images. Therefore, within the evaluated application scope, the lack of dense color or texture information, the reduced event rate for static objects, and the current immaturity of event-based algorithms could be mitigated through task-specific event representation, event filtering, keyframe usage, and hybrid sensing strategies when needed.

However, an important practical failure case was observed under flickering indoor illumination. When a flickering light source, such as a fluorescent lamp, is present in the scene, the DVS may respond to periodic brightness fluctuations and generate a large number of flicker-induced events. These events are not caused by object motion or camera motion, but by illumination modulation. As a result, they can interfere with event-based processing by increasing the event rate and by creating temporal patterns that are difficult to distinguish from true motion-induced events. This limitation is particularly critical for high-speed motion-based applications such as gesture recognition and SLAM. Gesture recognition estimates hand motion from short-window event clusters, and SLAM estimates camera pose from active event features generated by scene motion or camera motion. In flicker-dominated environments, flicker-induced events can overlap temporally with true motion events. Because both gesture recognition and SLAM rely on high-speed temporal changes, the system may not reliably distinguish illumination-induced events from motion-induced events. As a result, gesture decision-making can fail due to corrupted motion vectors, and SLAM can fail due to unstable feature extraction, incorrect feature matching, or loss of tracking. In contrast, person detection was less affected by flicker in the evaluated setting. Person detection is a relatively low-speed detection task compared with gesture recognition and SLAM, and the target motion is spatially localized and temporally slower than flicker-induced global event activity. Therefore, flicker components can be suppressed using a flicker filter that removes temporally periodic or spatially global event patterns before event image generation. After applying this filtering, the remaining event representation preserves the motion-induced human contour while suppressing flicker-

induced background activity. Therefore, flickering did not become a major issue for person detection in the evaluated indoor condition.

This analysis clarifies the practical boundary of the proposed system. The DVS-based approach is effective for motion-centric low-latency applications when the dominant event source is object motion or camera motion. However, flickering illumination can become a critical failure mode because it creates non-motion events that resemble meaningful temporal changes. Therefore, future work should include more systematic flicker-robust event filtering, adaptive threshold control, illumination-frequency suppression, and hybrid DVS–CIS sensing for environments containing fluorescent or other temporally modulated light sources.

In conclusion, this paper presented a system-level validation study of DVS-based low-latency machine vision. To clarify the source of latency reduction, the total latency was divided into sensing latency and processing latency. Sensing latency consists of pixel-level event generation latency and readout latency, which were separately defined and analyzed based on the DVS operating principle, temporal contrast thresholding, illumination-dependent pixel response, and event readout architecture. Processing latency consists of event representation latency and algorithm latency, and this was the experimentally measured latency in the three application demonstrations. The measured processing latencies were 92 ms for person detection on Exynos 7570, 20 ms for gesture recognition on Exynos 5422, and 15.9 ms for SLAM on Snapdragon 845. These results show that, when DVS event streams are represented and processed according to the latency requirement of each task, practical low-latency machine vision can be achieved on mobile edge platforms.

The results also show that neuromorphic sensing is not only a sensor-level improvement but also an algorithm-level enabler. By generating sparse, asynchronous, motion-selective events, the DVS reduces data transfer, memory access, and unnecessary computation before the machine vision algorithm begins. Therefore, latency-critical applications can be implemented with lower processor requirements than would be needed for dense frame-based sensing. Overall, DVS-based neuromorphic vision provides an efficient and practical pathway for responsive, low-power, and resource-aware machine vision in AIoT, robotics, mobile devices, and AR/VR systems.

Author Contributions: Conceptualization, writing—review and editing, P.K.J.P.; technical discussion, J.K. (Junseok Kim); project administration, J.K. (Juhyun Ko); supervision, Y.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Samsung Electronics.

Data Availability Statement: Third-party data. Restrictions apply to the availability of these data. The datasets presented in this article are not readily available because they are proprietary to Samsung Electronics and are subject to corporate confidentiality restrictions. Requests to access the datasets should be directed to the first author, Paul K. J. Park (paulpark@gachon.ac.kr).

Acknowledgments: The authors are thankful to Weiheng Liu, Qiang Wang, Hyunku Lee, Jiwon Im, Junhyuk Park, Bongki Son, Gyubeom Im, Yohan Roh, Yunjae Shu, Changwoo Shin, Hyunsurk Ryu, and Evgeny Soloveichik for providing valuable data and useful comments.

Conflicts of Interest: The authors declare that this study received funding from Samsung Electronics. The funder had the following involvement with the study: study design, data collection and analysis, preparation of the manuscript, and the decision to submit it for publication.

Abbreviations

The following abbreviations are used in this manuscript:

AER	Address Event Representation
AI	Artificial Intelligence
AIoT	Artificial Intelligence of Things
AR	Augmented Reality
CIS	CMOS Image Sensor
CMOS	Complementary Metal-Oxide Semiconductor
CNN	Convolutional Neural Network
DoF	Degree(s) of Freedom
DVS	Dynamic Vision Sensor
GPU	Graphics Processing Unit
HCI	Human–Computer Interaction
HDR	High Dynamic Range
IMU	Inertial Measurement Unit
IoT	Internet of Things
MIPI	Mobile Industry Processor Interface
MTP	Motion-to-Photon
NPU	Neural Processing Unit
PnP	Perspective-n-Point
RGB	Red, Green, Blue
RPN	Region Proposal Network
SLAM	Simultaneous Localization and Mapping
VGA	Video Graphics Array
VR	Virtual Reality

References

- Gallego, G.; Delbruck, T.; Orchard, G.; Bartolozzi, C.; Taba, B.; Censi, A.; Leutenegger, S.; Davison, A.; Conradt, J.; Daniilidis, K.; et al. Event-based vision: A survey. *IEEE Trans. Pattern Anal. Mach. Intell.* **2022**, *44*, 154–180. [[CrossRef](#)] [[PubMed](#)]
- Lichtsteiner, P.; Posch, C.; Delbruck, T. A 128×128 120 dB 15 μ s Latency Asynchronous Temporal Contrast Vision Sensor. *IEEE J. Solid-State Circuits* **2008**, *43*, 566–576.
- Cimarelli, C.; Millan-Romera, J.; Voos, H.; Sanchez-Lopez, J. Hardware, algorithms, and applications of the neuromorphic vision sensor: A review. *Sensors* **2025**, *25*, 6208. [[CrossRef](#)] [[PubMed](#)]
- Chakravarthi, B.; Verma, A.A.; Daniilidis, K.; Fermuller, C.; Yang, Y. Recent event camera innovations: A survey. In Proceedings of the European Conference on Computer Vision Workshops, Milan, Italy, 29 September–4 October 2024.
- Gehrig, D.; Scaramuzza, D. Low-latency automotive vision with event cameras. *Nature* **2024**, *629*, 1034–1040. [[CrossRef](#)] [[PubMed](#)]
- Verma, A.; Vaghela, A.; Chakravarthi, B.; Chanda, K.; Yang, Y. Event-based Graph Representation with Spatial and Motion Vectors for Asynchronous Object Detection. In Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, Tucson, AZ, USA, 6–10 March 2026.
- Wang, S.; Wang, X.; Jin, L.; Jiang, B.; Zhu, L.; Chen, L.; Tian, Y.; Luo, B. Towards Low-Latency Event Stream-based Visual Object Tracking: A Slow-Fast Approach. *arXiv* **2025**, arXiv:2505.12903.
- Bhattacharyya, P.; Mitton, J.; Page, R.; Morgan, O.; Menzies, B.; Homewood, G.; Jacobs, K.; Baesso, P.; Trickett, D.; Mair, C.; et al. Helios: An Extremely Low Power Event-Based Gesture Recognition for Always-on Smart Eyewear. In Proceedings of the European Conference on Computer Vision Workshops, Milan, Italy, 29 September–4 October 2024.
- Niu, J.; Zhong, S.; Lu, X.; Shen, S.; Gallego, G.; Zhou, Y. ESVO2: Direct Visual-Inertial Odometry with Stereo Event Cameras. *IEEE Trans. Robot.* **2025**, *41*, 2164–2183. [[CrossRef](#)]
- Miller, R. Response time in man-computer conversational transactions. In Proceedings of the AFIPS Fall Joint Computer Conference, San Francisco, CA, USA, 9–11 December 1968.
- Nielsen, J. *Usability Engineering*; Academic Press: Boston, MA, USA, 1993.
- Attig, C.; Rauh, N.; Franke, T.; Krems, J. System Latency Guidelines Then and Now—Is Zero Latency Really Considered Necessary? In Proceedings of the International Conference on Engineering Psychology and Cognitive Ergonomics, Vancouver, BC, Canada, 9–14 July 2017.

13. Thorpe, S.; Fize, D.; Marlot, C. Speed of processing in the human visual system. *Nature* **1996**, *381*, 520–522. [\[CrossRef\]](#) [\[PubMed\]](#)
14. Shneiderman, B. Response time and display rate in human performance with computers. *ACM Comput. Surv.* **1984**, *16*, 265–285. [\[CrossRef\]](#)
15. VanRullen, R.; Thorpe, S.J. The time course of visual processing: From early perception to decision-making. *J. Cogn. Neurosci.* **2001**, *13*, 454–461. [\[CrossRef\]](#) [\[PubMed\]](#)
16. Cazzato, D.; Bono, F. An application-driven survey on event-based neuromorphic computer vision. *Information* **2024**, *15*, 472. [\[CrossRef\]](#)
17. Li, E.; Zeng, L.; Zhou, Z.; Chen, X. Edge AI: On-demand accelerating deep neural network inference via edge computing. *IEEE Trans. Wirel. Commun.* **2020**, *19*, 447–457. [\[CrossRef\]](#)
18. Mitra, S.; Acharya, T. Gesture recognition: A survey. *IEEE Trans. Syst. Man Cybern. Part C Appl. Rev.* **2007**, *37*, 311–324. [\[CrossRef\]](#)
19. Hirsh, I.; Sherrick, C. Perceived Order in Different Sense Modalities. *J. Exp. Psychol.* **1961**, *62*, 423–432. [\[CrossRef\]](#) [\[PubMed\]](#)
20. Köpüklü, O.; Gunduz, A.; Kose, N.; Rigoll, G. Real-time hand gesture detection and classification using convolutional neural networks. In Proceedings of the IEEE International Conference on Automatic Face and Gesture Recognition, Lille, France, 14–18 May 2019.
21. Molchanov, P.; Yang, X.; Gupta, S.; Kim, K.; Tyree, S.; Kautz, J. Online detection and classification of dynamic hand gestures with recurrent 3D convolutional neural networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016.
22. Amir, A.; Taba, B.; Berg, D.; Melano, T.; McKinstry, J.; Di Nolfo, C.; Nayak, T.; Andreopoulos, A.; Garreau, G.; Mendoza, M.; et al. A Low Power, Fully Event-Based Gesture Recognition System. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017.
23. Stanney, K.; Lawson, B.D.; Rokers, B.; Dennison, M.; Fidopiastis, C.; Stoffregen, T.; Weech, S.; Fulvio, J.M. Identifying causes of and solutions for cybersickness in immersive technology: Reformulation of a research and development agenda. *Int. J. Hum.–Comput. Interact.* **2020**, *36*, 1783–1803. [\[CrossRef\]](#)
24. Stauffert, J.; Niebling, F.; Latoschik, M. Latency and cybersickness: Impact, causes, and measures. A review. *Front. Virtual Real.* **2020**, *1*, 582204. [\[CrossRef\]](#)
25. Elbamby, M.S.; Perfecto, C.; Bennis, M.; Doppler, K. Toward low-latency and ultra-reliable virtual reality. *IEEE Netw.* **2018**, *32*, 78–84. [\[CrossRef\]](#)
26. Mur-Artal, R.; Montiel, J.; Tardós, J. ORB-SLAM: A versatile and accurate monocular SLAM system. *IEEE Trans. Robot.* **2015**, *31*, 1147–1163. [\[CrossRef\]](#)
27. Elamin, A.; El-Rabbany, A.; Jacob, S. Event-Based Visual/Inertial Odometry for UAV Indoor Navigation. *Sensors* **2025**, *25*, 61. [\[CrossRef\]](#)
28. Cadena, C.; Carlone, L.; Carrillo, H.; Latif, Y.; Scaramuzza, D.; Neira, J.; Reid, I.; Leonard, J. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Trans. Robot.* **2016**, *32*, 1309–1332. [\[CrossRef\]](#)
29. Delbruck, T.; Mead, C. Adaptive Photoreceptor with Wide Dynamic Range. In Proceedings of the IEEE International Symposium on Circuits and Systems, London, UK, 30 May–2 June 1994.
30. Joubert, D.; Marcireau, A.; Ralph, N.; Jolley, A.; van Schaik, A.; Cohen, G. Event Camera Simulator Improvements via Characterized Parameters. *Front. Neurosci.* **2021**, *15*, 702765. [\[CrossRef\]](#) [\[PubMed\]](#)
31. Son, B.; Suh, Y.; Kim, S.; Jung, H.; Kim, J.; Shin, C.; Park, K.; Lee, K.; Park, J.; Woo, J.; et al. A 640 × 480 dynamic vision sensor with a 9 μm pixel and 300 Meps address-event representation. In Proceedings of the International Solid-State Circuits Conference, San Francisco, CA, USA, 5–9 February 2017.
32. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster R-CNN: Towards real-time object detection with region proposal networks. *IEEE Trans. Pattern Anal. Mach. Intell.* **2017**, *39*, 1137–1149. [\[CrossRef\]](#) [\[PubMed\]](#)
33. Park, P.; Kim, J.; Ko, J.; Chang, Y. Event-Based Machine Vision for Edge AI Computing. *Sensors* **2026**, *26*, 935. [\[CrossRef\]](#) [\[PubMed\]](#)
34. Park, P.; Kim, J.; Shin, C.; Lee, H.; Liu, W.; Wang, Q.; Roh, Y.; Kim, J.; Ater, Y.; Soloveichik, E.; et al. Low-latency interactive sensing for machine vision. In Proceedings of the IEEE International Electron Devices Meeting, San Francisco, CA, USA, 7–11 December 2019.
35. Lee, J.; Lee, K.; Ryu, H.; Park, P.; Shin, C.; Woo, J.; Kim, J. Real-time motion estimation based on event-based vision sensor. In Proceedings of the IEEE International Conference on Image Processing, Paris, France, 27–30 October 2014.
36. Lee, J.; Delbruck, T.; Pfeiffer, M.; Park, P.; Shin, C.; Ryu, H.; Kang, B. Real-time gesture interface based on event-driven processing from stereo silicon retinas. *IEEE Trans. Neural Netw. Learn. Syst.* **2014**, *25*, 2250–2263. [\[CrossRef\]](#) [\[PubMed\]](#)

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.