# E-Cube: Event Enhanced Efficient Video Streaming for Drones

Jingao Xu[♥♦], Longfei Shangguan[♠], Danyang Li[♥], Yunhao Liu[♥], Zheng Yang[♥✉]
♥Tsinghua University, ♠University of Pittsburgh, ♦The University of Hong Kong
{xujingao13,lidanyang1919,yunhaoliu,hmilyyz}@gmail.com,longfei@pitt.edu

## Abstract

This paper presents E-Cube, a framework designed to enhance the efficiency of mobile video streaming systems. Our core insight is that *content correlations* among frames, which are critical for video streaming but challenging to extract in dynamic scenes, are often already available as intermediate outputs from other mobile computing subsystems. By adopting a cross-subsystem design, E-Cube repurposes these intermediate results obtained from event cameras to reduce computation, energy consumption, and bandwidth usage of existing video streaming systems, while simultaneously improving video quality. We implement E-Cube on a drone-embedded chip through software-hardware co-design, and plugged E-Cube into three prevalent drone video streaming systems for industrial inspection. Evaluations in one of the world's largest oil fields and on public datasets demonstrate E-Cube can save over 35% network bandwidth overhead and reduce drone video streaming energy consumption by over 12% for H.265 and AV1, and 47% for advanced H.266, all while achieving improved video quality.

***CCS Concepts:*** • **Computer systems organization → Embedded and cyber-physical systems**; **Real-time systems**.

**ACM Reference Format:**
Jingao Xu, Longfei Shangguan, Danyang Li, Yunhao Liu, Zheng Yang. 2026. E-Cube: Event Enhanced Efficient Video Streaming for Drones. In *European Conference on Computer Systems (EUROSYS '26), April 27–30, 2026, Edinburgh, Scotland Uk.* ACM, New York, NY, USA, 20 pages. https://doi.org/10.1145/3767295.3769375

## 1 Introduction

Due to their cost effectiveness, high efficiency, and versatility, consumer drones have become prevalent in various industry sectors, such as industrial inspection[8, 54], aerial monitoring[12, 27], and network relay[33, 48]. **Video streaming** serves as a fundamental component of drone-based applications and mobile systems more broadly – mobile devices capture video data through onboard cameras as they traverse areas of interest, then transmit this data wirelessly to remote servers for real-time analysis and decision-making.

**Table 1.** Overall Performance Comparison. E-Cube leads to a reduction in BitRate and power consumption, while achieving better PSNR (Peak Signal-to-Noise Ratio).

| Video Streaming Solution* | | BitRate (Mb/s) | Power† Cons. (W) | PSNR (dB) |
|---|---|---|---|---|
| H.265[50] | w/o E-Cube | 12.6 | 2.5 | 36.5 |
| | **w/ E-Cube** | **7.2 (42.9%↓)** | **2.2 (12.0%↓)** | **39.2** |
| AV1[16] | w/o E-Cube | 9.7 | 9.8 | 38.1 |
| | **w/ E-Cube** | **6.1 (37.1%↓)** | **6.9 (29.6%↓)** | **39.4** |
| H.266[14] | w/o E-Cube | 5.9 | 17.4 | 39.6 |
| | **w/ E-Cube** | **3.8 (35.6%↓)** | **9.1 (47.7%↓)** | **40.1** |
| Swift[17] (DNN-based System) | | 3.7 | 45.2 | 40.3 |

* Evaluated in dynamic scenarios with 2K 60FPS video feeds.
† Event data capturing, preprocessing, and E-Cube's computational power consumption are already included in the analysis.

Video streaming systems primarily rely on coding-based standards like H.26x[14, 50, 52], VP9[38], and AV1[16], consisting of (*i*) a video coding layer (VCL) that compress raw video frames through frame/block *division* and *prediction*; and (*ii*) a network abstraction layer (NAL) that prioritizes video packets' transmission (§2). While these coding-based streaming systems have demonstrated their efficacy in various mobile applications, our pilot study on drone-based industrial inspection at an oil-field (Fig. 1) reveals these solutions are less suitable for complex, dynamic industrial scenarios. We summarize our findings below:

(*i*) **The low frame prediction accuracy impairs video codec performance.** In industrial environments, the low inter-frame similarity impairs *inter-frame prediction* accuracy, resulting in an increased compression loss and a drop in video quality. To handle this issue, the *frame/block division* module has to incorporate more less-compressed frames (i.e., I-frames) to maintain video quality, which in turn raises the bitrate and further strains network bandwidth (§3.1).

(*ii*) **The intensive block-matching operations exacerbate energy consumption.** Inter-frame prediction depends on resource-intensive block-matching, where a codec finds similar blocks between frames to seek temporal redundancy. With notable content shifts in industrial scenarios, block-matching encounters a wider search area and more operations. Such heightened complexity increases computational demands, thereby heavily draining the drone's battery (§3.2).

✉ Zheng Yang is the corresponding author.


*EUROSYS '26, April 27–30, 2026, Edinburgh, Scotland Uk*

ACM ISBN 979-8-4007-2212-7/26/04
https://doi.org/10.1145/3767295.3769375

(*iii*) **The foreground-background blending undermines RoI-priority streaming.** Video codecs segment critical regions of interest (RoIs) in VCL, and prioritize their transmission in NAL. Yet, in industrial settings, video content evolves as a complex overlay of drone motion and scene dynamics, making it hard to separate foreground and background elements for clear RoI segmentation and priority transmission. For industrial inspection tasks, where detecting foreground intruders is vital, the resulting videos with unclear RoIs can impede effective threat detection and response (§3.3).

In summary, these system issues impair core *frame and block division* and *inter-frame prediction* modules, preventing them from establishing ***content correlations*** among adjacent video frames, which fundamentally degrades video quality, compression efficiency, and energy performance.

**Key Insight from Holistic Mobile System Perspective.** While obtaining content correlations poses significant challenges for video streaming, mobile devices routinely perform such computations as a fundamental capability in most vision tasks. For instance, visual odometry (VO) and visual-inertial odometry (VIO) modules essential for drone/robot control and navigation rely on calculating pixel-level associations (i.e., optical flow) and motion features, which are exactly the intermediate results that constitute content correlations[54, 55]. Building upon this observation, we aim to transcend traditional *video streaming* boundaries and approach this problem from a comprehensive *mobile system* perspective, exploring whether those intermediate results generated by *computing* subsystems can directly provide the accurate content correlations for efficient video streaming.

**E-Cube: an Event Enhanced Efficient video streaming solution.** This paper represents among the first attempts to translate such cross-subsystem design insights into a practical solution. Specifically, we explore a new opportunity arising from a novel vision modality - event camera - that has gained prominence in mobile vision tasks. Unlike frame cameras with fixed capture intervals (e.g., 10-60ms), event cameras asynchronously record pixel-level scene changes into an event stream at ultra-high temporal resolution (in $\mu s$) with minimal power (in mW)[21], making them have gained widespread adoption on drones for high-speed object tracking[19, 55], VO/VIO[46, 63], etc.. Aligned with our insight, the rich temporal data of event cameras can be leveraged to establish *content correlations* among adjacent video frames with minimal power overhead, which can then be exploited to enhance video streaming (§4).

E-Cube is a plug-in system solution that enables video streaming subsystems to reuse intermediate results from computing subsystems, achieving *higher video compression ratios*, *improved video quality*, and *reduced energy consumption*. During drone movement, E-Cube first leverages the event streams to estimate inter-frame motion vectors that

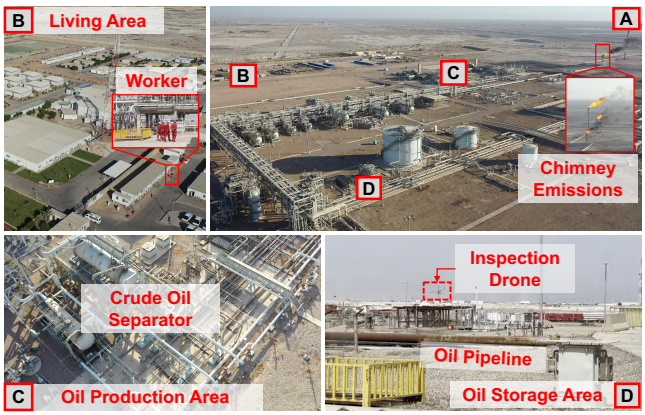



**Figure 1.** E-Cube's adoption for industrial inspection in one of the world's largest oil fields (>170$km^2$). Drones stream videos to back-end servers for real-time analysis.

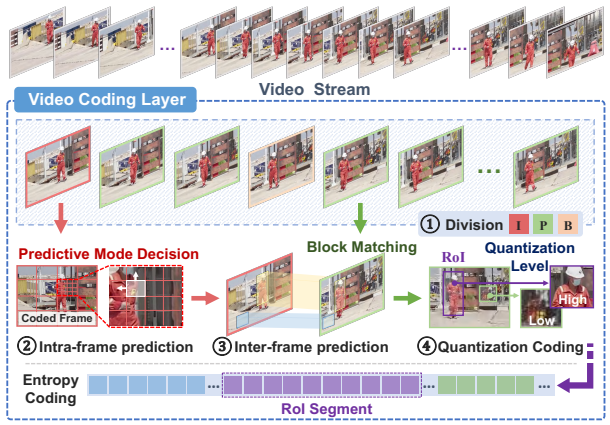
are then temporally connected and spatially merged to establish *content correlations* among adjacent frames (§5.1). Subsequently, E-Cube utilizes these correlations to enable efficient inter-frame prediction (§5.2), optimal intra-frame block division (§5.3), and RoI-prioritized video streaming (§5.4). These three modules work hand-in-hand to achieve superior video streaming performance with improved PSNR and reduced bandwidth and power consumption.

To fully realize such a cross-subsystem mobile architecture, we have implemented E-Cube on a commercial embedded Xilinx Zynq-7020 chip through software-hardware co-design. We design dedicated FPGA logic circuits to parallelize pixel-wise event processing, accelerating execution and enabling seamless integration of content correlations from computing subsystems into video streaming ones.

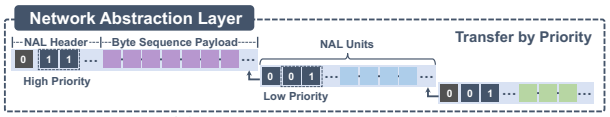
We have integrated E-Cube into three prevalent drone video streaming systems across different hardware platforms, including a DJI-adopted H.265 standard implemented on a Zynq-7020 chip[1] using FPGA, and H.266 and AV1 standards implemented on an Nvidia Jetson TX2[4] using C++. We further deploy these three video streaming systems on industrial drones and compare their performance in the presence and absence of E-Cube based on the oil field traces (Fig. 1) and a gold-standard drone video dataset.

Table 1 presents the overall performance gain brought by E-Cube. As seen, E-Cube's integration could save over 35% network bandwidth overhead and reduce energy consumption by over 12% (for hardware-implemented H.265) and 29%-47% (for software-implemented AV1 and H.266), all while achieving better video quality. Additionally, benefit from E-Cube, the coding-based streaming solution (i.e., H.266) could achieve performance comparable to a current advanced learning-based solution (i.e., Swift), while consuming nearly one-fifth the power on drones.

In summary, this paper makes the following contributions.




**(a)** Video Coding Layer




**(b)** Network Abstraction Layer

**Figure 2.** Workflow of prevalent mobile video streaming systems (e.g., H.26x, VP9, AV1).

• We conduct a systematic examination of existing video streaming frameworks and identify their fundamental limitations in complex industrial scenarios.
• We make an innovative architectural attempt in modern mobile system design. Instead of pushing the envelope of computer vision algorithms for video streaming, we bridge the traditionally isolated streaming and computing subsystems in mobile devices, enabling *computing* intermediates to be effectively utilized by *video streaming* components.
• We propose a suite of solutions that leverages the rich temporal information of event data to enhance video streaming.
• We deploy three E-Cube-integrated video streaming systems on drones. Extensive evaluations demonstrate E-Cube's superior performance.

## 2 Video Streaming Primer

While specific technical details may vary, prevalent video streaming systems share a video coding layer (VCL) plus a network abstraction layer (NAL) architecture. The workflow is illustrated in Fig. 2 and functional details can be found in Appendix-A. Briefly, VCL contains:

① **Frame and Block division**. Each video frame is categorized as either an I-frame, a P-frame, or a B-frame. The video frame is further divided into blocks with different pixel sizes (e.g., 8×8, 16×16, 64×64) and type (i.e., I, P, B), based on frame content dynamics such as motions and textures.

② **Intra-frame prediction** minimizes spatial redundancy in I-frames (or blocks) by applying compression techniques (e.g., JPEG[34]) and encoding the prediction residuals.

③ **Inter-frame prediction** computes motion vectors to identify pixel movements from previous I-frames to P- or B-frames as consecutive frames contain similar content. It then estimates pixel values in these frames using the I-frame data. Only the differences between these estimates and actual values (a.k.a., residuals) will be encoded. Typically, this process accounts for 60-70% of the total compression ratio[17].

④ **Quantization and entropy coding** reduce visual and coding redundancy by replacing fine data with coarser quantified values. Different areas within a frame may be assigned varying quantization parameters to retain specific details.

NAL then prioritizes data delivery based on network conditions and application requirements by: $(i)$ segmenting video stream into datagrams; $(ii)$ adding metadata to datagram, describing their content and importance. Finally, NAL chooses to transmit or store datagrams, as illustrated in Fig. 2b.

## 3 Lessons Learned From Pilot Study

To assess the effectiveness of the current video streaming solutions in complex industrial situations, we conducted a pilot study over three months from May to August 2023 at the Middle East Majnoon oil field. In our pilot study, we examined two drone-based video streaming solutions for industrial inspection: AV1[16], known for its versatility, and H.266[14], representing the state-of-the-art in commercial mobile streaming technology. The drone roams the oil field regularly and transmits the video stream back to the server.

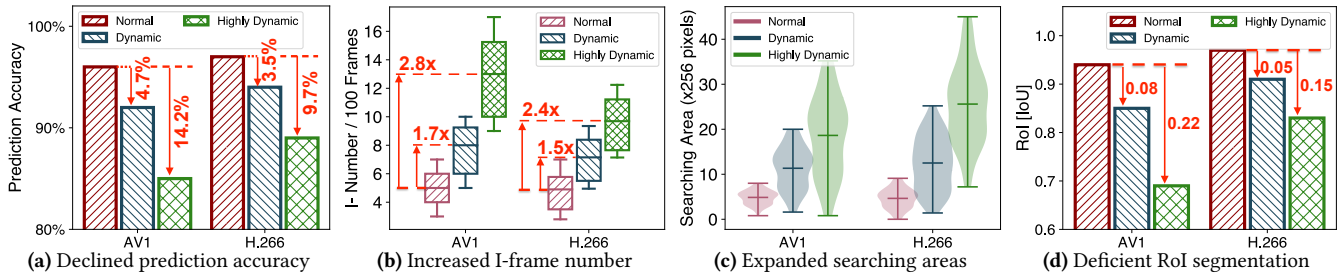
**Root cause of scene dynamics**. We found that the drones experience a variety of scene dynamics within their field of view. This can be attributed to two main reasons.

• Firstly, the drone's limited battery (i.e., DJI Mavic with a 45Wh battery only sustains a 30min flight) necessitates it to fly at a high speed of up to $10m/s$ to cover more ground. This high-speed results in significant changes in the video content across frames, with new content constantly emerging.
• Secondly, industrial settings are inherently complex and dynamic. Spatial-wise, tightly arranged facilities lead to intricate textures in images (Fig. 1c). Temporal-wise, activities like frequent worker movement, regular traffic (Fig. 1b), and chimney emissions (Fig. 1a) all add to scene dynamics.

We found these scene dynamics hinder the efficient application of AV1 and H.266, leading to inferior video encoding, higher energy consumption, and low adaptability for RoI-based applications. To better illustrate these issues, we compare AV1 and H.266 in three levels of scene dynamics, namely, normal (**N**), dynamic (**D**), and highly dynamic (**H**). Detailed explanations of these modes can be found in §7.1.

### 3.1 C1: Degraded Video Codec Performance

Current video codecs assume a strong temporal correlation exists between successive frames and leverage such correlation for content prediction and compression. However, our pilot study showed that in industrial scenarios, the significant variation in consecutive frames due to scene dynamics challenges this assumption. The large content gap leads to inaccurate motion vector estimation, resulting in increased inter-frame prediction residuals and degraded video quality.




**(a)** Declined prediction accuracy **(b)** Increased I-frame number **(c)** Expanded searching areas **(d)** Deficient RoI segmentation

**Figure 3.** The performance of two drone video streaming systems in different scene dynamics settings.

**Table 2.** Degraded performance with scene dynamics

| System | Scene* | BitRate (Mb/s) | Power Cons. (W) | PSNR (dB) [Overall] | PSNR (dB) [RoI] |
|---|---|---|---|---|---|
| AV1 | N | 6.2 | 6.6 | 41.9 | 42.7 |
| | D | 9.6 (**54.8%**↑) | 10.4 (**57.8%**↑) | 39.2 | 39.6 |
| | H | 13.7 (**120.9%**↑) | 14.7 (**122.7%**↑) | 37.1 | 38.4 |
| H.266 | N | 3.2 | 11.2 | 42.2 | 43.5 |
| | D | 5.4 (**68.7%**↑) | 14.8 (**32.1%**↑) | 39.7 | 40.9 |
| | H | 6.7 (**109.3%**↑) | 17.4 (**55.3%**↑) | 38.5 | 40.1 |

* N: Normal; D: Dynamic; H: Highly Dynamic. Performance is compared with the normal scene (↑).

Fig. 3a shows the accuracy of inter-frame prediction for H.265 and AV1 across three scene dynamics settings, focusing on the ratio of correctly predicted pixel locations to total predictions. We can see the accuracy drops by 14.2% for AV1 and 9.7% for H.266 when switching from the normal to highly dynamic scenes. The decline in inter-frame prediction accuracy leads to a notable reduction in video quality PSNR, over 3dB for both codecs, as detailed in column 5 of Table 2.

Moreover, the large content gap also pushes the video codec to adopt a shorter group-of-picture (GoP) length (i.e., retaining more less-compressed I-frames) in an attempt to maintain coding quality. As Fig. 3b illustrates, the average count of I-frames per 100 frames over doubles in highly dynamic scenarios for both AV1 (from 4.7 to 13.2) and H.266 (from 4.5 to 10.6). This adjustment causes a 120.9% and 109.3% increase in video data transmission (bit-rate), as shown in column 3 of Table 2.

### 3.2 C2: Excessive Resource Overhead

To determine the motion vector for each block in the current frame, the block-matching module adopts various searching strategies (e.g., diamond search[65], and TZ-search[41]) to locate the matching I-block on the latest I-frame (Appendix-A). However, within industrial scenarios, the growing content gaps between successive frames require the block-matching module to enlarge its search area, resulting in a higher number of matching operations. The associated computation overhead grows dramatically in more advanced codecs (e.g., AV1, H.266) since these codecs allow for dynamic block size adjustment (e.g., from 128×128 to 4×4) that add the necessity to go through multiple block sizes in a search.

Fig. 3c shows the averaged search area for a 16×16 block in different levels of scene dynamics settings. The rise in scene dynamics results in an almost threefold increase in the averaged search area (measured in 256 pixels) for each block, escalating from 5.5 to 20.8 for AV1 and from 4.6 to 25.9 for H.266. These intensive computation overheads lead to a remarkable increase in power consumption, as documented in Table 2. As the scene transitions from normal to highly dynamic, the power consumption increases from 6.6W and 11.2W to 14.7W and 17.4W for AV1 and H.266, respectively. Using a representative around 100Wh onboard battery (e.g., DJI Inspire series[5]), our measurements imply that in highly dynamic scenes the encoder alone draws 14.7-17.4W, i.e., around 15-17% of the pack per hour. The energy budget reduces available propulsion power and shortens the drone's flight endurance under otherwise identical conditions.

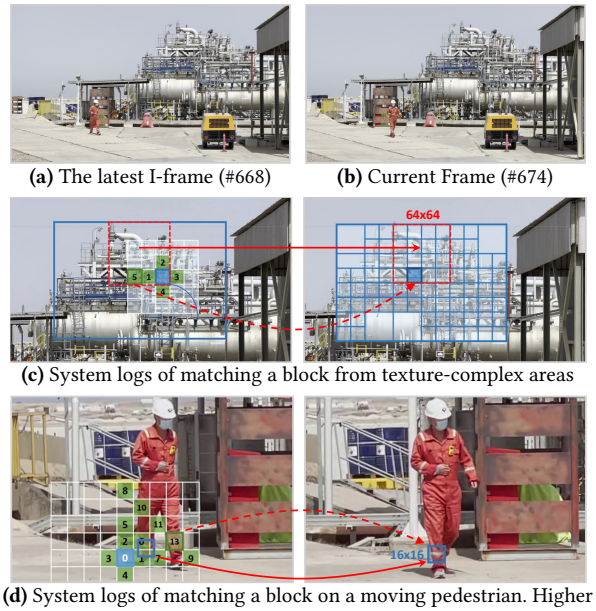
### 3.3 C3: Deficient RoI Prioritization

The rapid speed of drones and the highly dynamic scenes encountered in complex industrial scenarios often result in sudden changes in video content from one frame to another. Consequently, distinguishing between the foreground objects such as intruding pedestrians and vehicles, and the background in each frame presents a significant challenge. This difficulty can impede the extraction of RoI in continuous video frames. As demonstrated in Fig. 3d, the intersection-over-union (IoU) accuracy for both AV1 and H.266 codecs drops by 0.22 and 0.15, falling below 0.7 and 0.8.

Inaccurate RoI extraction leads to two major issues. First, the quantization coding in the VCL would fail to ensure high-quality encoding of important regions. Second, the NAL would be unable to tag packets from critical regions with metadata for priority transmission. These drawbacks hinder the delivery of high-quality RoI video to back-end applications. Column 6 in Table 2 documents the RoI-specific streaming quality, where the PSNR decreases by 4.3dB for AV1 and 3.4dB for H.266 compared to normal scenarios.

## 4 E-Cube: The Design Space

Based on the lessons learned, we find that video streaming performance degradation in dynamic scenes stems from the weak ability to compute *temporal content correlations*. Current codecs address this by employing heavy CV algorithms




**(a)** The latest I-frame (#668) **(b)** Current Frame (#674)

**(c)** System logs of matching a block from texture-complex areas

**(d)** System logs of matching a block on a moving pedestrian. Higher resolution details are provided in Fig. 17 in Appendix-A

**Figure 4.** Illustrations of the temporal correlation matters.

for inter-frame and intra-frame prediction, inevitably creating a trade-off between accuracy and efficiency[14, 17, 43, 60].

To address these challenges, we question whether temporal content correlation computation must be performed by video streaming subsystems in isolation. Can we instead reuse intermediate results from computing tasks such as VO/VIO, SLAM, and optical flow estimation? An ideal way is to obtain the *temporal content correlations* between the current and reference frames (i.e., the latest I-frame) before coding the current frame, as elaborated below.

### 4.1 Temporal Correlation Matters

We illustrate the advantage of content correlation-guided design by comparing it with standard AV1. In these examples, AV1 attempts to match a block (denoted as □) from the current frame #674 (see Fig. 4b) to a block in the previous I-frame #668 (see Fig. 4a) for inter-frame prediction. We logged AV1's block search process. The blocks being searched are denoted as ( $i$ ), with the variable $i$ indicating search order in Fig. 4c.

• **Frame and Block Division**. The video codec starts by dividing the current frame into blocks based solely on textures. As shown in Fig. 4c, regions with complex textures like interlacing pipes are divided into many small blocks to preserve the structure details, while tank surface and sky areas with fewer textures are grouped into larger blocks.

However, due to the complex textures within each small block, locating the optimal match block in the reference frame still demands 3-5 searches. The substantial number of small blocks on the current frame, combined with multiple searches per block, will considerably elevate computational overhead. Worsening the situation, the high texture complexity of the current video content prompts the encoder to add more I-frames, anticipating scene changes[16, 50]. In our experiments, H.266 even directly treats frame #674 as a new, less-compressed I-frame, increasing bandwidth overheads.

In contrast, if the video codec analyzes the temporal relationship between the content of frame #668 and #674 before the division process, it would identify that the pixel content in these texture-rich regions is strikingly similar between these two frames, as illustrated by Fig. 4c. Consequently, the video codec ($i$) would not recognize #674 as a new I-frame, lowering bandwidth usage; and ($ii$) could opt for a larger block size in this area (e.g., the red box □ with 64×64 pixels shown in the figure), achieving a reduction in match operations without compromising prediction accuracy.

• **Inter-frame Prediction**. Next, we delve into the inter-frame prediction process. Given a block $b$ on frame #674, to find its optimal match block on frame #668, the video codec begins by examining the block that aligns with the same pixel location as block $b$ and then expands its search outward to nearby block candidates, as illustrated in Fig. 4d. To expedite the search, recent codecs prioritize blocks likely to match with block $b$, drawing on motion information obtained from a preceding frame, e.g., frame #673.

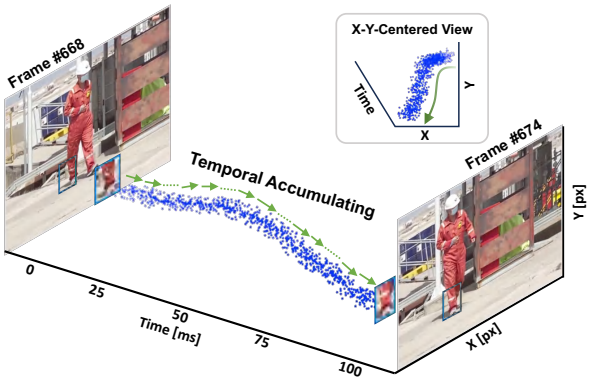
However, due to scene dynamics and complexity, motion information obtained from frame #673 fails to precisely predict motion in the current frame #674. Fig. 4d illustrates this issue: the codec prioritizes block search in the upper right area based on the motion information obtained from frame #673. However, this data inaccurately represents current human movement, leading the codec to mistakenly associate right leg motion with the left leg (as denoted by ⇢in the figure). Consequently, the codec has to search additional blocks (13 in total) before locating the optimal match block.

Conversely, if codecs can profile the temporal content correlations between the reference frame #668 and the current frame #674, rather than inferring it from the previous frame #673, it could identify the human motion pattern (denoted as → in Fig. 4d), leading to a reduction in block search.

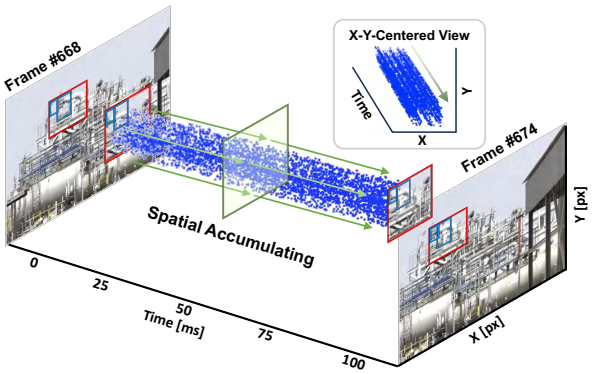
### 4.2 An Opportunity from Event Data

Obtaining content correlations typically involves computing optical flow between two frames, which includes resource-intensive steps like extracting features and estimating photometric errors[43, 60]. Integrating these processes represents a computationally demanding task for both *video streaming* and *computing* subsystems, which poses challenges for resource-limited mobile devices.

Recently, however, a newly emerging vision sensor, namely, *event camera*, may break this stalemate. As shown in Fig. 19 (Appendix-B.1), unlike frame cameras, event cameras asynchronously capture per-pixel brightness changes at *microsecond* resolution, while consuming very low power at *microwatts*[21]. Each time a pixel senses a brightness change, it




**(a)** Temporal accumulating of event stream




**(b)** Spatial merging of event stream

**Figure 5.** Illustrations of event-based content temporal correlations. Only positive events will be used in E-Cube and are presented in (a) and (b).

produces an event $e_k = (\boldsymbol{p}, t_k, b_k)$, detailing the time $t_k$, pixel location $\boldsymbol{p} = (x, y)^T$, and polarity $b_k$. The polarity, shown as blue for +1 and red for -1, indicates an increase or decrease in light intensity compared to the previous reading at that pixel. Working principles are detailed in Appendix-B.1.

Event cameras' ultra-high sampling rate and sensitivity capture rapid pixel-level changes in dynamic industrial settings, aiding high-speed object/self-motion tracking, SLAM, and fast obstacle avoidance on drones[21]. We find the resulting event stream also records object movement direction and distance across video frames, aiding in swift extraction of motion vectors (denoted by → in Fig. 5) as detailed in §5.2.1. These motion vectors can be leveraged to enhance video streaming efficiency in three key aspects.

• **Alleviating the overhead of inter-frame prediction (§5.2).** By connecting the motion vectors across frames, we can derive the movement trajectory of each individual object (e.g., humans) shown in consecutive frames. The pixel regions on each frame that align with this trajectory exhibit inherent correlations. This enables us to directly obtain inter-frame content correlations without the need for prediction. For instance, in Fig. 5a, we trace the worker's right foot's movement (marked as □) from lifting to landing between frames #668 and #674 by linking motion vectors. To encode a specific block □ in frame #674, we can directly locate its corresponding block in frame #668 by retracing the trajectory, eliminating the need for exhaustive block searches.

• **Improving the efficiency of intra-frame block division (§5.3).** We can merge those adjacent regions that share parallel trajectories into a large block as the structure and pixel content within these regions remain consistent over different frames. As shown in Fig. 5b, the small pixel regions denoted as □ are merged into a large block (denoted as a red box □). Employing a large block not only decreases the count of blocks on the current frame to be matched but also reduces the number of candidate blocks on the reference frame to be checked, improving the efficiency.

• **Enhancing the RoI segmentation performance (§5.4).** By comparing Fig. 5a and Fig. 5b, a clear distinction is seen in motion vectors between foreground RoIs (e.g., moving pedestrians) and the static background (e.g., oil tanks) in terms of vector direction and length. In the X-Y-centered view, the background's event stream consistently shifts towards the lower right, while the RoIs exhibit a zigzag pattern moving leftward. This variation in motion patterns serves as a useful guide for detecting and segmenting foreground RoIs.

Obtaining content correlations based on event streams can be expedited in chips (detailed in Appendix-E). It consumes 9-20× lower power consumption compared to the optical-flow-based[60] and learning-based[26] algorithms.

## 5 E-Cube: System Design

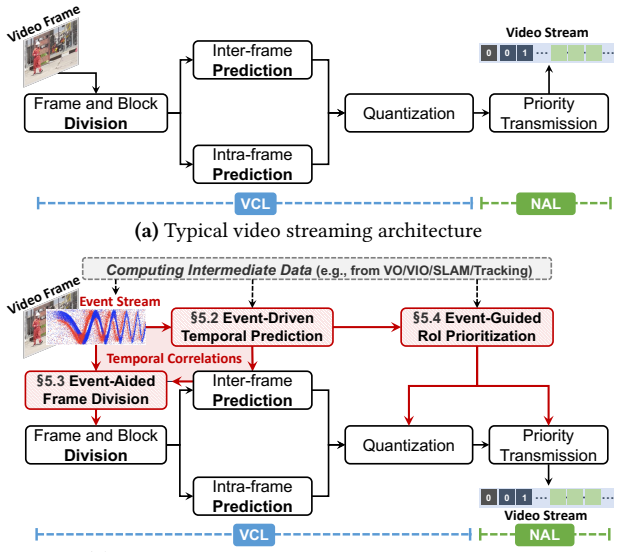
### 5.1 *Intermediate-Data-Reused* Architecture

We design and implement E-Cube to harvest the above opportunities. Fig. 6 compares the video streaming pipelines without (Fig. 6a) and with (Fig. 6b) E-Cube. As illustrated in Fig. 6b, from a high-level perspective, E-Cube can reuse intermediate results from computing subsystems such as motion vectors, feature matching results, and optical flow to assist video streaming. Regarding the specific dataflow, when encoding a raw video frame, E-Cube employs the event stream starting from the latest I-frame up to the present to derive the temporal content correlations between these two frames. The correlation is then leveraged for accurate and efficient *frame/block division* (§5.3), *inter-frame prediction* (§5.2), and *RoI prioritization* (§5.4), as elaborated below.

### 5.2 Event-Driven Temporal Prediction

**5.2.1 Obtain Motion Vector from Event Stream.** E-Cube partitions the 2D $x$-$y$ plane into multiple 4×4 pixel regions. Motion vectors are calculated by monitoring events that are collected within each pixel region on a window basis, with a duration of $\tau = 3.33ms$. The pixel region and the window length $\tau$ are determined based on two considerations:

• First, the 4x4 size matches the smallest block size in standard codecs, ensuring compatibility. Additionally, current commercial event cameras have a maximum resolution of




**(a)** Typical video streaming architecture

**(b)** E-Cube-integrated video streaming architecture

**Figure 6.** Overview of E-Cube.

720p (1280x720)[40], while 4K images are 3840×2160. Each event pixel links to a 3×3 region in 4K images, making the 4×4 size algorithmically suitable as the minimum unit.

• Second, for 60fps videos, $\tau = 3.33ms$ corresponds to one-fifth of the inter-frame intervals, aligning computation with video frames. Additionally, within this short period (e.g., 1/5 of the inter-frame interval for 60fps video), linear motion assumptions still hold and these assumptions simplify the motion vector calculation. In our real-world inspection scenarios and across our dataset (Table 3), even in highly dynamic situations, 89% of motion vectors follow linear patterns within $\tau$ windows.

Inspired by E-Flow[13], we estimate the motion vector by determining the tangent plane of a point cloud in the temporal domain, as shown in Fig. 7a. Briefly, for all positive events $E_i^{t_k} = \{e(\boldsymbol{p}, t)\}$ within the 4×4 pixel region $b_i$ centered at $\boldsymbol{p_i}$ and starting from time $t_k$, we define the function $\Sigma_e$ that maps each event's pixel location $\boldsymbol{p}$ to time $t$:

$$\begin{aligned} \Sigma_e : \mathbb{N}^2 &\rightarrow \mathbb{R} \\ \boldsymbol{p} &\mapsto \Sigma_e(\boldsymbol{p}) = t \end{aligned} \tag{1}$$

In the $x$-$y$-$t$ space, $\Sigma_e$ represents a surface created by the time of all positive events, as illustrated in Fig. 7a. The gradient of $\Sigma_e$ reflects the motion velocity of the visual content:

$$\nabla\Sigma_e = \left(\frac{1}{v_x}, \frac{1}{v_y}\right)^T. \tag{2}$$

Details on the derivation of Eq.2 can be found in Appendix-B.2. We employ the finite difference method[13] to approximate $\nabla\Sigma_e$ and compute the motion velocity $\boldsymbol{v} = (v_x, v_y)$ within each spatio-temporal unit using Eq.2. Notably, before computing event-based motion vectors, we apply denoising

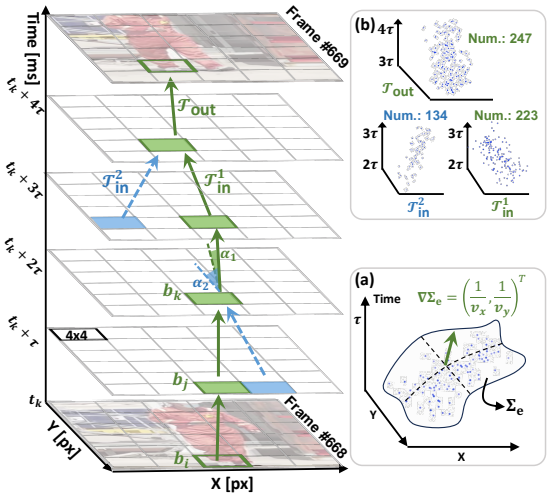



**Figure 7.** Illustration of establishing content temporal correlation from event stream.

and preprocessing to the event stream following the methodology described in [13]. The entire computation process can be accelerated using FPGA implementation, with detailed implementation specifics provided in Appendix-E.

Assuming the pixel location of $\boldsymbol{p_i} + \tau\boldsymbol{v}$ lies in the pixel region $b_j$, this suggests that the pixel region $b_i$ (a.k.a., start block) at time $t_k$ has transitioned to pixel region $b_j$ (a.k.a., end block) at time $t_k + \tau$, as illustrated by Fig. 7. We record such correlation with a motion vector $\mathcal{T}_{t_k}^{t_k+\tau}(b_i) = b_j$.

### 5.2.2 Temporal Content Correlation Profiling.

After gathering motion vectors over 3.33ms intervals, we link them across time to profile temporal correlations between frames. When the end of one vector aligns with another's start, they form a longer vector tracking the same content over an extended period. Ideally, each 4x4 pixel region should have a *one-to-one* mapping. with at most one motion vector starting and another one ending at it. However, due to scene dynamics, multiple motion vectors often converge to the same pixel region. Fig. 7 illustrates this issue where two motion vectors converge to the same pixel region at $t_k + 2\tau$ and $t_k + 4\tau$. Formally, let $\{\mathcal{T}_{\text{in}}^i\}$ be the set of incoming motion vectors that map to an outgoing vector $\mathcal{T}_{\text{out}}$, we select from $\{\mathcal{T}_{\text{in}}^i\}$ the most suitable motion vector that can accurately depict content motion based on the following two criteria:

• *Continuity.* The motion direction of the same video content tends to be consistent over very short intervals $\tau = 3.33ms$, rarely undergoing abrupt changes. E-Cube computes the angle $\alpha_i = \langle \mathcal{T}_{\text{in}}^i, \mathcal{T}_{\text{out}} \rangle$ between each incoming and outgoing motion vector. The continuity score, i.e., $\cos\alpha_i$, is calculated. $\mathcal{T}_{\text{in}}^i$ with a larger score will be more likely connected.

• *Event number consistency.* The number of events generated by the same object in adjacent time intervals should be comparable. E-Cube calculates the difference in event numbers, $\Delta_i = ||E_{\text{in}}^i| - |E_{\text{out}}||$, to assess their similarity. Lower $\Delta_i$ indicates a higher likelihood of the vectors describing the same

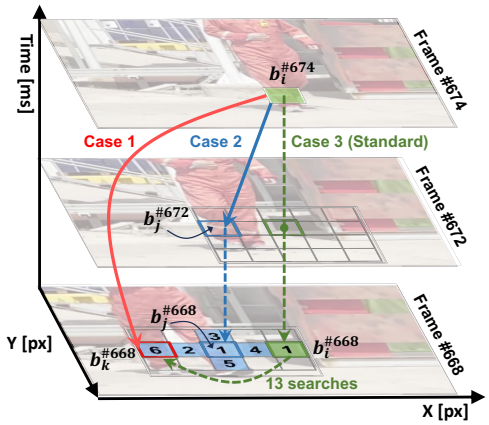



**Figure 8.** Illustration of inter-frame prediction with established content correlations.

object. As shown in Fig. 7b, E-Cube selects $\mathcal{T}_{\text{in}}^1$ over $\mathcal{T}_{\text{in}}^2$ due to its closer event count match with $\mathcal{T}_{\text{out}}$.

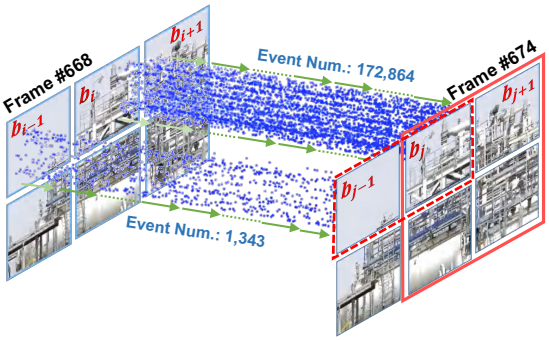
E-Cube selects the best incoming motion vector $\mathcal{T}_{\text{in}}^i$ by jointly considering these two parameters as normalized $\cos \alpha_i$ - normalized $\Delta_i$. By iteratively connecting motion vectors over time intervals, E-Cube traces the trajectories of content motions and establishes their temporal correlation.

**5.2.3 Prediction with Temporal Correlation.** The cross-frame content temporal correlations established by E-Cube facilitate block matching in inter-frame prediction[1]. For each block in the current frame, there are three cases during block matching. Fig. 8 illustrates these cases when E-Cube operates block matching for block $b_i^{\#674}$ in the current frame #674.

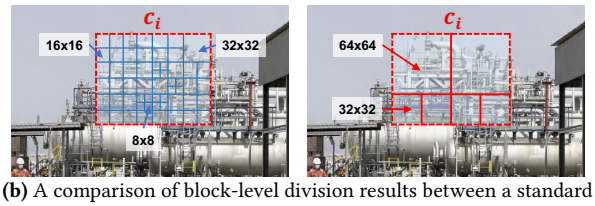
- Case 1 (denoted by →): the motion trajectory (i.e., a sequence of motion vectors connected over time) originates from a block $b_k^{\#668}$ in the I-frame (#668) and directly points to block $b_i^{\#674}$. In this case, E-Cube can immediately pick $b_k^{\#668}$ as the prediction result without running block searching.
- Case 2 (denoted by → + ⇢): E-Cube does not establish a temporal correlation between $b_i^{\#674}$ and a block in frame #668. However, it does identify a correlation with a block in a preceding frame within the current GoP, for instance, $b_j^{\#672}$ in frame #672. In such case, E-Cube initiates the block search at $b_j^{\#668}$ instead of starting at $b_i^{\#668}$ in frame #668. This approach reduces the searches to only 6, cutting the effort by 50% compared to the standard method that requires 12 searches starting from a more distant block $b_i^{\#668}$.
- Case 3 (denoted by ⇢): in rare cases where block $b_i^{\#674}$ fails to establish correlations with any blocks in previous frames, the codec resorts to the conventional block searching.

Our experiments in §7.4 show that nearly 50% of blocks align with Case 1 and 30% with Case 2 during inter-frame prediction. Only around 20% of blocks correspond to Case 3 that can not benefit from E-Cube. This indicates that E-Cube could substantially enhance inter-frame prediction.

[1]The block division optimization will be elaborated in §5.3.




**(a)** Region merging based on event stream




**(b)** A comparison of block-level division results between a standard codec (left) and E-Cube-integrated codec (right)

**Figure 9.** E-Cube in the frame/block division process.

### 5.3 Event-Aided Frame/Block Division

**5.3.1 Pixel Region Merging.** Neighboring pixel regions can be merged if they meet the following two criteria:

- *Relative location consistency.* As shown in Fig. 9a, consider $b_i$ and $b_{i+1}$ as two adjacent pixel regions in frame #668, and $b_j$ and $b_{j+1}$ as two neighboring pixel regions in frame #674. If pixel region $b_i$ correlates with $b_j$, then its immediate neighbor pixel region $b_{i+1}$ should exhibit a similar temporal correlation with $b_j$'s neighbor pixel region $b_{j+1}$. In other words, if $\mathcal{T}_{t_k}^{t_k+\tau}(b_i) = b_j$, then $\mathcal{T}_{t_k}^{t_k+\tau}(b_{i+1}) = b_{j+1}$.
- *Event number consistency.* In Fig. 9a, although adjacent regions $b_{i\to j}$ and $b_{i-1\to j-1}$ show parallel trajectories, they should not merge due to differing content. E-Cube further employs event number consistency, requiring less than a 5% difference in the number of generated events between mergeable regions. For instance, $b_j$ and $b_{j-1}$ won't merge due to their large event number discrepancy (over 100×), due to the vast texture difference between sky and pipe.

After merging pixel regions, we obtain a plethora of larger regions $C = \{c_i\}$, each having a temporal correlation with a merged pixel region on the reference frame.

**5.3.2 Frame-level Division.** E-Cube assesses frame type by comparing the current frame's content with the preceding I-frame. With temporal correlations, E-Cube calculates inter-frame content similarity $r_I$, indicating the proportion of the frame covered by correlated pixel regions in $C$:

$$r_I = \frac{\sum_{c_i} w_i * l_i}{w * l},$$

where $l_i$ and $w_i$ are the length and width of the pixel region $c_i$, and $w$ and $l$ are the frame dimensions. If $r_I$ is below a predetermined threshold $\theta_I$, the frame is designated as a

new I-frame; if it exceeds, E-Cube classifies it as P-frame or B-frame based on $c_i$ sizes.

**P-frame and B-frame classification**. E-Cube calculates the proportion, $r_p$, of those pixel regions in $C$ with both their length and width are greater than 32 pixels:

$$r_p = \frac{|\{c_i \in C : l_i \geq 32 \wedge w_i \geq 32\}|}{|C|}.$$

$r_p$ reflects how smaller pixel regions in $C$ are consolidated into larger blocks. Higher $r_p$ values imply more static backgrounds, suitable for B-frame encoding which leverages bidirectional prediction[52]. E-Cube uses a threshold $\theta_p$ to classify frames: those with $r_p \leq \theta_p$ as P-frames and higher $r_p$ as B-frames. Based on the analysis of 50,000 H.266 coded frames, we set the adjustable $\theta_i$ and $\theta_p$ at 60% and 30%.

**5.3.3 Block-level Division.** After merging pixel regions, the current frame is divided into two distinct areas according to with and without correlations to the reference frame. E-Cube applies different block division strategies to them.

**Areas with temporal correlations.** In these areas, it's easier to identify consistent content regions in reference frames. Therefore, E-Cube divides these areas into larger yet fewer blocks, which will reduce the number of candidate blocks to be checked during inter-frame prediction. However, the pixel region merging process (§5.3.1) does not guarantee that the size of each merged region $c_i$ aligns with the macro-block sizes (e.g., 64×64, 8×8) preset by the video codec.

E-Cube tackles this challenge by conceptualizing block division as a two-dimensional area-filling problem. Briefly, E-Cube uses a greedy algorithm that gives priority to filling $c_i$ with the largest available macro-blocks first. As compared in Fig. 9b, unlike the original codec dividing a texture-rich region $c_i$ into many small blocks (i.e., 16×16 and 8×8), E-Cube divides $c_i$ using two large 64×64 and four 32×32 blocks.

**Areas without temporal correlations** are typically from emerging scenes. Existing codecs divide them by texture complexity, i.e., using smaller blocks for complex textures (e.g., intersecting pipes) and larger ones for simpler areas (e.g., sky). However, existing codecs apply multi-round Gabor[42] or wavelet[20] transforms on the frame for texture detection, heightening computational demands. E-Cube simplifies this by using the event number as a direct measure of texture richness[55]. Areas producing a high event number are indicative of textural complexity and are divided into smaller blocks, enhancing the efficiency of block division.

### 5.4 Event-Guided RoI Segmentation

E-Cube segments the RoI by separating events originating from dynamic objects from those of the background. The insight behind this separation lies in the distinct directions of event streams from foreground and background. Specifically, background pixel changes on the camera plane are primarily due to the camera's (or drone's) movement, with a motion direction opposite to that of the camera. Conversely, RoI pixel changes are caused by the overlay of the camera's movement and the RoI's own motion, creating a unique event stream pattern different from that caused by the background.

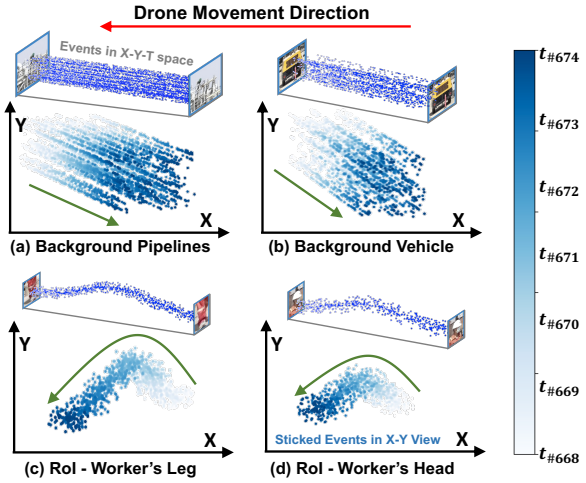



**Figure 10.** Illustrations of the directional differences between RoI- and background-triggered event streams.

Fig. 10 shows event streams from four areas between frames at time $t_{\#668}$ and $t_{\#674}$, including both the original event stream in the $x$-$y$-$t$ space and its projection on the current $x$-$y$ plane, where darker points represent events generated later. As seen, the streams' direction differs noticeably between RoIs and backgrounds: background events typically move opposite to the camera direction, while RoI events lack this correlation. Such difference aids in RoI segmentation.

E-Cube implements an IMU-based ego-motion compensation algorithm for filtering RoI-triggered events based on existing works[19, 55], refined by integrating motion vectors from streaming to enhance accuracy despite IMU drift and noise issues. Algorithm details are provided in Appendix-C. E-Cube matches the accuracy of advanced DNN-based segmentation[25, 28] but is 10× faster in inference (§7.4). After obtaining RoI regions, E-Cube ensures they are encoded in high precision and transmitted in high priority for back-end tasks. Details can be found in Appendix-D.

## 6 Implementation of E-Cube on Drones

We implement E-Cube on a Xilinx Zynq-7020 chip[1], which consists of a processing system (PS) and a programmable logic (PL) module, as shown in Fig. 22. The PS features a dual-core ARM Cortex-A9 processor (#A1 and #A2), while the PL expedites algorithm execution through FPGA. The software-hardware co-design details, including data input/output facilitation, voltage adaptation, and synchronization between RGB and event cameras, can be found in Appendix-E. The integration of E-Cube with the drone's flight control subsystem and reuse of intermediate data proceeds as follows:

**Hardware.** We implement E-Cube on DJI Matrice 300 and AMOVLAB P-450 (Fig. 11a) drones for industrial inspection.

**Table 3.** Dataset Description

| Dataset | Label | Total Frames | Drone Flight Speed (m/s) | Typical Scenario |
|---|---|---|---|---|
| Oil Field | Normal | 2,069,042 | <5m/s | **Pipeline Inspection** (with static pipelines and oil tanks) |
| | Dynamic | 1,674,543 | 5~10m/s | **Living Area Safety Inspection** (with frequent entry and exit of workers and vehicles) |
| | Highly Dynamic | 842,654 | >10m/s | **Large-scale Oil Production Area Inspection** (with frequent movement of workers and dynamic lighting) |
| VisDrone[64] | Normal | 145,613 | N/A | **Farms and Factories** (with sparse crowds) |
| | Dynamic | 73,295 | N/A | **City Roads** (with moving pedestrians and vehicles) |
| | Highly Dynamic | 42,996 | N/A | **Busy Urban Traffic** |

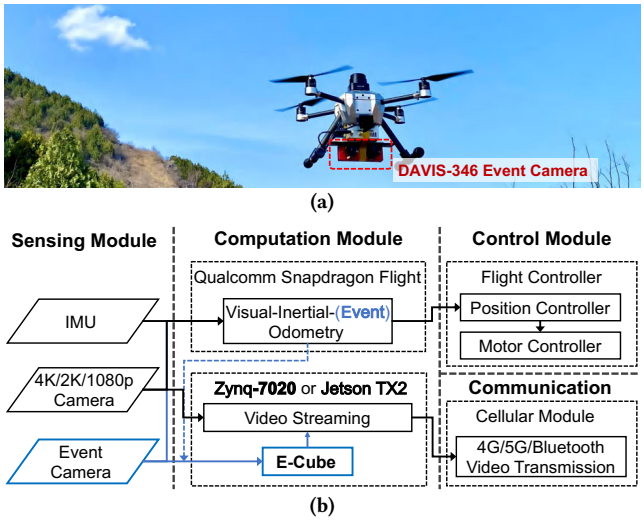



**Figure 11.** E-Cube's implementation on an industrial drone

The drones are simultaneously equipped with event cameras (one DAVIS-346[2] and one Prophesee IMX636 HD[40]), and frame-based cameras with resolutions of 1080p, 2K, or 4K. As depicted in Fig. 11b, each drone has two on-board computational units: (*i*) a Qualcomm Snapdragon Flight for drone pose estimation using event-based (switchable) visual-inertial odometry (VIO) algorithms[43, 63]; and (*ii*) either a Xilinx Zynq-7020 chip or an Nvidia Jetson TX2 with an AUVIDEA J90 carrier board[4]. E-Cube leverages intermediate results such as motion vectors computed by the flight controller's VO/VIO modules (blue dotted data flow in Fig. 11b), and introduces them into the Jetson TX2 or Zynq-7020 platforms to enhance video streaming performance. The resulting video streams are then sent to the communication module for transmission via wireless technologies.

**Software.** We develop the software stack in C++ and use the Robot Operating System (ROS Melodic[44]) for inter-module communication. The open-source event camera driver[37] sends event data from the camera to either Jetson TX2 or Zynq for processing. Event data denoising, pre-processing (e.g., spatiotemporal reorganization), and event-based motion vector extraction utilize algorithms proposed in E-Flow[13].

# 7 Evaluation

In this section, we first present the experimental methodology (§7.1), followed by the overall performance of E-Cube compared with existing solutions (§7.2). We then investigate the efficiency and end-to-end latency of E-Cube in §7.3. Further, we conduct an ablation study to understand each functional module in E-Cube (§7.4).

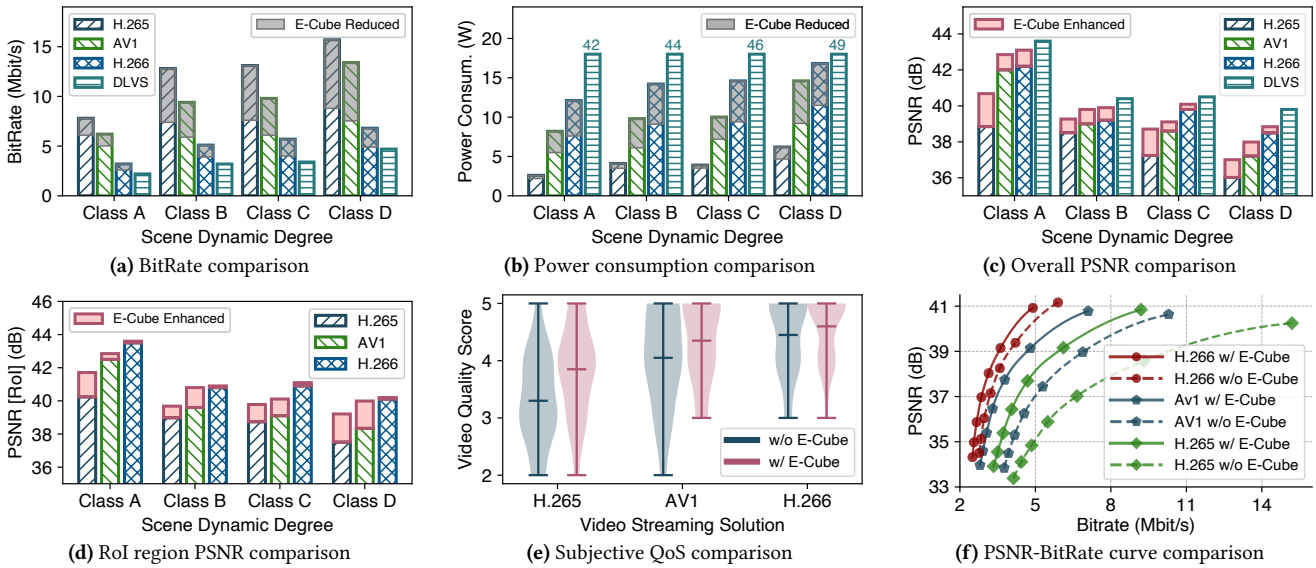
## 7.1 Experimental Methodology

**Field studies in the oil field.** We integrate E-Cube into the drone's flight system (§6) and deploy 8 drones in the oil-field to perform industrial inspections, where drones encode and stream videos, using E-Cube, to a server in real-time. To compare various video streaming solutions, we leverage the raw videos and event data collected by the drones to build a dataset for trace-driven evaluation.

**Trace-driven evaluations.** Following the standard video streaming evaluation methodology[17, 57, 58], we conduct comprehensive trace-driven evaluations based on: (*i*) the aforementioned handcrafted dataset with timestamp-aligned events and videos; (*ii*) VisDrone[64], a widely recognized large-scale video collection recorded by drones, supplemented with simulated event data generated through a comprehensive video-to-event neural network[22]. Details characterizing these datasets are summarized in Table 3.

**Scene dynamics**. We classified video clips into four classes from A to D based on scene dynamics, as detailed in Table 5. Briefly, **A** includes normal scenes, **B** and **C** encompass dynamic scenes, and **D** represents highly dynamic scenes. The evaluation of scene dynamics combines objective metrics including video content spatial and temporal correlation defined by ITU-T P.910[47], and subjective user perception. Appendix-F.1 shows representative scenes from each class.

**Metrics.** We evaluate video streaming performance using two key metrics: peak signal-to-noise ratio (PSNR, measured in dB) and BitRate (measured in Mbps). Higher PSNR indicates better image quality; lower BitRate suggests efficient compression. Additionally, we conducted Quality of Experience (QoE) assessments with 20 oilfield workers, rating video clarity and stability on a 1 (worst) to 5 (best) scale.




**(a)** BitRate comparison **(b)** Power consumption comparison **(c)** Overall PSNR comparison

**(d)** RoI region PSNR comparison **(e)** Subjective QoS comparison **(f)** PSNR-BitRate curve comparison

**Figure 12.** Overview Performance of E-Cube

**Baselines.** We integrate E-Cube into three drone video streaming systems: a hardware-based H.265 on Zynq-7020 (FPGA), and software-based H.266 and AV1 on Jetson TX2 (C++). Additionally, a state-of-the-art deep-learning video streaming (DLVS) system, Swift[17], is implemented on Jetson TX2 (GPU) for comparison. Configuration details for each codec are provided in Appendix-F.2.

**Power consumption measurement.** For software-based H.266, AV1, and Swift, we use the Linux command `tegrastats` (combined with external USB power meter validation for cross-verification) to monitor run-time power usage brought by each module of E-Cube. For Zynq-7020, we employed hardware current probes on VCC rails with 1kHz sampling rate, measuring all sensor input, baseboard, PS, and PL power consumption separately. The power consumption (in $W$) is determined by dividing this energy overhead by the video's duration. All measurements include complete system overhead: event camera consumption (<0.3W), E-Cube processing costs, and baseline codec power.

## 7.2 Overall Performance Comparison

**7.2.1 BitRate.** We first investigate the bitrate reduction achieved by E-Cube, indicating improved compression rates. As depicted in Fig. 12a , in class A, E-Cube reduces the bandwidth bitrate by around 20% across various video streaming solutions while maintaining the same video PSNR. In classes B and C, it provides an average bitrate saving of 42%, 37%, and 26% for H.265, AV1, and H.266 respectively; And in highly dynamic scenes (class D), the savings increase to 44%, 43%, and 27%, with E-Cube enabling H.266 to achieve lower or comparable bandwidth costs to the DLVS. These bitrate savings stem from two factors: (*i*) E-Cube's strategy for selecting I-frames, based on the current frame's predictability, aligns with encoding needs, decreasing low-compression I-frames; and (*ii*) E-Cube aids each block in finding optimal matches during inter-frame prediction, reducing residual data for encoding.

**7.2.2 Power Consumption.** As shown in Fig. 12b, integrating E-Cube results in a significant reduction in power consumption by efficiently identifying optimal matches during inter-frame prediction. Importantly, the power consumption of both the event camera and E-Cube's algorithm is accounted for in this experiment.

For H.265, which employs a heuristic block searching strategy during inter-frame prediction, sacrificing accuracy for fewer searches (typically not exceeding six searches per block), E-Cube's improvement is relatively modest, at about 10%. As for AV1, E-Cube's integration leads to power savings exceeding 28% across all scenarios (classes A-D). The recent H.266 standard allows for multi-reference frame prediction, expanding potential reference frames beyond the immediate preceding I-frame[14]. However, this method further increases the number of block-matching operations required. Integrated with H.266, E-Cube's capacity to directly locate the potential best match on each frame becomes even more advantageous, leading to over 35% power saving in all scenarios. Comparatively, the high power demand of deep neural network inference, nearing 50W, is 5-8× that of existing approaches, posing challenges for real-time, low-cost operation on drones.

**7.2.3 Video Quality.** We evaluated video quality improvement with E-Cube across different codecs at a fixed 15Mbps bit-rate. Results depicted in Fig. 12c show for H.265 and AV1, E-Cube increases PSNR by 0.85dB, 0.76dB, 0.62dB, and 0.82dB in scene classes A-D, respectively. As for H.266, the gain hovers around 0.3dB, placing its performance comparable

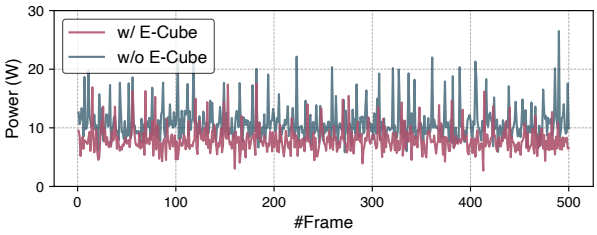



**Figure 13.** Power Measurements

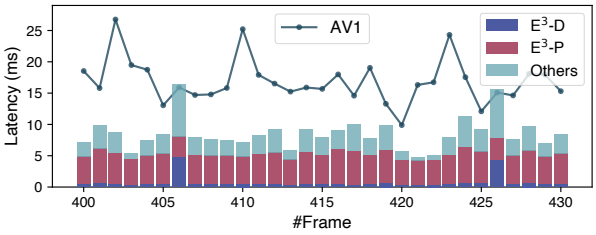



**Figure 14.** End-to-End Latency Measurements

with the latest DLVS solution, Swift, which record over 40dB PSNR. This quality enhancement is due to E-Cube's accurate block predictions during inter-frame prediction, reducing video quality degradation from residual accumulation.

We further examine the video quality in RoI regions (e.g., moving vehicles and pedestrians) to demonstrate the effectiveness of E-Cube in RoI prioritization. As depicted in Fig. 12d, E-Cube shows significant improvements in PSNR for RoI regions across scene classes A-D, with increases of over 0.4, 1.1, 0.7, and 1.6dB for H.265 and AV1. This enhancement is more pronounced in highly dynamic scenes (classes B and D) due to E-Cube's effective RoI segmentation. Compared to H.266, E-Cube offers a modest gain of around 0.3dB but is more lightweight, as detailed in §7.4.

**7.2.4 User Experience.** We evaluate the objective QoE each solution provides. Fig. 12e depicts the distribution of ratings from 20 workers on the output video quality by different codecs with and without E-Cube. For original H.265, AV1, and H.266, the average scores are 3.2, 3.9, and 4.5. With E-Cube, these scores increase to 3.9, 4.2, and 4.6, respectively.

**7.2.5 Overall PSNR-BitRate Curve.** To provide a comprehensive performance comparison of different solutions, we utilize the widely recognized BitRate-PSNR curve[17], where each point on this curve represents the minimum bitrate required to stream the original video at a specified video quality. Fig. 12f depicts that integrating E-Cube shifts these curves towards the PSNR-axis for all codecs. This indicates more effective compression, achieving comparable video quality with reduced bandwidth requirements.

In summary, the above results demonstrate that by leveraging the temporal correlations acquired from the event stream, E-Cube's integration reduces computation, power, and bandwidth overhead while enhancing video quality.

### 7.3 Efficiency Evaluation

We stream a 10s video of 500 frames from Class C using AV1 codec, assessing both scenarios: with and without E-Cube integration. We focus on the system's power consumption during this process. The results, depicted in Fig. 13, show a notable decrease in average power usage from 13.4W (standard AV1) to 9.8W with E-Cube, translating to a 26.9% reduction. A key observation is the significant reduction in power consumption 'spikes' when E-Cube is active. These spikes, typically linked to the energy-intensive I-frames that demand complete block division and intra-frame prediction, are less frequent compared to those P-frames and B-frames, which primarily depend on inter-frame prediction.

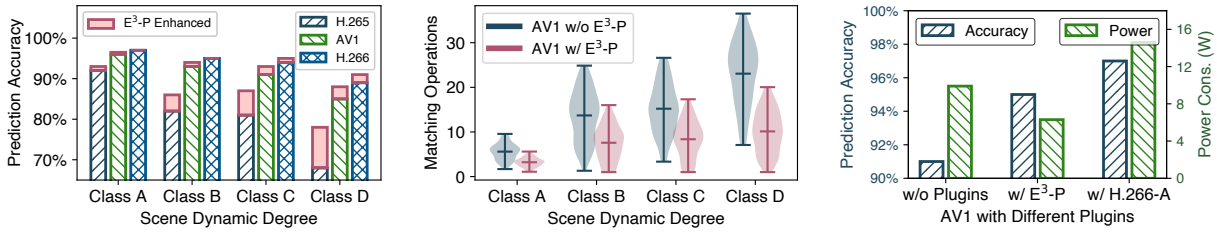
A further examination focuses on frames #400 to #430 of the streaming process, where we scrutinize the latency contributions of E-Cube's modules, $E^3$-D and $E^3$-P, alongside other codec components. The latency analysis is presented in Fig. 14. As seen, during the processing of P-frames and B-frames, the $E^3$-D module's activity is minimal, contributing less than 5% to the total frame latency. In contrast, the $E^3$-P module is more dominant, responsible for over 60% of the latency. However, for selected I-frames (frames #406 and #426), the $E^3$-D module is more actively involved, leading to increased latency contributions, along with a rise in power consumption of other codec components. Without E-Cube, three I-frames appear within the 30-frame segment, evident from the spikes at frames #403, #410, and #423. In contrast, with E-Cube integrated, only two I-frames occur. Overall, E-Cube reduces the average latency per frame by around 40%, showcasing its efficiency in video streaming.

### 7.4 System Micro-Benchmarks

We abbreviate the three functional modules of E-Cube, event-assisted temporal prediction, frame/block division, and RoI segmentation, as $E^3$-P, $E^3$-D, and $E^3$-R, respectively.

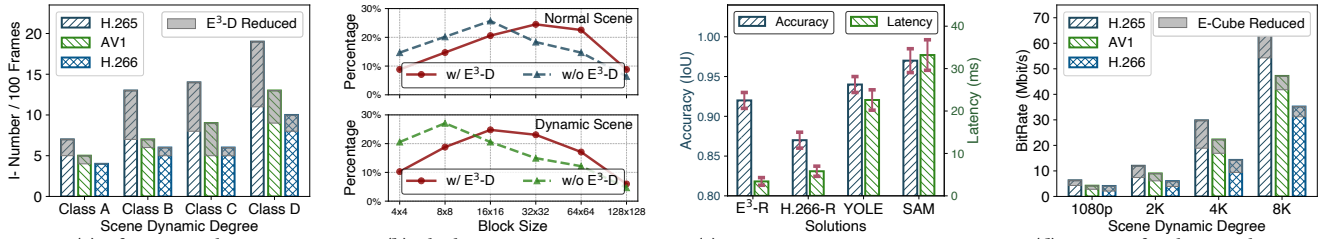
**7.4.1 Performance of Temporal Correlation Estimation.** We evaluate E-Cube's core functionality: obtaining temporal content correlations via event data. It is compared with frame-based LK-flow[60], DNN-based FlowNet 2.0[26] and SpyNet[45]. We selected video clips from our dataset under normal lighting conditions (45-55dB) and low-light conditions (<25dB) to validate the robustness of our results. Results in Table 4 show E-Cube surpasses LK-flow in accuracy (measured by average end-point pixel error, AEE[60]) and matches FlowNet and SpyNet. E-Cube's energy consumption is 8-20× lower than these methods, particularly on Zynq platforms where FlowNet and SpyNet cannot currently be deployed, favoring its integration into streaming systems.

Moreover, under low-light conditions, E-Cube demonstrates improved accuracy compared to LK-Flow and approaches the performance of DNN-based solutions (4.9 *vs* 4.5/4.8 pixel errors), exhibiting less performance degradation than under normal lighting. The advantage stems from event




**(a)** Increased Prediction Accuracy **(b)** Decreased Block-matching Operations **(c)** Accuracy and Efficiency Comparison

**Figure 15.** Performance Lift Brought by Event-Driven Temporal Prediction (E$^3$-P)




**(a)** I-frame Number **(b)** Block-size Percentage **(c)** RoI Segmentation Accuracy **(d)** Impact of Video Resolution

**Figure 16.** Performance Lift Brought by Event-aided Division (E$^3$-D) and RoI Prioritization (E$^3$-R)

**Table 4.** Temporal Correlation Comparison

| Method | Accuracy (in AEE, pixel error) | | Energy Usage (in Power, W) | |
|---|---|---|---|---|
| | Normal Light | Low Light | Zynq-7020 | Jetson TX2 |
| E-Cube | 3.9 | 4.9 | **0.5** | **1.2** |
| LK-Flow[60] | 6.1 | 8.7 | 4.6 | 9.6 |
| FlowNet 2.0[26] | 2.6 | 4.8 | N/A | 24.6 |
| SpyNet[45] | **2.3** | **4.5** | N/A | 28.9 |

cameras' high dynamic range, typically reaching 120dB, demonstrating E-Cube's effectiveness in enhancing video streaming performance even in challenging lighting conditions.

**7.4.2 Performance of E$^3$-P.** Fig. 15a shows accuracy improvement in inter-frame prediction. For H.265 and AV1, E-Cube enhances inter-frame prediction accuracy by over 1.5% across all scenarios, especially in highly dynamic scenes (i.e., Class D) where increases range from 4% to 9.3%. Even for H.266, E-Cube achieves 1-3% accuracy improvement in dynamic scenes. E-Cube not only improves accuracy but also reduces block-matching operations needed. Fig. 15b illustrates that AV1's searches during inter-frame prediction are halved on average after integrating E-Cube. Additionally, extreme cases requiring more than 20 matches are significantly reduced, greatly enhancing coding efficiency.

We evaluate inter-frame prediction accuracy and energy consumption of AV1 integrated with E-Cube and H.266 with inter-frame prediction add-on plugin (H.266-A) in dynamic scenes (Class B). As shown in Fig. 15c, original AV1 has 91% prediction accuracy and consumes 8.7W. Integrating E-Cube improves accuracy to 95.8% while reducing power consumption to 5.7W. Conversely, while H.266-A increases accuracy to 97.2%, it significantly raises energy consumption to 15.2W, which could adversely impact drone flight endurance. E-Cube effectively enhances prediction accuracy while reducing power consumption.

**7.4.3 Performance of E$^3$-D.** At the frame level, as depicted in Fig. 16a, E$^3$-D reduces I-frame frequency in the video stream. For H.265, E$^3$-D achieves average I-frame reductions of 28%, 46%, 43%, and 42% across Class A-D scenes, respectively. For AV1 and H.266, E$^3$-D cuts I-frame numbers by over 15% in dynamic scenes, particularly Class C. This stems from E-Cube's effective correlation of dynamic objects across frames, preventing unnecessary I-frame increases and lowering streaming bitrate. At the block level, E$^3$-D shifts from using predominantly smaller blocks (e.g., 4×4) to favoring larger blocks (i.e., over 16×16). This change will decrease the number of blocks required for inter-frame prediction, thereby boosting coding efficiency. Fig. 16b shows this shift, with the ratio of block sizes like 32×32 increasing by 7% and 12% in normal and dynamic scenarios, respectively. Meanwhile, the 64×64-sized blocks also increase 9% and 5%.

**7.4.4 Performance of E$^3$-R.** In our evaluation, E$^3$-R is compared with H.266-R[14], YOLE[15], and SAM[28] in varied scenarios. H.266-R is an add-on for H.266 focusing on RoI segmentation, while YOLE and SAM are deep neural network frameworks for RoI segmentation using event data and standard images. Fig. 16c shows E$^3$-R's average IoU for RoI segmentation at 0.92, better than H.266-R's 0.87 but below YOLE's 0.94 and SAM's 0.97. Yet, E$^3$-R's latency of 3.4ms is significantly less than others.

**7.4.5 Impact of Video Resolutions.** We measure the bitrates of output video streams from different codecs, both with and without E-Cube integration, in dynamic environments. As depicted Fig. 16d, E-Cube consistently reduces bitrates across all solutions: by over 35% for 1080p, 32% for 2k, 24% for 4k, and 9% for 8k resolutions. Notably, for H.265, the bitrate savings reached up to 40% for 1080p and 2k inputs.

These results confirm E-Cube's effectiveness across various video resolutions, enhancing video streaming efficiency.

## 8 Related work

**Video coding.** Widely used video codecs for drone-based lightweight video streaming include H.264[52], H.265[50], VP9[38], AV1[16], H.266[14], etc.. H.265 dominates mobile video streaming, AV1 excels in service versatility, and H.266, used in advanced drones like DJI-Phantom and DJI-Inspire, represents cutting-edge mobile streaming technology. However, in complex industrial contexts, these codecs' coding accuracy and efficiency suffer due to a lack of content temporal correlations between frames, as elaborated in §3.

Nowadays, various DNN-based video coding solutions hit the mainstream[11, 57, 59, 62], with numerous applications such as large-scale surveillance [29], mobile-cloud gaming [7], short video streaming [30], and volumetric video streaming [23, 24, 32]. However, they demand significant computational resources from GPUs, and typically focus on *server-to-mobile* video distribution. Such overhead renders them impractical for the reverse *mobile-to-server* video streaming on resource-limited drones.

E-Cube leverages intermediate results from computing modules to enhance video streaming design principles, reducing computation, power, and bandwidth requirements while improving video quality. It is compatible with advanced codecs for further enhancements.

**Mobile video streaming.** Recently, various video streaming approaches like bitrate selection[31, 49, 56], video preprocessing[53, 61], and server-side super-resolution[58] have been used for performance enhancement of video streaming applications. Different from these approaches that focus on how to deliver high-quality (QoE) video streams with network bandwidth constraints, E-Cube emphasizes generating better quality and RoI-prioritized video streams with resource and bandwidth constraints. E-Cube is orthogonal to, and could be jointly used with previous works.

**Event-based systems.** Event cameras offer numerous potential advantages over conventional frame-based cameras, including high temporal resolution, low latency, high dynamic range, and low power consumption[21]. Recent systems use them for high-speed SLAM[46], object tracking[35], HDR image reconstruction[36], and obstacle avoidance on drones[19, 55]. DJI, Amazon, Sony, and Samsung are increasingly incorporating event sensors in their mobile platforms like handheld cameras and drones[6, 18]. E-Cube is the pioneering work that harnesses the rich temporal information of event data to enhance video streaming systems.

## 9 Limitation and Future Direction

**The Camera Resolution Issue.** As depicted in §7.4.5, we observe diminishing gains brought by E-Cube with increasing video resolutions. This limitation stems from the current event camera's maximum hardware resolution of 720p. In 4K and 8K, each event camera pixel corresponds to larger areas (e.g., 9-16 pixels) in the video, reducing temporal correlation accuracy. This affects E-Cube's prediction and division efficiency in high-resolution frames. As event camera technology advances, with incorporation by companies like PROPHESEE and Sony[6, 40], we anticipate more significant benefits from E-Cube in video streaming.

**Non-linear Motion and Static Scenarios.** Our $\tau$=3.33ms temporal window ensures linear motion assumptions hold for most realistic scenarios, even during aggressive drone maneuvers. However, when scene changes become extremely abrupt, such as irregular drone movements or erratic object motion, the linear assumption may fail, reducing temporal correlation accuracy. Potential solutions include shorter time windows or advanced event-based algorithms for flow estimation, though both increase computational burden. Similarly, event cameras cannot capture information in completely static scenarios, diminishing E-Cube's effectiveness. Nevertheless, static scenes pose minimal challenges for video streaming systems, as existing codecs already handle such content efficiently without requiring additional assistance.

**RGB *vs*. Event Camera.** This work aims to demonstrate our core insight: *whether intermediate results from computing subsystems in mobile systems can enhance video streaming subsystems.* E-Cube currently utilizes rich temporal data from event cameras within computing subsystems to validate the feasibility of this design approach. Future work will explore more universal RGB-based camera solutions. However, obtaining pixel-level, instantaneous content correlations from RGB cameras cannot be achieved through hardware characteristics alone as with event cameras, requiring algorithmic design and system-level acceleration. This represents a significant avenue for our subsequent research efforts.

## 10 Conclusion

We have presented the design and implementation of E-Cube, a framework that leverages a novel sensing modality, event camera, to enhance video streaming on lightweight mobile devices. E-Cube fully leverages rich temporal data captured by the event camera to directly establish *temporal content correlations* among adjacent video frames. E-Cube further exploits the correlations to enhance frame/block division, inter-frame prediction, and RoI segmentation. E-Cube can serve as a plug-in module and be integrated into prevalent codecs for better streaming performance with lower resource overhead. Extensive evaluations in a large-scale oil field and on a public dataset demonstrate its superior performance.

## Acknowledgments

We thank the anonymous reviewers and our shepherd, Olaf Spinczyk, for their constructive feedback that improved the presentation of this paper. This work is supported in part by the National Natural Science Foundation of China under grant No. 62302254.

## References


[1] 2018. *Zynq-7000 SoC Data Sheet: Overview.* Data Sheet DS190. Xilinx, Inc. https://docs.amd.com/v/u/en-US/ds190-Zynq-7000-Overview

[2] 2019. *DAVIS 346 — Dynamic and Active-pixel Vision Sensor (Data Sheet).* Technical Report. iniVation AG. https://inivation.com/wp-content/uploads/2019/08/DAVIS346.pdf

[3] 2020. *PS and PL-Based 1G/10G Ethernet Solution.* Technical Report XAPP1305. AMD (formerly Xilinx). https://docs.amd.com/v/u/en-US/xapp1305-ps-pl-based-ethernet-solution

[4] 2021. *Jetson TX2 Series System-on-Module Data Sheet.* Data Sheet v1.8. NVIDIA Corporation. https://openzeka.com/en/wp-content/uploads/2022/08/Jetson-TX2-Series-Module-Datasheet-v1.8.pdf

[5] DJI 2025. *DJI Inspire 3 — Specifications.* DJI. https://www.dji.com/inspire-3/specs

[6] Sony Semiconductor Solutions Corporation 2025. *Event-based Vision Sensor (EVS) — Products & Solutions.* Sony Semiconductor Solutions Corporation. https://www.sony-semicon.com/en/products/is/industry/evs.html

[7] Ahmad Alhilal, Tristan Braud, Bo Han, and Pan Hui. 2022. Nebula: Reliable low-latency video transmission for mobile cloud gaming. In *Proceedings of the ACM Web Conference.*

[8] Ali J Ben Ali, Zakieh Sadat Hashemifar, and Karthik Dantu. 2020. Edge-SLAM: edge-assisted visual simultaneous localization and mapping. In *Proceedings of the ACM Mobisys.*

[9] AMD (formerly Xilinx) 2025. *AXI DMA LogiCORE IP Product Guide* (v7.1 ed.). AMD (formerly Xilinx). https://docs.amd.com/r/en-US/pg021_axi_dma

[10] AMD (formerly Xilinx) 2025. *Libmetal and OpenAMP User Guide.* AMD (formerly Xilinx). https://docs.amd.com/r/en-US/ug1186-zynq-openamp-gsg

[11] Ghufran Baig, Jian He, Mubashir Adnan Qureshi, Lili Qiu, Guohai Chen, Peng Chen, and Yinliang Hu. 2019. Jigsaw: Robust live 4k video streaming. In *Proceedings of the ACM MobiCom.*

[12] Arjun Balasingam, Karthik Gopalakrishnan, Radhika Mittal, Mohammad Alizadeh, Hamsa Balakrishnan, and Hari Balakrishnan. 2021. Toward a Marketplace for Aerial Computing. In *Proceedings of the ACM DroNet.*

[13] Ryad Benosman, Charles Clercq, Xavier Lagorce, Sio-Hoi Ieng, and Chiara Bartolozzi. 2013. Event-based visual flow. *IEEE transactions on neural networks and learning systems* 25, 2 (2013), 407–417.

[14] Benjamin Bross, Ye-Kui Wang, Yan Ye, Shan Liu, Jianle Chen, Gary J Sullivan, and Jens-Rainer Ohm. 2021. Overview of the versatile video coding (VVC) standard and its applications. *IEEE Transactions on Circuits and Systems for Video Technology* 31, 10 (2021), 3736–3764.

[15] Marco Cannici, Marco Ciccone, Andrea Romanoni, and Matteo Matteucci. 2019. Asynchronous convolutional networks for object detection in neuromorphic cameras. In *Proceedings of the IEEE/CVF CVPR Workshops.*

[16] Yue Chen, Debargha Murherjee, Jingning Han, Adrian Grange, Yaowu Xu, Zoe Liu, Sarah Parker, Cheng Chen, Hui Su, Urvang Joshi, et al. 2018. An overview of core coding tools in the AV1 video codec. In *Proceedings of the IEEE PCS.*

[17] Mallesham Dasari, Kumara Kahatapitiya, Samir R Das, Aruna Balasubramanian, and Dimitris Samaras. 2022. Swift: Adaptive video streaming with layered neural codecs. In *Proceedings of the USENIX NSDI.*

[18] DJI. 2020. DJI Industrial Drones. https://www.dji.com/products/industrial. Accessed: 2025-09-12.

[19] Davide Falanga, Kevin Kleber, and Davide Scaramuzza. 2020. Dynamic obstacle avoidance for quadrotors with event cameras. *Science Robotics* 5, 40 (2020), eaaz9712.

[20] Marie Farge. 1992. Wavelet transforms and their applications to turbulence. *Annual review of fluid mechanics* 24, 1 (1992), 395–458.

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

[22] Daniel Gehrig, Mathias Gehrig, Javier Hidalgo-Carrió, and Davide Scaramuzza. 2020. Video to Events: Recycling Video Datasets for Event Cameras. In *Proceedings of the IEEE/CVF CVPR.*

[23] Ehab Ghabashneh, Chandan Bothra, Ramesh Govindan, Antonio Ortega, and Sanjay Rao. 2023. Dragonfly: Higher Perceptual Quality For Continuous 360 Video Playback. In *Proceedings of the ACM SIGCOMM.*

[24] Bo Han, Yu Liu, and Feng Qian. 2020. ViVo: Visibility-aware mobile volumetric video streaming. In *Proceedings of the ACM MobiCom.*

[25] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. 2017. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision.* 2961–2969.

[26] Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. 2017. Flownet 2.0: Evolution of optical flow estimation with deep networks. In *Proceedings of the IEEE/CVF CVPR.*

[27] Sagar Jha, Youjie Li, Shadi Noghabi, Vaishnavi Ranganathan, Peeyush Kumar, Andrew Nelson, Michael Toelle, Sudipta Sinha, Ranveer Chandra, and Anirudh Badam. 2021. Visage: enabling timely analytics for drone imagery. In *Proceedings of the ACM MobiCom.*

[28] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. 2023. Segment anything. *arXiv preprint arXiv:2304.02643* (2023).

[29] Jinyang Li, Zhenyu Li, Ri Lu, Kai Xiao, Songlin Li, Jufeng Chen, Jingyu Yang, Chunli Zong, Aiyun Chen, Qinghua Wu, et al. 2022. LiveNet: a low-latency video transport network for large-scale live streaming. In *Proceedings of the ACM SIGCOMM.*

[30] Zhuqi Li, Yaxiong Xie, Ravi Netravali, and Kyle Jamieson. 2023. Dashlet: Taming Swipe Uncertainty for Robust Short Video Streaming. In *Proceedings of the USENIX NSDI.*

[31] Xianshang Lin, Yunfei Ma, Junshao Zhang, Yao Cui, Jing Li, Shi Bai, Ziyue Zhang, Dennis Cai, Hongqiang Harry Liu, and Ming Zhang. 2022. GSO-simulcast: global stream orchestration in simulcast video conferencing systems. In *Proceedings of the ACM SIGCOMM.*

[32] Yu Liu, Bo Han, Feng Qian, Arvind Narayanan, and Zhi-Li Zhang. 2022. Vues: practical mobile volumetric video streaming through multiview transcoding. In *Proceedings of the ACM MobiCom.*

[33] Yunfei Ma, Nicholas Selby, and Fadel Adib. 2017. Drone relays for battery-free networks. In *Proceedings of the ACM Sigcomm.*

[34] Michael W Marcellin, Michael J Gormish, Ali Bilgin, and Martin P Boliek. 2000. An overview of JPEG-2000. In *Proceedings the IEEE DCC.*

[35] Anton Mitrokhin, Cornelia Fermüller, Chethan Parameshwara, and Yiannis Aloimonos. 2018. Event-based moving object detection and tracking. In *Proceedings of the IEEE IROS.*

[36] Mohammad Mostafavi, Lin Wang, and Kuk-Jin Yoon. 2021. Learning to reconstruct hdr images from events, with applications to depth and flow prediction. *Proceedings of the IEEE IJCV* (2021).

[37] Elias Mueggler, Basil Huber, and Davide Scaramuzza. 2014. Event-based, 6-DOF pose tracking for high-speed maneuvers. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems.* IEEE, 2761–2768.

[38] Debargha Mukherjee, Jim Bankoski, Adrian Grange, Jingning Han, John Koleszar, Paul Wilkins, Yaowu Xu, and Ronald Bultje. 2013. The latest open-source video codec VP9-an overview and preliminary results. In *Proceedings of the IEEE PCS.*

[39] Raul Mur-Artal and Juan D Tardós. 2017. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE Transactions on Robotics* 33, 5 (2017), 1255–1262.

[40] PROPHESEE. [n.d.]. PROPHESEE Event Camera Spec. https://www.prophesee.ai/event-camera-evk4/. Accessed: 2025-09-12.

[41] N Purnachand, Luis Nero Alves, and Antonio Navarro. 2012. Improvements to TZ search motion estimation algorithm for multiview video coding. In *Proceedings of the IEEE IWSSIP*. IEEE.
[42] Shie Qian and Dapang Chen. 1993. Discrete gabor transform. *IEEE transactions on signal processing* 41, 7 (1993), 2429–2438.
[43] Tong Qin, Peiliang Li, and Shaojie Shen. 2018. Vins-mono: A robust and versatile monocular visual-inertial state estimator. *IEEE Transactions on Robotics* 34, 4 (2018), 1004–1020.
[44] Morgan Quigley, Ken Conley, Brian P. Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y. Ng. 2009. ROS: an open-source Robot Operating System. In *ICRA Workshop on Open Source Software*.
[45] Anurag Ranjan and Michael J. Black. 2017. Optical Flow Estimation using a Spatial Pyramid Network. In *Proceedings of the IEEE CVPR*.
[46] Henri Rebecq, Timo Horstschäfer, Guillermo Gallego, and Davide Scaramuzza. 2016. Evo: A geometric approach to event-based 6-dof parallel tracking and mapping in real time. *IEEE Robotics and Automation Letters* 2, 2 (2016), 593–600.
[47] ITUT Recommendation. 2008. ITU-T P. 910. *Subjective video quality assessment methods for multimedia applications* (2008).
[48] Ramanujan K Sheshadri, Eugene Chai, Karthikeyan Sundaresan, and Sampath Rangarajan. 2021. SkyHAUL: A Self-Organizing Gigabit Network In The Sky. In *Proceedings of the ACM MobiHoc*.
[49] Bruce Spang, Shravya Kunamalla, Renata Teixeira, Te-Yuan Huang, Grenville Armitage, Ramesh Johari, and Nick McKeown. 2023. Sammy: smoothing video traffic to be a friendly internet neighbor. In *Proceedings of the ACM SIGCOMM*.
[50] Gary J Sullivan, Jens-Rainer Ohm, Woo-Jin Han, and Thomas Wiegand. 2012. Overview of the high efficiency video coding (HEVC) standard. *IEEE Transactions on circuits and systems for video technology* 22, 12 (2012), 1649–1668.
[51] Jarno Vanne, Marko Viitanen, and Timo D Hämäläinen. 2014. Efficient mode decision schemes for HEVC inter prediction. *IEEE Transactions on Circuits and Systems for Video Technology* 24, 9 (2014), 1579–1593.
[52] Thomas Wiegand, Gary J Sullivan, Gisle Bjontegaard, and Ajay Luthra. 2003. Overview of the H. 264/AVC video coding standard. *IEEE Transactions on circuits and systems for video technology* 13, 7 (2003), 560–576.
[53] Hao Wu, Xuejin Tian, Minghao Li, Yunxin Liu, Ganesh Ananthanarayanan, Fengyuan Xu, and Sheng Zhong. 2021. Pecam: privacy-enhanced video streaming and analytics via securely-reversible transformation. In *Proceedings of the ACM MobiCom*.
[54] Jingao Xu, Hao Cao, Zheng Yang, Longfei Shangguan, Jialin Zhang, Xiaowu He, and Yunhao Liu. 2022. {SwarmMap}: Scaling up real-time collaborative visual {SLAM} at the edge. In *Proceedings of the USENIX NSDI*.
[55] Jingao Xu, Danyang Li, Zheng Yang, Yishujie Zhao, Hao Cao, Yunhao Liu, and Longfei Shangguan. 2023. Taming Event Cameras with Bio-inpired Architecture and Algorithm: A Case for Drone Obstacle Avoidance. In *Proceedings of the ACM MobiCom*.
[56] Shichang Xu, Subhabrata Sen, and Z Morley Mao. 2020. CSI: Inferring mobile ABR video adaptation behavior under HTTPS and QUIC. In *Proceedings of the EuroSys*. 1–16.
[57] Hyunho Yeo, Chan Ju Chong, Youngmok Jung, Juncheol Ye, and Dongsu Han. 2020. Nemo: enabling neural-enhanced video streaming on commodity mobile devices. In *Proceedings of the ACM MobiCom*.
[58] Hyunho Yeo, Hwijoon Lim, Jaehong Kim, Youngmok Jung, Juncheol Ye, and Dongsu Han. 2022. NeuroScaler: neural video enhancement at scale. In *Proceedings of the ACM SIGCOMM*.
[59] Anlan Zhang, Chendong Wang, Bo Han, and Feng Qian. 2022. YuZu:Neural-Enhanced volumetric video streaming. In *Proceedings of the USENIX NSDI*.
[60] Gangfu Zhang and Hubert Chanson. 2018. Application of local optical flow methods to high-velocity free-surface flows: Validation and application to stepped chutes. *Experimental Thermal and Fluid Science* 90 (2018), 186–199.
[61] Tan Zhang, Aakanksha Chowdhery, Paramvir Bahl, Kyle Jamieson, and Suman Banerjee. 2015. The design and implementation of a wireless video surveillance system. In *Proceedings of the ACM MobiCom*.
[62] Wenxiao Zhang, Feng Qian, Bo Han, and Pan Hui. 2021. Deepvista: 16k panoramic cinema on your mobile device. In *Proceedings of the Web Conference*.
[63] Yi Zhou, Guillermo Gallego, and Shaojie Shen. 2021. Event-based stereo visual odometry. *IEEE Transactions on Robotics* 37, 5 (2021), 1433–1450.
[64] Pengfei Zhu, Longyin Wen, Dawei Du, Xiao Bian, Heng Fan, Qinghua Hu, and Haibin Ling. 2021. Detection and tracking meet drones challenge. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44, 11 (2021), 7380–7399.
[65] Shan Zhu and Kai-Kuang Ma. 2000. A new diamond search algorithm for fast block-matching motion estimation. *IEEE transactions on Image Processing* 9, 2 (2000), 287–290.

# A Video Streaming

Fig. 2 illustrates the working pipeline of the VCL and NAL in a typical video streaming system.

① **Frame and Block division**. Each video frame is categorized as either an I-frame (intra-coded), a P-frame (predictive-coded), or a B-frame (bi-predictive-coded). I-frames are self-contained and less compressed, P-frames are predicted from previous I or P frames, and B-frames are predicted based on both their preceding and succeeding I or P frames. Typically, recent codecs will adjust Group-of-picture (GoP) length (i.e., the number of frames between two I-frames) based on scene dynamics to balance compression efficiency and error resilience.

Each frame is further divided into blocks with different pixel sizes (e.g., 8×8, 16×16, 64×64) and type (I, P, B), based on content dynamics such as motion and texture. For instance, areas with rich texture or high motion, especially at the edges, typically lead to more I-frames composed of smaller I-blocks, due to the unpredictability of content changes. The frame and block divisions serve as a video codec's initial stage and bedrock, whose results significantly impact video streaming performance.

② **Intra-frame prediction** minimizes spatial redundancy in I-frames by dividing images into compact blocks and predicting pixel values using neighboring pixels[34]. Different prediction modes, such as horizontal, vertical, and diagonal, are employed to determine the most appropriate prediction for each block. The residual, which is the difference between predicted and actual pixel values, is then encoded and stored. By exploiting the spatial correlation between neighboring pixels, intra-frame prediction significantly reduces the amount of data required to represent an I-frame.

④ **Inter-frame prediction** diminishes temporal redundancy between consecutive frames by employing motion estimation and compensation to predict pixel values in P- or B-frames, using information from previously encoded frames. As a result, most raw frame contents will not be transmitted. Like intra-frame prediction, frames are divided into blocks,

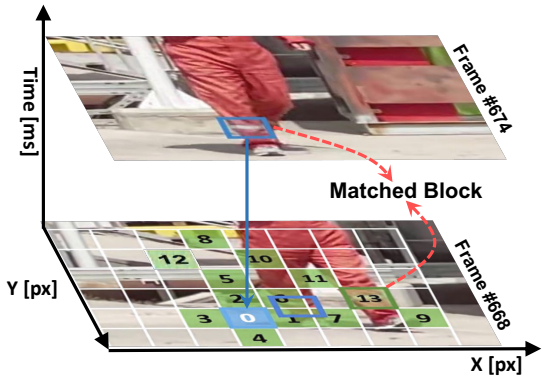



**Figure 17.** An illustration of 13 block matching attempts to find a matched block in inter-frame prediction.

and motion vectors are calculated for each block by searching for the most similar block in the reference frame. These motion vectors and the residual error between predicted and actual blocks are encoded, achieving substantial data reduction while preserving image quality. Fig. 17 illustrates a real-world case study for inter-frame prediction where a block in frame #674 conducts 13 block matching attempts to find its matched block in the previous I-frame (#668).

④ **Quantization and entropy coding** further reduce visual and coding redundancy. The former exploits human visual insensitivity to certain details, replacing fine data with coarser quantified values. Additionally, it adapts quantization parameters (indicating compression ratios) based on application needs, ensuring high visual importance RoIs maintain quality and accurate reconstruction[50]. The latter leverages lossless compression of 0-1 bit-streams via Shannon, Huffman, or arithmetic coding[14, 52].

After VCL, the NAL priorities data delivery based on network conditions and back-end application requirements by $(i)$ segmenting the video bit-streams into smaller packets; and $(ii)$ adding metadata to each packet to indicate its content and importance. Eventually, NAL transmits prioritized bit-streams over network or stores them in memory.

In the past decades, numerous mobile video streaming solutions have been developed. However, the core insight behind reducing video content redundancy is consistent across these standards, with differences primarily arising from the specific algorithms employed and block division granularity for intra-frame and inter-frame prediction. This similarity suggests that E-Cube can be integrated into various video streaming frameworks to enhance their performance.

# B Event-based Vision

## B.1 Event Camera Primer

Event cameras are advanced sensors different from traditional cameras. They have smart pixels, similar to the photoreceptor cells in biological retinas. Each pixel can trigger events individually. Instead of capturing images at set intervals, event cameras detect changes in brightness for each pixel, providing a continuous stream of events with *microsecond* resolution.

As shown in Fig. 18, let $t_{k-1}$ be the last time when an event fired at a pixel location $\boldsymbol{p}_k$, and $I_{k-1} = I(\boldsymbol{p}, t_{k-1})$ be the intensity level at the such pixel at time $t_{k-1}$. A new event is fired at the same pixel location at time $t_k$ once the difference between the intensity $I_{k-1}$ and $I_k$ is larger than a pre-defined threshold $C > 0$. In other words, an event is generated if $\|I(\boldsymbol{p}, t_k) - I(\boldsymbol{p}, t_{k-1})\| \geq C$ (positive event, $p_k = 1$) or $\|I(\boldsymbol{p}, t_k) - I(\boldsymbol{p}, t_{k-1})\| \leq -C$ (negative event, $p_k = -1$). To better highlight what happens across the entire sensor, we compare the output of an event camera with a conventional camera in Fig. 19. As seen, a conventional camera captures frames at a fixed rate and with obvious motion blur; an event camera continuously outputs the polarity of brightness changes in the form of a spiral of events in space-time.

According to the above analysis, event cameras offer a more convenient way to establish pixel associations compared to traditional cameras. Since event cameras only process scene changes, they inherently emphasize the differences between consecutive frames. This focus on changes facilitates the establishment of direct associations between pixels across different frames, as it highlights the areas of the scene that are most relevant for tracking motion or other dynamic elements. By prioritizing these informative regions and capturing them with a high temporal resolution, event cameras simplify the task of associating pixels and tracking their movement over time, which is complementary to the characteristics of frame-based cameras.

## B.2 Event-based Motion Vector

We set the first partial derivatives of $\Sigma_e$ as: $\Sigma_{e_x} = \partial\Sigma_e/\partial x$ and $\Sigma_{e_y} = \partial\Sigma_e/\partial y$. For $\Sigma_e$, we have:

$$\Sigma_e(\mathbf{p} + \Delta\mathbf{p}) = \Sigma_e(\mathbf{p}) + \nabla\Sigma_e^T \Delta\mathbf{p} + o(\|\Delta\mathbf{p}\|),$$

with $\nabla\Sigma_e = (\partial\Sigma_e/\partial x, \partial\Sigma_e/\partial y)^T$.

The sub-functions of $\Sigma_e$ depend on a single variable, either $x$ or $y$. Since time is a strictly increasing function, $\Sigma_e$ has a nonzero derivative at any point. This allows us to apply the inverse function theorem around a specific location, represented as $\mathbf{p} = (x, y)^T$:

$$\frac{\partial\Sigma_e}{\partial x}(x, y_0) = \frac{\mathrm{d}\Sigma_e|_{y_0}}{\mathrm{d}x}(x) = \frac{1}{v_x(x, y_0)},$$
$$\frac{\partial\Sigma_e}{\partial y}(x_0, y) = \frac{\mathrm{d}\Sigma_e|_{x_0}}{\mathrm{d}y}(y) = \frac{1}{v_y(x_0, y)}.$$

$\Sigma_e|_{x_0}$, $\Sigma_e|_{y_0}$ being $\Sigma_e$ restricted respectively to the center point of a block $y = y_0$ and $x = x_0$. The gradient $\nabla\Sigma_e$ can then be written as:

$$\nabla\Sigma_e = \left(\frac{1}{v_x}, \frac{1}{v_y}\right)^T. \tag{3}$$

The vector $\nabla\Sigma_e$ measures the rate and the direction of change of time with respect to the space, and its components

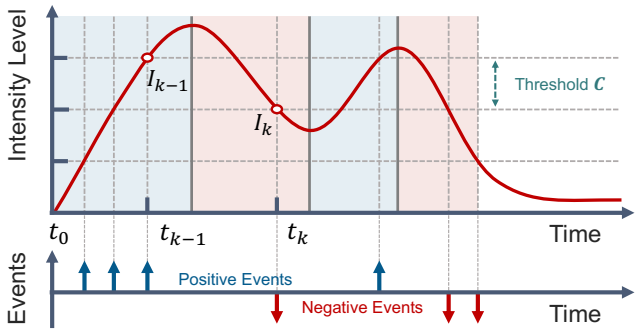



**Figure 18.** The working principle of event camera.

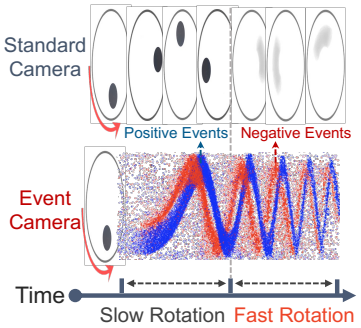



**Figure 19.** Event *vs* Frame Camera.

are also the inverse of the components of the velocity vector estimated at **p**. E-Cube also leverages the event de-noising algorithm proposed by E-Flow[13] to optimize the accuracy of Eq.3 in practice.

# C Event-IP Hybrid Motion Compensation for RoI Segmentation

## C.1 Baseline: Motion Compensation for Events

Motion compensation transforms the pixel location of an event $e_i$ from $t_i = t_0 + \Delta t$ back to $t_0$ using the camera's motion, as estimated by IMU readings[19]. There are two cases:
$(i)$ For background-triggered events, the compensated location $\hat{\mathbf{p}}_0$ at $t_0$ aligns with its actual location $\mathbf{p}_0$ since changes are due to drone movement alone.
$(ii)$ For RoI-triggered events, the compensated location $\hat{\mathbf{p}}_0$ at $t_0$ will not align with $\mathbf{p}_0$, as it does not factor in the RoI's own movement. Fig. 20b shows that in the frame of compensated events, RoI events appear scattered while background events form a sharp contour. Such differences can be leveraged to aid in RoI segmentation.

Consider a point $P$ in the camera coordinate system, associated with an event $e_i$ at $t_i$, having a spatial location $P_C(x, y, z)$. The event's pixel location $P_{uv}(u, v, 1)$ is determined by the pinhole projection model:

$$P_{uv} = \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \frac{1}{z} \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \frac{1}{z} K P_C, \tag{4}$$

where $K \in \mathbb{R}^{3\times3}$ is the camera intrinsic parameter matrix.

To compensate for camera movement, we use the camera's rotation matrix $R$ and translation vector $t$, estimated by IMU readings. This helps us calculate the adjusted point $P'_C$ as $P'_C = RP_C + t$. By combining these equations, we estimate $e_i$'s previous pixel location $P'_{uv}$ at $t_0$ as:

$$\begin{aligned} P'_{uv} &= \frac{1}{z'} K P'_C = \frac{1}{z'} K \left(RP_C + t\right) \\ &= \frac{1}{z'} K \left[ R\left(zK^{-1}\right)\left(\frac{1}{z} K P_C\right) + t \right] \\ &= \frac{z}{z'} K R K^{-1} P_{uv} + \frac{1}{z'} K t \\ &\approx K R K^{-1} P_{uv} + \frac{1}{z'} K t. \end{aligned} \tag{5}$$

The baseline method[19], omits both the camera's translational motion $t$ and the term $\frac{1}{z'}Kt$, since the depth $z$ ($z'$) is unknown in a monocular setup. However, when the target depth $z$ is small or the camera translation $t$ is large, such simplification leads to significant errors in compensation.

## C.2 E-Cube Solution: Event-MV Hybrid Compensation

To mitigate the large compensation errors in baseline, in E-Cube, we resort to leveraging the intermediate motion vectors (MV) estimated through the *inter-frame prediction* in video streaming to refine the calculation of $P'_{uv}$. In particular, for the current frame being coded, E-Cube identifies the areas in previously coded frames where motion vectors display consistent patterns, indicating stability. The stability can be evaluated by [51], and the selected regions are typically static background regions. We denote these regions as $\mathcal{B}$ and the motion vectors linked to each pixel in $\mathcal{B}$ as $m_{uv}$. The Event-IP hybrid motion compensation procedure then proceeds based on the following equation:

$$P'_{uv} = \begin{cases} P_{uv} + m_{uv}, & \text{if } (u,v) \in \mathcal{B} \\ KRK^{-1}P_{uv}, & \text{if } (u,v) \notin \mathcal{B} \end{cases} \tag{6}$$

## C.3 Put Together: Fine-grained RoI Detection

Once we've attained accurate motion compensation for the event data, we generate an event-count image $\mathcal{N}$ with each pixel $\mathcal{N}_{uv}$ reflecting the number of compensated events on this pixel. A practical image $N$ is shown in Fig. 20d.

We further examine the event distribution to distinguish different types of events using a time-based solution. The more uniformly the generation time of all events whose compensated location is $N_{uv}$, the more likely these events are background-triggered, as only moving RoIs will bring an additional time-cluttered event burst on specific pixels.

**Event-Time Image** $T$**.** We define the event-time image (Fig. 20e) $T$ with each pixel calculating the average time of events in $N_{uv}$ as follows:

$$T_{uv} = \frac{1}{N_{uv}} \sum e_i(t) : e'_i(\mathbf{p}) = (u, v). \tag{7}$$

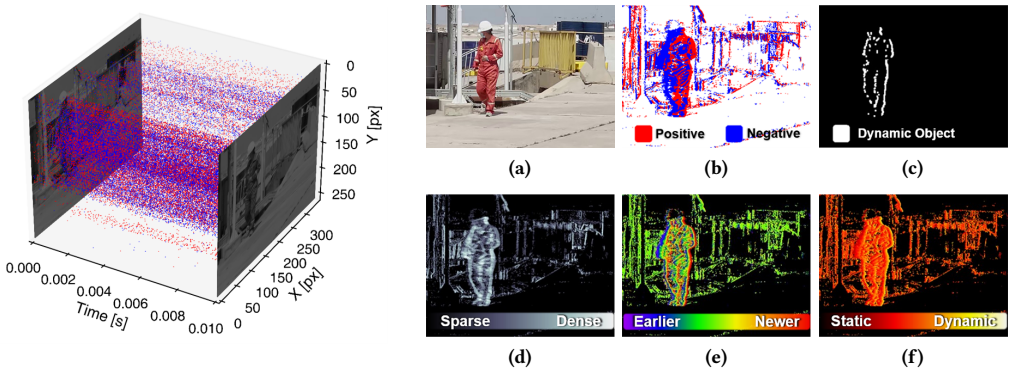



**Figure 20.** An example of leveraging E-Cube for RoI segmentation. **Left**: 3D plot of event data (within 10ms). **Right**: (a) An RGB image; (b) Generated event frame after motion compensation which wraps the pixel location of each event in the time window back to $t_0$; (c) RoI segmentation result; (d) Event-count image; (e) Event-time image; (f) Normalized timestamp image.

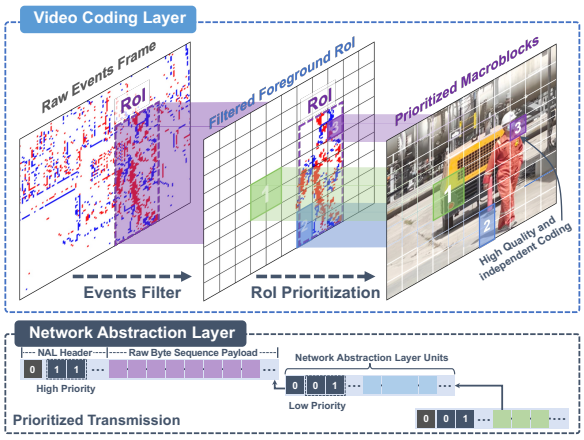



**Figure 21.** Illustration of event-assisted RoI segmentation and prioritized transmission in E-Cube.

**Normalized Mean Time Stamp Image** $\rho$ (Fig. 20f). Finally, we calculate the normalized mean time stamp image $\rho$:

$$\rho_{uv} = \frac{T_{uv} - \bar{T}}{\Delta t}. \tag{8}$$

Once a threshold $\tau$ is defined, pixels with $\rho_{uv} > \tau$ can be identified as part of the RoIs. After thresholding, the binary image will undergo morphological operations to eliminate noise and output the final segmentation results, as shown in Fig. 20c.

## D RoI-Prioritized Video Streaming

After obtaining crucial RoIs, E-Cube will ensure they are (*i*) encoded in high precision, (*ii*) transmitted in high priority, and (*iii*) decoded independently for back-end tasks. This would allow back-end applications to prioritize important video data even in poor link conditions with limited bandwidth. E-Cube's design spans VCL and NAL two layers.

**VCL: Adaptive Encoding.** E-Cube doubles the bitrate for RoIs compared to non-RoI areas, ensuring RoIs' encoding quality. To achieve this, E-Cube adopts half of the quantization parameter (than that is used in regular regions) on RoI regions during the quantization encoding in VCL.

**VCL: Independent Decoding Support.** For enhanced stability, RoIs in E-Cube are designed for independent decoding at the back-end application layer, avoiding dependence on coded blocks from other regions. E-Cube uses a slice/tile mechanism from video coding standards, facilitating the partitioning of video frames to improve coding and transmission efficiency. Each slice/tile maintains coding and decoding independence. In E-Cube, video frames are segmented so that different RoIs are contained within these independent slices/tiles. Additionally, E-Cube fine-tunes the slice/tile areas to align with the shapes of the RoIs for optimal encoding.

**NAL: Prioritized Transmission.** E-Cube further enhances the reliability of RoI packets during network transmission by prioritizing them. This prioritization is implemented using the Supplemental Enhancement Information (SEI) feature of video coding standards. Generally, SEI provides key data for video packets, such as their importance, priority, and data associations. As shown in the NAL layer of Fig. 21, in E-Cube, RoI packets are marked with a high-priority symbol, usually "11", in their SEI metadata during conversion into bit-streams. This ensures they receive preferential treatment in transmission, reducing the likelihood of corruption and loss.

## E E-Cube's On-Chip Implementation

We first implement E-Cube on a Zynq-7020 chip, as illustrated in Fig. 22.

• **PL:** We have developed dedicated logic circuits on the FPGA for event camera's pixels, allowing for independent processing of each event as it occurs. Building on this, E-Cube accelerates event-related operations that are amenable to parallel and pipeline processing, such as event data denoising, acquisition of motion vectors (refer to §5.1 in submission), profiling of temporal correlations (refer to §5.2 in submission), and block merging (refer to §5.3 in submission). These modules are plugged into the video streaming pipeline to enhance its operational efficiency.

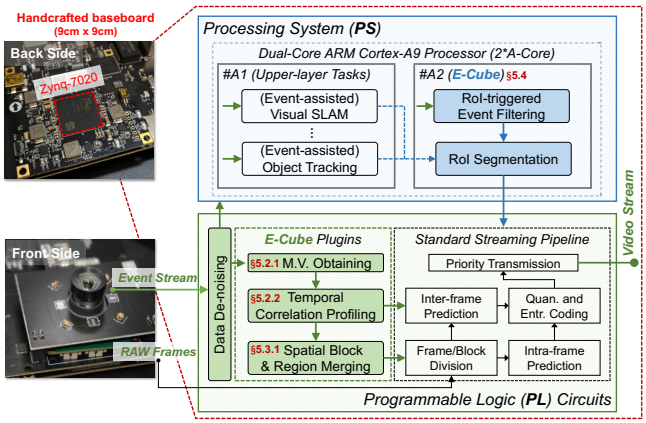



**Figure 22.** E-Cube's implementation on a Zynq chip.

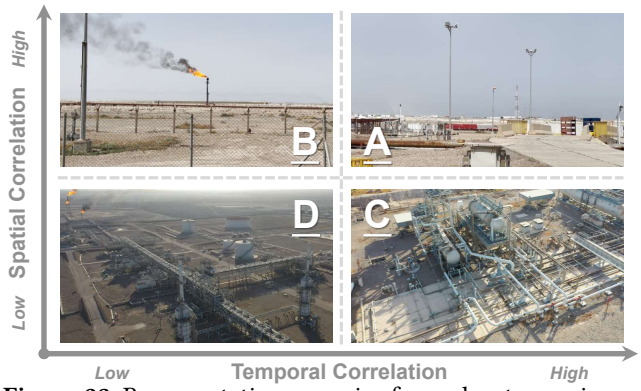



**Figure 23.** Representative scenarios for each category in our dataset.

**Table 5.** Dataset Summary

| Class (Description) | Temporal / Spatial Corr. | #Video Clips | #Total Frames |
|---|---|---|---|
| **A** (Normal) | High / High | 2,343 | 1,188,600 |
| **B** (Dynamic) | Low / High | 1,982 | 1,053,200 |
| **C** (Dynamic) | High / Low | 2,541 | 1,276,300 |
| **D**(Highly Dynamic) | Low / Low | 2,762 | 1,563,800 |

• **PS:** We first implement a core-isolation strategy to segregate the processing resources of #A2 from PS, reducing the impact of CPU scheduling on task execution. We realize it by building a Linux OS with boot parameter `isolcpus=<cpu #A2>`. On #A2, we run tasks such as event filtering and fine-grained RoI segmentation (§5.3 in submission), which require frequent memory access and are not readily implementable on FPGA. Current drone systems often utilize event data for high-level tasks like visual-odometry (VO[39]), visual-inertial-odometry (VIO[43]), and high-speed target tracking[55]. E-Cube could repurpose the intermediate outcomes of these algorithms (as indicated by #A1 ⇢ #A2) for acceleration.

• **Data flow in-between:** We further leverage the physical-level direct memory access (DMA) technique[9] to transmit intermediate data among PL, #A1, and #A2. Compared with network-level solutions such as PL-PS ethernet interface[3] and OpenAMP[10], DMA ensures data interaction processes would not be interrupted by CPU scheduling.

# F Evaluation Details

## F.1 Dataset Description

We list the public dataset VisDrone, and the trajectories we used in the handcrafted oil-field dataset in Table 3. We select representative trajectories with different difficulty levels (in terms of environmental dynamics, path length, drone flight speed, etc..) Fig. 23 shows four representative scenes in class **A** (normal), **B** (temporal dynamic), **C** (spatial dynamic), and **D** (highly dynamic), respectively. A dataset summary is presented in Table 5.

## F.2 Parameter Selection

For a fair comparison, we ensure each codec uses the same intra-frame prediction method, i.e., JPEG-2000[34] with quantization parameters QP=25 for RoIs and QP=50 for other regions. A higher QP value preserves more detail in the compressed image, leading to higher quality but also a larger file size. Typically, a QP value between 1-30 is used for high-quality images/regions.

For inter-frame prediction, we use the TZ-search[41] based on prismatic search templates with *step size*, *maximum searching range* (indicating maximum matching times for each block), and *early termination similarity criteria* set as 3, 40, and 94% respectively. We further adopt the Hoffman algorithm in the entropy coding process and disable the QoE-oriented adaptive bitrate (ABR) functions in NAL, both before and after E-Cube's integration.