An Asynchronous Intensity Representation for Framed and Event Video Sources

Andrew C. Freeman
acfreeman@cs.unc.edu
University of North Carolina
Chapel Hill, North Carolina, USA

Montek Singh
montek@cs.unc.edu
University of North Carolina
Chapel Hill, North Carolina, USA

Ketan Mayer-Patel
kmp@cs.unc.edu
University of North Carolina
Chapel Hill, North Carolina, USA

ABSTRACT

Neuromorphic "event" cameras, designed to mimic the human vision system with asynchronous sensing, unlock a new realm of high-speed and high-dynamic-range applications. However, researchers often either revert to a framed representation of event data for applications, or build bespoke applications for a particular camera's event data type. To usher in the next era of video systems, accommodate new event camera designs, and explore the benefits of asynchronous video in classical applications, we argue that there is a need for an asynchronous, source-agnostic video representation. In this paper, we introduce a novel, asynchronous intensity representation for both framed and non-framed data sources. We show that our representation can increase intensity precision and greatly reduce the number of samples per pixel compared to grid-based representations. With framed sources, we demonstrate that by permitting a small amount of loss through the temporal averaging of stable pixel values, we can reduce our representational sample rate by more than half, while incurring a drop in VMAF quality score of only 4.5. We also demonstrate lower latency than the state-of-the-art method for fusing and transcoding framed and event camera data to an intensity representation, while maintaining 2000× the temporal resolution. We argue that our method provides the computational efficiency and temporal granularity necessary to build real-time intensity-based applications for event video.

CCS CONCEPTS

• **Computing methodologies** → Image compression; **Image representations**; *Image processing*.

KEYWORDS

event representation, event video, video processing, event vision

ACM Reference Format:
Andrew C. Freeman, Montek Singh, and Ketan Mayer-Patel. 2023. An Asynchronous Intensity Representation for Framed and Event Video Sources. In *Proceedings of the 14th ACM Multimedia Systems Conference (MMSys '23), June 7–10, 2023, Vancouver, BC, Canada.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3587819.3590969

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
MMSys '23, June 7–10, 2023, Vancouver, BC, Canada
© 2023 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 979-8-4007-0148-1/23/06…$15.00
https://doi.org/10.1145/3587819.3590969

1 INTRODUCTION

In this paper, we introduce a method for transcoding both framed video and modern event camera data to a novel asynchronous video representation. Our representation enables rate-distortion tradeoffs to be systematically explored based on the temporal stability of individual pixels. The approach is computationally efficient, lossy compressible, and suitable for real-time vision applications.

For **framed video** sources, our method unlocks potentially useful per-pixel tradeoffs between precision and sample rate. Traditional synchronous framed video representations have a fixed and uniform sample rate. Consequently, a spatial region of a video which is dynamic (changing greatly) has the same number of intensity samples as a spatial region which is relatively static (unchanging). Thus, there is much data redundancy in the raw representation, which is currently only addressed at the level of the compressor. Furthermore, the precision of a framed video is bounded by the maximum representable value in a single image frame, typically 8-12 bits per color channel. We demonstrate that framed video when transcoded into our proposed asynchronous representation unlocks a tradeoff between precision and sample rate that allows us to represent temporally stable intensities with greater precision at lower sample rates.

In addition, we address a fundamental representational weakness of **event sensors**. These sensors do not record traditional image frames; rather, each pixel independently generates a sequence of *change events*, where a bespoke time-based event triggers when the log intensity of the pixel increases or decreases by a certain amount. This technique allows for extreme temporal precision (on the order of microseconds) and high dynamic range, but prevents the sensor from capturing data where the intensity is unchanging or changing slowly. Thus, many users of these cameras augment event sensors with frame-based image capture to fill in the informational gaps (e.g., DAVIS cameras [18, 28]). However, these framed images tend to exhibit motion blur, and they are typically processed separately from the event data due to the slow computational performance of fusing the information across modalities. Additionally, each event expresses the intensity change *relative to* a previous event, so this raw representation does *not* support lossy compression or expose a rate-distortion tradeoff. These sensors are rapidly gaining traction for vision tasks and high-speed robotics, but their extraordinary data rates (up to millions of events per second) and ever-increasing spatial resolutions (now reaching 1080p) are not adequately addressed from a systems perspective. Instead, most research applications merely convert event data to a number of frame-based representations. These representations severely reduce the temporal resolution of the data, diminishing a major benefit to event sensing.

We present the following analogy to motivate our work. Imagine that traditional cameras from different companies all had their own proprietary raw data formats, with no common representation between them. Now imagine that many video applications could only operate on raw data from the cameras of a particular company, and that all other video applications could operate only on PNG image sequences derived from cameras' raw representations. That is effectively how the event camera ecosystem looks today: there is no codec equivalent to H.26X [49] for event video, meaning that event camera systems are not employing any significant lossy compression and are hand-built for particular cameras' raw data formats. We argue that to build the most effective real-time event video systems, the application-level video representation should *also* be asynchronous, and it should enable rate-driven lossy compression. Furthermore, we emphasize that framed video that with either a stationary camera (i.e., surveillance video) or with a sufficiently high frame rate relative to the captured motion, can equally be thought of under an event-based camera model.

We propose a unified video representation whose fundamental data type is an asynchronous **intensity event**. Our events do not represent intensity *change*, as opposed to the representations for event sensors; rather, each event *directly* encodes an intensity value. We can thus transcode both framed and event video data to our single representation, allowing us to address the above weaknesses with these data sources. We summarize our contributions as follows.

- Introduce a novel asynchronous video representation and transcoder for both framed and non-framed data sources.
- Evaluate the speed and visual quality of our representation on framed color video sources, demonstrating significantly higher precision (up to 14 bits, as opposed to 8 bits for the source videos) and a predictable rate-distortion curve. By permitting a small amount of loss, we can reduce the median event rate of our representation by 54.3% with a median drop in VMAF quality of only 4.5.
- Evaluate the effect of various transcoder parameters on rate and quality.
- Evaluate the speed and rate properties of our representation on event camera sources with and without additional framed data, and show the qualitative differences in appearance at various rates. When targeting a temporal precision of 500 samples per second or higher, our method incurs lower latency than the state-of-the-art method for frame/event fusion. In fact, our method maintains *microsecond* temporal resolution for intensities that are changing. We also release the first fully open-source implementation of this state-of-the-art fusion method, with numerous improvements and optimizations for high performance.
- Release our open-source transcoder framework, including utilities for visual playback of our video representation.

2 RELATED WORK

2.1 Framed Video

Under the framed video capture paradigm, the camera exposes its entire imaging plane for the same amount of time, producing a series of discrete 2D images, as in Fig. 1b. Although framed video representations benefit from decades of compression research for a variety of storage needs and streaming applications, they have many limitations.

For one, dynamic range is inseparable from the source's capture sensitivity, providing a practical upper limit to the intensity range of the representation [45]. Often, one records video at a high bit depth (up to 12 bits), then masters the video for viewing on both SDR and HDR displays [24]. The mastering process often involves adjusting brightness levels to show desired levels of detail. In this case, a higher-precision representation allows a filmmaker to have more flexibility in adjusting the dark and bright regions of the image during post-processing. Additionally, high precision helps to diminish the apparent edges within regions of intensity gradient.

Secondly, since pixel samples have a fixed frame rate, there is no inherent pixel averaging of stable video regions across multiple frames, causing high-contrast and high-speed scenes to suffer greatly from noise in the dark regions of the video. Even the advent of extremely high frame rate cameras did not bring with it a new paradigm of video representations [29]. Rather, these cameras produce large, spatio-temporally redundant framed video files using standard codecs. While some video container formats allow for variable frame rate video, the pixel samples remain temporally organized across frames [10], and software support for such videos is limited.

Finally, most computer vision algorithms depend on a framed interpretation of the world, digesting video as a sequence of frames with convolution [38]. Despite great progress in the efficacy of such systems for tasks like object recognition, object tracking, and scene reconstruction, it is disingenuous to suggest that their structures emulate the human eye's continuous and dynamic sensing [18].

2.2 Event Video

More recently, we have seen the adoption of asynchronous event-based cameras that aim to address the above weaknesses of frame-based video capture and representation. Rather than output frames, these sensors output time-based events using an Address Event Representation (AER), wherein an event expresses a discrete data point for a particular pixel location and point in time. Here, we examine several event-based cameras and their corresponding AER data formats.

2.2.1 DVS. Dynamic Vision Sensors (DVS) prioritize localized capture rate over global accuracy. DVS cameras (Fig. 1a) continuously measure the log intensity, $\widetilde{L} = \ln(I)$, incident upon a pixel $\langle x, y \rangle$, until the brightness changes such that $\widetilde{L}(x, y, t) - \widetilde{L}(x, y, t - \Delta t) = p\theta$, where $p \in \{-1, +1\}$ is the polarity of the brightness change, $\theta > 0$ is the threshold, and Δt is the time elapsed since the pixel last recorded such a brightness change [18]. When the brightness changes across the threshold $p\theta$, the pixel outputs an asynchronous *event*, represented by the tuple $\langle x, y, p, t \rangle$. The parameter θ is determined by a variety of camera gain settings, together with the effects of scene illumination. It is thus not easily determinable by the camera user, and may change drastically during the course of a video recording.

The DVS approach has significant advantages. First, change events are quickly fired by pixels without incurring the frame latency of a synchronous sensor. Second, a high dynamic range is achieved due to logarithmic compression of the intensity. Third, when the intensity at a pixel is relatively stable, few events are fired,

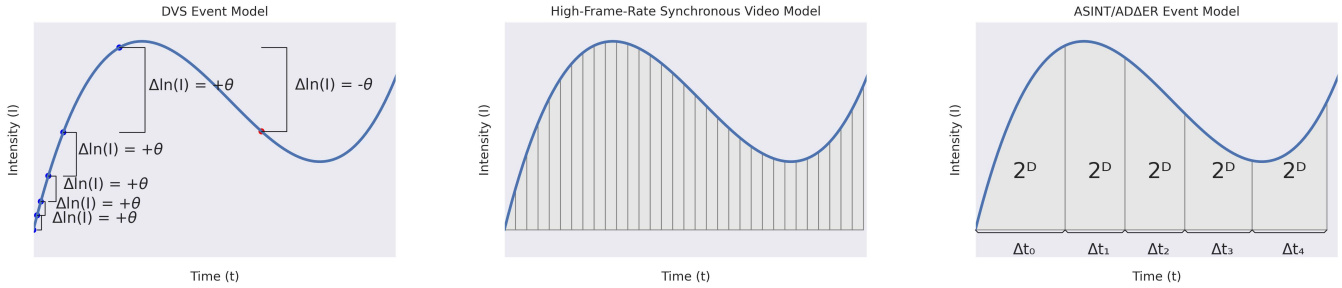


Figure 1: Comparison of video models. (a) A new DVS event is fired whenever ln(I) changes by the threshold θ. Since differences are sensed, intensity reconstruction is challenging. (b) A framed video contains one sample for each pixel at synchronous points in time, with fixed lengths between samples. (c) An ADΔER event indicates that the accumulated intensity units reach 2^D. D controls the per-pixel sensitivity, which we show as constant here for illustrative purposes. In our ADΔER transcoder, however, D adapts to changes in the intensities being integrated, to reduce the event rate.

thereby conserving bandwidth. The key weaknesses of DVS are poor pixel noise characteristics and the difficulty of reconstructing actual intensity values. The pixel values suffer from noise because log conversion is performed by a subthreshold MOS transistor, and the sensed value is an instantaneous voltage sample as opposed to an accumulation over an integrating interval. Due to the differencing nature of sensing, the high-quality reconstruction of actual intensity values becomes difficult as noise accumulation leads to drift. Because of this shortcoming, DVS sensors are typically used when capturing the edges of moving objects is of primary importance. This primarily includes robotics applications where fast response times are necessary and human visual quality is not important, such as object detection and tracking, gesture recognition, and SLAM (simultaneous localization and mapping). On the other hand, numerous learning-based [35, 36, 40] and direct [37, 39, 47] methods for framed reconstructions of DVS data are proposed in the literature, but these are limited to reconstructing 100-1000 frames per second even on state-of-the-art hardware and low-resolution (240 × 180 pixels) sensors [36].

2.2.2 DAVIS and ATIS. Dynamic and Active Pixel Vision Sensors (DAVIS) aim to improve the usability of DVS technology by co-locating active (APS) pixels alongside the DVS pixels on the sensor chip [18]. These active pixels capture discrete frames at a fixed rate, just as a traditional camera. APS data can improve the quality of intensity reconstructions by preventing drift in DVS accuracy [9, 30, 31, 39]. The resulting intensity reconstructions can exhibit very high dynamic range, and are useful in vision applications where broader scene understanding is important.

Asynchronous Time-based Image Sensors (ATIS) take a different approach than DAVIS for capturing intensities alongside DVS events. Integrator subpixels on these sensors express absolute intensities by outputting two AER-style events, where a short time between the events corresponds to a bright pixel while a long time between events corresponds to a dark pixel [18]. Although these intensity measurements are accurate, they trigger only when the DVS subpixel records an intensity change event [33]. Thus, ATIS does not provide absolute intensities for every pixel at a given point in time, but only at high-contrast moving edges.

ATIS and DAVIS aid vision tasks, such as object detection, by augmenting contrast-based events with absolute intensity measurement. However, they produce two distinct streams of data, and require the development of a sophisticated supporting software infrastructure to fuse the two streams to fully take advantage of their capabilities. For DAVIS in particular, the intensity frames help one to focus the camera's lens, use classical vision algorithms alongside event-based algorithms for comparison or augmentation, and achieve better performance in intensity reconstruction through the fusion of event and framed data. DAVIS has emerged as a prominent event-based sensing technology for research, while ATIS has not seen widespread commercial adoption. In this paper, we focus our attention on the DAVIS sensor.

2.2.3 Event Video Representations. While some applications process DVS events directly with spiking neural networks (SNNs) [5, 12, 13], many resort to grid-based representations of the events, which are more amenable to classical vision pipelines and do not require specialized neuromorphic network hardware [2, 6, 19, 27, 34, 48]. We outline several of these grid-based representations in Tab. 1, highlighting the tradeoffs researchers make between temporal precision and ease of processing.

Although researchers have explored rate control schemes for DVS streams, the proposed systems perform rate adaptation only at the application level [20], or at the camera source by adjusting biases for θ [7, 11], rather than exploring a rate-distortion control scheme for event representation and transmission. In a DAVIS camera, the stream of APS frames is a separate representation from the DVS event stream. Researchers may fuse the active and dynamic information as discussed above, but often still fall back to framed representations to incorporate the disparate streams. This approach temporally quantizes the events, discarding much of the high-rate advantage of DVS capture. While Banerjee et al. uniquely propose a system for application-driven DAVIS rate control on a networked client [3, 4], their scheme does not unify the event and the framed

Table 1: Comparison of event video representations. The AD∆ER representation in transcode mode (ii) achieves the most flexibility for event-based applications, while it can also be trivially transformed to a frame-based intensity representation for use in classical applications.

Name	Representation	Continuous?	Full-sensor intensities?	Transformable to DVS?	Compatible with CNNs?	Compatible with SNNs?
DVS	$\langle x, y, p, t \rangle$	✓		–		✓
DAVIS	$\langle x, y, p, t \rangle$, $[w, h, I]$	DVS only	APS only	–	APS only	DVS only
DAVIS image fusion	$[w, h, I]$		✓		✓	
DVS event frames	$[w, h, \Delta L]$				✓	
DVS time surfaces	$[w, h, t]$				✓	
DVS voxel grid	$[w, h, n, \sum p \text{ or } \frac{\partial L}{\partial t}]$				✓	
ATIS	$\langle x, y, p, t \rangle$, $\langle x, y, I, t \rangle$	✓		–		✓
AD∆ER event frames	$[w, h, \frac{2^D}{\Delta t}]$		✓		✓	
DAVIS (i) → AD∆ER	$\langle x, y, D, \Delta t \rangle$	✓	✓			✓
DAVIS (ii) → AD∆ER	$\langle x, y, D, \Delta t \rangle$	✓	✓	✓		✓
DVS (iii) → AD∆ER	$\langle x, y, D, \Delta t \rangle$	✓		✓		✓

information under a single representation, meaning that the streams still require separate application-level logic.

Additionally, the application ecosystem for event vision is fragmented between numerous raw data representations. Data sets are variously distributed in a human-readable text format [22, 28], a `.bag` format for the Robot Operating System [22, 23, 28, 36], an `.hdf5` format for learning-based Python tasks [21, 22], an `.aedat2` format designed for operability with the jAER software [21, 22], or a `.mat` format for use in MATLAB applications [31, 47]. Event camera manufacturers iniVation and Prophesee each have their own software suites providing applications, but each operates only on their proprietary camera data formats. While some tools exist to transform raw camera data to one of the above intermediate representations, none of them fundamentally changes the underlying data types. We argue that, if one must already transform the raw data to prepare it for an application, we should instead transcode it to a simple, fast, compressible, non-proprietary data representation which is both application agnostic and camera agnostic. Then, applications may ingest this single representation, and support for future event sensing modalities can be implemented in the data transcoder rather than in individual applications.

3 PROPOSED REPRESENTATION

3.1 Motivation

While we recognize the utility of a DVS-style representation for contrast-based events, such a format is not amenable to representing absolute intensities. With a DVS-style approach, inducing loss is feasible only in terms of mitigating sensor noise and allowing an application to ignore events it deems unimportant. That is, removing a DVS event in a given pixel's stream will necessarily affect the interpretation of later events in that pixel's stream, since each event conveys intensity information *relative* to a previous event. For the same reason, even the DVS-style ATIS representation, which can encode intensity information, is ill-suited as a general, adaptable, and compressible representation. Specifically, these representations do not support receiver-driven lossy compression of their events.

In response, we propose the **A**ddress, **D**ecimation, Δt **E**vent **R**epresentation (**AD∆ER**, pronounced "adder") format as the "narrow waist" representation for asynchronous video data. This representation draws from a proposed Asynchronous Integration (ASINT) camera model [16, 41, 42]. As illustrated in Fig. 1c, a pixel $\langle x, y, c \rangle$ continuously integrates light, firing an AD∆ER event $\langle x, y, c, D, \Delta t \rangle$ when it accumulates 2^D intensity units (e.g., photons), where D is a *decimation threshold* and Δt is the time elapsed since the pixel last fired an event. We measure t in clock "ticks," where the granularity of the clock tick length is user adjustable. Unlike ATIS events, a *single* AD∆ER event directly specifies an intensity, I, by $I = \frac{2^D}{\Delta t}$. The key insight of this model is **the dynamic, pixelwise control of** D during transcode. Raising D for a pixel will decrease its event rate, while lowering D will increase its event rate. With this multifaceted D control, we can ensure that pixel sensitivities are well-tuned to scene dynamics, avoiding the rate and compression issues of the ATIS model while maintaining the strengths of intensity events.

Although related work focused on emulating the proposed ASINT sensor and developing early compression schemes and applications [15–17], here we distinguish the function of the sensor from its representation. That is, we look to buffered processing techniques to encode optimal AD∆ER representations for an arbitrary set of inputs, and we posture AD∆ER as a transcoded representation for *non*-ASINT sources. With this representation, we have a unified format for which to build compression schemes and applications targeting a wide variety of camera technologies. With source-specific transcoder modules and an abstract AD∆ER library, we can cast video from arbitrary camera sources (both framed and non-framed) into the single, unified AD∆ER data representation.

3.2 AD∆ER Transcoder Fundamentals

To explain how we transform data into AD∆ER, we must first describe our novel pixel model and parameters common to all transcode types.

3.2.1 Event Pixel List Structure. We will first examine the integration of a single AD∆ER pixel model. We represent an AD∆ER pixel

state with a linked list. Each node in the list consists of a decimation factor, D, an intensity integration measurement, I, and a time interval measurement, Δt. The connections between nodes carry an ADΔER event representing the intensity that the parent node integrated *before* the child node was created. When initializing a new list, we set D to be the maximal value possible to represent the first intensity we intend to integrate. To maintain precision, the values I and Δt in the list are floating point numbers. Since ADΔER is source-agnostic, we interpret the incoming values as generic "intensity units", rather than tying them to a real-world measure of intensity, such as photons, and we can scale the input values of individual data sources according to our needs.

Let us walk through a visual example of this list structure in Fig. 2. Suppose our first intensity to integrate is 101. We initialize our head node with $D = \lfloor \log_2(101) \rfloor = 6$ in Fig. 2a.

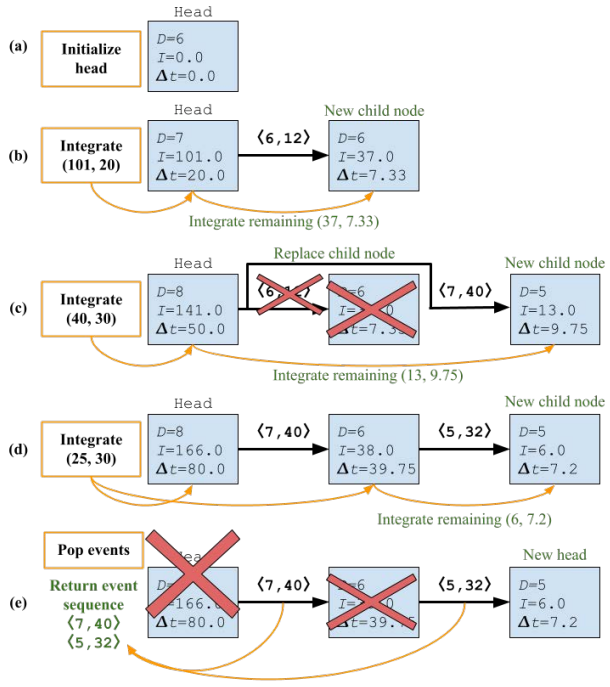


Figure 2: Structure of event pixel lists under continuous integration.

Now, suppose we integrate the intensity 101, spanning 20 time units (ticks). The head only accumulates 64/101 of the intensity units before saturating its $2^D = 64$ integration. Thus, the time spanned for this partial integration is $(64/101) \cdot 20 = 12.67$ ticks. We can represent an ADΔER event (without its spatial coordinate), illustrated in angle brackets, as the node connection in Fig. 2b. These events consist of integer numbers, so we use $\lfloor \Delta t \rfloor = 12$. At this stage, we create a child node to represent the remaining integration. The child takes on the head node's D (i.e., $D = 6$). We integrate the remaining $101 - 64 = 37$ intensity units for the child node, spanning $(37/101) \cdot 20 = 7.33$ ticks. Finally, we increment the head node's D value and integrate the remaining $I = 37, \Delta t = 7.33$ intensity.

Let us now integrate 40 intensity units over 30 ticks, as in Fig. 2c. The head node saturates its $2^D = 128$ integration, so we *replace* the child node with a new node, connected by the new event $\langle D = 7, \Delta t = \lfloor 20 + (27/40) \cdot 30 \rfloor = 40 \rangle$. In this way, we can minimize the number of ADΔER events required to represent an intensity sequence, maximizing the D and Δt values of the events. As before, we integrate the remaining intensity after spawning the child to both the head and the child nodes.

Finally, let us integrate 25 intensity units over 30 ticks, as in Fig. 2d. We first integrate the head event. It does *not* reach its $2^D = 256$ integration threshold, so we do not replace its event nor replace the child node. Then we integrate the child node with the same intensity, reaching its $2^D = 32$ threshold with 19 intensity units and an additional $(19/25) \cdot 30 = 22.8$ ticks. Thus, we increment D, create an event for the child, $\langle D = 5, \Delta t = \lfloor 9.75 + (19/25) \cdot 30 \rfloor = 32 \rangle$, and spawn another node in the list to integrate the remaining 6 intensity units across $(6/25) * 30 = 7.2$ ticks.

The node connections are a queue which a transcoder outputs when there is a significant change in the intensities being integrated. When such a criterion is met, we can dequeue the events from the list, and the tail node becomes the new head, as in Fig. 2e. This linked list structure minimizes the number of ADΔER events required to represent a sequence of intensity integrations and eliminates the "speckle" artifacts present in the work of Freeman et al. [15–17].

3.2.2 User-Tunable Parameters. A user may set a number of transcode parameters according to their needs.

Δt_s: Number of ticks per second. This parameter defines the temporal resolution of the ADΔER stream.

Δt_{ref}: Number of ticks for a standard length integration (e.g., an input frame exposure time, when the source is a framed video). This parameter, along with Δt_s, determines the accuracy and data rate.

M: ADΔER contrast threshold. When a pixel's incoming normalized intensity exceeds its baseline intensity by this threshold (either positively or negatively), then the pixel's event queue will be output, its integration state reset, and its baseline set to the new intensity. Unlike the DVS model, which detects changes in log intensity, we look at the change in absolute intensity. In this paper, M is uniform for every pixel, although future applications can dynamically adjust M per-pixel with no added complexity. Below, we evaluate the effect of M on transcoded quality and event rate.

Δt_{max}: The maximum Δt that any event can span. Suppose that there is a static scene; this parameter is directly correlated with the event rate. That is, halving Δt_{max} would result in a doubling of the event rate. In practice, however, the user will simply want to ensure that Δt_{max} is sufficiently large enough so that the desired amount of intensity averaging will occur in temporally stable regions of the video, but small enough so that the given application will receive pixel updates at a fast enough rate.

3.2.3 Binary Representation. A raw ADΔER file begins with a header containing metadata that describe the resolution of the video, the file specification version, the endianness, Δt_s, Δt_{ref}, and Δt_{max}. Immediately after this header, there is a sequence of raw ADΔER events in the following format.

x: unsigned 16-bit x address.

y: unsigned 16-bit y address.

c: optional unsigned 8-bit color channel address. If the video is monochrome, then the c value is absent.

D: unsigned 8-bit decimation value.

Δt: time spanned since the last event of this pixel.

We use a packed representation, so that each event is 9-10 bytes, depending on the presence of the color channel. However, we emphasize that this style of event representation is highly compressible, as demonstrated in [15, 17]. We note that the range of natural values for D is $[0, 127]$, since 2^{127} is the maximum representable unsigned integer in our language of choice, although D values will typically span the range $[0, 30]$. We additionally reserve $D = 254$ as a special symbol that indicates that the event represents a 0-intensity, which cannot be communicated with $D \in [0, 127]$ alone.

3.2.4 Implementation Details. We implemented our entire framework in the Rust language. Non-Rust dependencies include OpenCV [8] for image ingest, processing, and visualization, and FFmpeg [14] for framed video coding. The framework is fast, highly parallel, and optimized to take advantage of CPU resources. For all speed evaluations in this paper, we used an 8-core AMD Ryzen 2700x CPU, with output files written to a RAM disk. We provide our open-source implementation at github.com/ac-freeman/adder-codec-rs/releases/tag/v0.2.0.

4 FRAMED VIDEO SOURCES

We begin by describing how we apply our ADΔER pixel model to framed video sources.

4.1 Transcoder Details

We conceptualize a video frame as a matrix of intensities integrated over a fixed time period, assuming that the intensity for each pixel may change only at instantaneous moments between frames (Fig. 1b). Δt_{ref} defines the integration time of each frame, and we set $\Delta t_s = \Delta t_{ref} F$ ticks for the source video frame rate, F. Since each ADΔER pixel integrates exactly one intensity per input frame, we can easily parallelize this process by integrating groups of pixels on many CPU threads, then collecting the resulting ADΔER events into a single vector to write out after integrating each frame. Since each pixel is processed independently in the ADΔER model, temporally interleaving the events of different pixels is unnecessary. For example, Pixel A may fire events A_1, then A_2, while Pixel B may have an event B_1 that falls between the times of A_1 and A_2. In this case, it is valid to encounter A_1 and A_2 before B_1 in the ADΔER stream, but we will never encounter A_2 before A_1.

4.1.1 Framed Reconstruction and Preserving Temporal Coherence. To easily view and evaluate the effects of our ADΔER transcode with traditional methods, we can perform a framed reconstruction of the data. For this, we simply maintain a counter of the running timestamp, T for each pixel in the ADΔER stream. When reading an event $\langle x, y, D', \Delta t' \rangle$ for a given pixel, we normalize the intensity per frame to $I_{frame} = \frac{2^{D'} \Delta t_{ref}}{\Delta t'}$, then set this as the pixel value for all frames in $[T/\Delta t_{ref}, (T + \Delta t')/\Delta t_{ref}]$.

We must, however, ensure that intensity changes only occur along frame temporal boundaries. Otherwise, there would be temporal decoherence as certain pixels fire slightly earlier than others, unless the source video is recorded at an extremely high frame rate. To this end, we ignore any intensity remaining for a given pixel list state after it generates a new ADΔER event, thus ensuring that the temporal start of each event corresponds to the beginning of a frame in the source. We can simply infer the time between events when processing the events, by rounding T up to the next multiple of Δt_{ref}. Our framed reconstruction program handles this case automatically, since both the source data type and Δt_{ref} are encoded in the header of the ADΔER file.

4.2 Optimizing Δt_{ref}

Since we define Δt_s relatively to Δt_{ref}, it is crucial to understand the effect of Δt_{ref} on the transcoded representation's quality. We transcoded a diverse set of 10 framed videos to ADΔER with $M = 0$ (for the most accurate representation) at various choices of Δt_{ref}. We then performed framed reconstruction of the ADΔER streams to use the VMAF [26] metric to calculate the perceptual quality of the reconstructions compared to the source videos. We found that at least 255 ticks per input frame is a necessary parameter for strong reconstruction quality, due to the *bit depth* of the source videos (namely, 8 bits per channel). The maximum intensity of the source frame is $\frac{255}{255} = 1$ intensity units per tick in $\Delta t_{ref} = 255$, which we can represent with $\langle D = 7, \Delta t = 128 \rangle$. The minimum intensity is 0, which we may represent with $\langle D = 254, \Delta t = 255 \rangle$. However, if $\Delta t_{ref} = 254$, we see that the maximum intensity $\frac{255}{254} = 1.004$ is not representable with an integer Δt value. Thus, we must set Δt_{ref} large enough that we may represent any source intensity without losing accuracy, such that $\frac{I_{max}}{\Delta t_{ref}} \leq 1.0$.

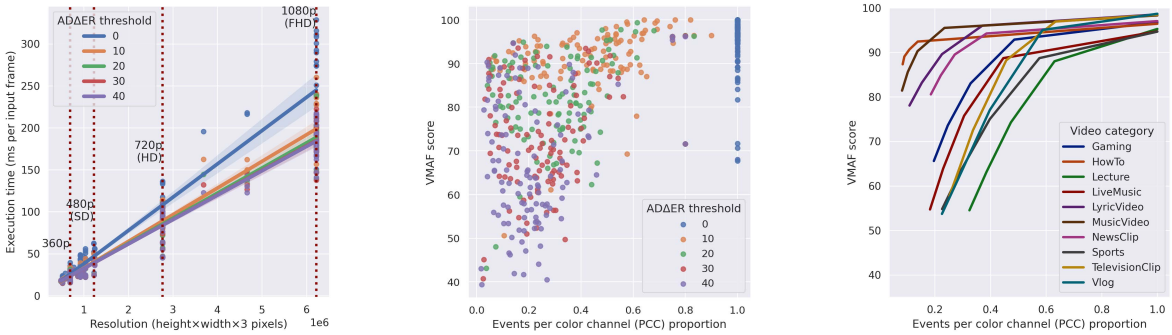
4.3 Evaluation

To evaluate our ADΔER representation on framed video sources, we employed 112 videos from a subset of the YT-UGC video compression data set [46]. These 20-second videos span 10 categories (listed in Fig. 4) and have resolutions 360p, 480p, 720p, and 1080p. We used the variants of the videos provided with H.264 compression at the CRF-10 quality level, and each video has 24-bit color.

We transcoded each video to ADΔER with parameters $\Delta t_{ref} = 255$ and $\Delta t_{max} = \Delta t_{ref} \cdot 120 = 30600$, such that the longest Δt representable by any generated ADΔER event can span 120 input frames. As described in Sec. 4.1, we set $\Delta t_s = \Delta t_{ref} F$ for each video. For example, a 24 FPS video has $\Delta t_s = 6120$. For each video, we varied the ADΔER threshold parameter $M \in \{0, 10, 20, 30, 40\}$. We recorded the execution time on our test machine (Sec. 3.2.4) for each transcode, as illustrated in Fig. 3a. We see that larger values of M yield slightly faster transcode operations, and that the time required to process an input frame scales linearly with video resolution.

Finally, we reconstructed framed videos from our ADΔER representations as described in Sec. 4.1.1, so that we can examine the quality properties of our representation. We gathered the VMAF score of each ADΔER video in the M range, compared to its reference video. Fig. 4 shows that the median VMAF score for each video category is extremely high, in the 95-99 range, and that perceptual quality decreases as M increases.

If we examine the number of ADΔER events per pixel color channel at various levels of M as a proportion compared to $M = 0$ (Figs. 3b and 3c), we see a clear rate-distortion curve dependent on the choice of M. By merely setting $M = 10$, we see a median reduction in ADΔER rate by 54%, and a median reduction of VMAF score by only 4.5 across our data set. In our tests, we observe a



(a) (b) (c)

Figure 3: Quantitative results of ADΔER on framed sources. (a) Effect of resolution on execution time, at various M. Performance scales linearly with resolution. (b) Scatter plot of the rate-distortion for all videos at various M. (c) Particular rate-distortion curves for one video in each category of the data set. Increasing M yields lower-quality, but lower-rate ADΔER videos.

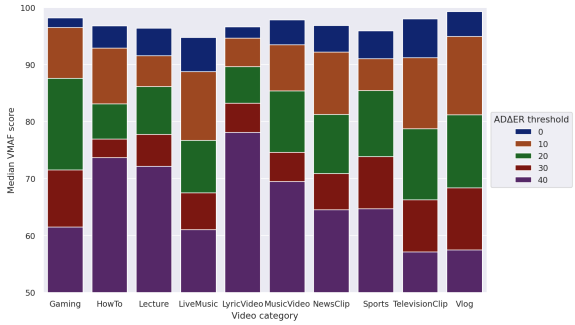


Figure 4: Median VMAF score for each framed video category after transcoding to ADΔER at various M. Quality decreases as M increases for every category. Less dynamic video categories such as `LyricVideo` have little quality degradation as M increases, while more dynamic categories such as `TelevisionClip` and `Vlog` degrade greatly.

maximum ADΔER precision of 44 dB or 14 bits, demonstrating a significant increase over the 8-bit precision of the source videos. The precision tends to increase as pixels can integrate (and thus average) longer sequences of intensities with higher M, thus widening the range of represented values.

As a qualitative example, Fig. 7a shows instantaneous frame samples from the ADΔER transcodes with our range of M values[1]. With higher M, we observe a greater "smearing" effect in low-contrast regions, less overall detail, and multicolored ghosting artifacts from scene transitions or fast camera motions. Surprisingly, the `LiveMusic` and `Sports` examples shown are both outliers for the VMAF score with $M = 0$ in Fig. 3b, despite appearing high quality to a casual observer. This is due to the imprecision of some events with $\Delta t > \Delta t_{ref}$, near high-contrast transition points, since we round Δt to an integer to form an ADΔER event. Furthermore, since we represent each color channel with an independent event pixel list, a small error in one color channel of a pixel will have a compound effect on the VMAF perceptual quality of that pixel. These two videos exhibit jittery camera motion, suggesting that rapid intensity changes may make such quantization errors more frequent.

[1]We provide the framed reconstructions of these videos as supplementary material at bit.ly/41onrGs.

5 EVENT VIDEO SOURCES

While in Sec. 4 we demonstrated that ADΔER can effectively represent framed video sources asynchronously, we here discuss ADΔER's utility in representing video sources which are already asynchronous. Specifically, we explore the representation of the DAVIS 346 camera's DVS event and APS frame data in the `.aedat4` file format.

5.1 Event-Based Double Integral

As we discussed in Sec. 2.2, many applications that utilize mixed-modality sources (frames and events) either reconstruct an image sequence at a fixed frame rate or process the frame data separately from the events. Furthermore, while APS frames provide absolute intensity measurements across the whole sensor, they can often be blurry in difficult exposure scenarios: low scene illumination, fast motion, or narrow aperture. Meanwhile, the higher sensitivity of DVS pixels and their fast responsiveness allow them to capture events with subframe precision. Thus, DAVIS-based reconstruction methods must address the crucial problem of *deblurring* the APS frame data. Most of these efforts, however, utilize slow machine learning techniques which severely limit the practical reconstructed frame rate, obviating a primary benefit to using event cameras [43, 44]. One non-learned method for framed reconstruction, however, is the Event-based Double Integral (EDI) [31]. The authors introduced the EDI model to find a sharp "latent" image, L, for a blurry APS image, by integrating the DVS events occurring during the APS image's exposure time. Crucially, this paper involves an optimization technique for determining the θ value of the DVS sensor at a given point in time; that is, the method deblurs an APS image with various choices of DVS sensitivity, θ, and the sharpest latent image produced corresponds to the optimal choice of θ. Both the APS deblurring and θ optimization techniques are vital steps in our ADΔER transcoder pipeline for event video sources.

5.1.1 Our Implementation. Unfortunately, as with much of the literature on event-based systems, the EDI model was not implemented with practical, real-time performance in mind. Rather, the authors released their implementation as an obfuscated MATLAB program, which can only reconstruct a few frames of video per second. Furthermore, their implementation uses a custom MATLAB format for the DAVIS data, underscoring our argument about the prevalence of domain-specific event data representations in Sec. 2.2.3. Therefore, we implemented the EDI model from the ground up in Rust, designing our program to decode data in the packet format produced directly by the DAVIS 346 camera. Our implementation can reconstruct a frame sequence for this resolution at 1000+ FPS, two orders of magnitude faster than the baseline MATLAB implementation. To allow for realistic and repeatable performance evaluation, we simulate the packet latency of pre-recorded videos to reflect their real-world timing. The user can also generate one deblurred image for each APS frame, rather than a high-rate frame sequence. Our implementation supports a connected DAVIS 346 camera as input, unlocking the potential for practical, real-time DAVIS applications.

5.2 Transcoder Details

We identify three ways in which we can transcode DAVIS data to ADΔER. Each mode utilizes the same per-pixel integration scheme as described in Sec. 3.2, but the actual intensities that we *integrate* depend on the mode we choose. In each case, we receive input data from our EDI implementation running alongside our transcoder. Unlike our framed source transcoder, where we determine Δt_s from our choice of Δt_{ref} and the video frame rate, here we set $\Delta t_s = 1 \times 10^6$ to match the temporal resolution of the DAVIS 346. Then, Δt_{ref} defines the desired length of a deblurred frame in modes (i) and (ii) below, and the intensity scale in mode (iii) below. We illustrate our pipeline in Fig. 5.

5.2.1 Mode (i): Deblurred frame sequence → ADΔER. If we reconstruct a high-rate framed sequence with EDI, we can trivially use our framed-source transcoder technique as described in Sec. 4.1. Qualitatively, this method produces the best-looking results, since the EDI model inherently unifies positive- and negative-polarity events in log space. That is, we can accumulate a pixel's multiple events occurring over a given frame interval to arrive at a final intensity value for the given frame. Any intensities outside the APS frame's intensity range $[0, 255]$ are clamped only at this point.

5.2.2 Mode (ii): Deblurred APS frame + DVS events → ADΔER. We may also choose to use EDI to simply deblur each APS frame, and input the DVS events occurring outside that frame's exposure time individually in our ADΔER transcoder. Since DVS events express log intensity relative to a previous latent value, we maintain a matrix of the most recent log intensity for each pixel, by which we calculate the intensity represented by the incoming DVS event. When we ingest our deblurred frame, we scale the frame intensities I to the range $L \in [0, 1]$, and set the latent log intensity for each pixel by $\widetilde{L} = \ln(1 + L)$. We interpret a DVS event as specifying the exact moment an intensity *changes*. Suppose that, for a given pixel, we ingest a deblurred frame intensity of $I_0 = 20$ that spans time $[t' - \Delta t_{ref}, t']$, where the frame has 8-bit values and $\Delta t_{ref} = 1000$. Then, we have latent value $\widetilde{L_0} = 0.0755$. Suppose we have a sequence of DVS events $\langle p, t \rangle$ occurring after time t', $\{E_1 = \langle 0, t' + 500 \rangle, E_2 = \langle 1, t' + 800 \rangle, E_3 = \langle 1, t' + 1200 \rangle\}$. When we encounter E_1, we first *repeat* the previous integration to fill the time elapsed. That is, we integrate $\frac{L_0 \cdot 500}{\Delta t_{ref}} = 10$ intensity units over 500 ticks.

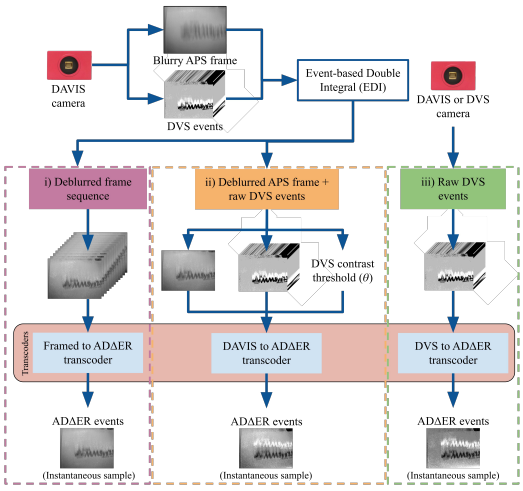
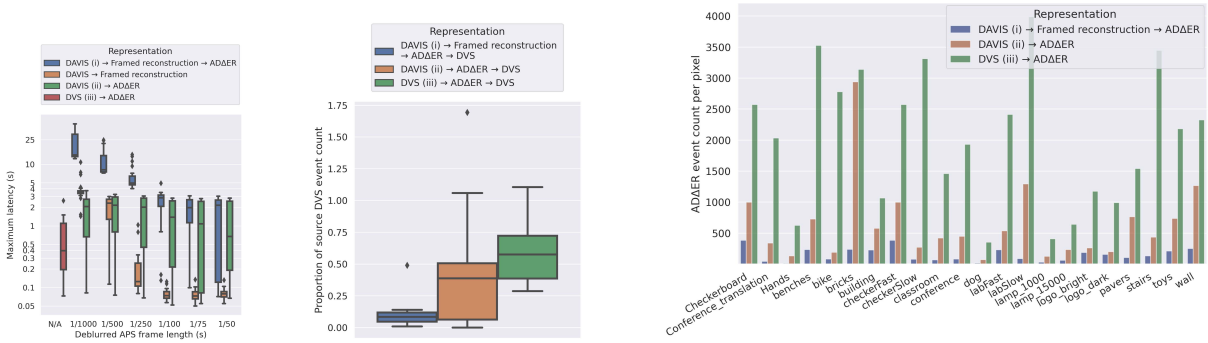


Figure 5: Pipeline for transcoding event camera data to ADΔER. Our transcoder supports three modes. Modes (i) and (ii) incorporate deblurred APS frames from a DAVIS sensor, while mode (iii) uses DVS events alone.

Then, we simply set the new latent intensity as follows. Supposing $\theta = 0.15$, our latent value L_1 becomes $L_1 = \exp(\widetilde{L_0} - 0.15) - 1.0 = -0.0718$. Since we cannot integrate negative intensities, however, we must clamp L_1 and $\widetilde{L_1}$ to 0.0. At this stage, we do not know how long the pixel maintains this intensity, so we do not immediately integrate an additional value. Rather, when we encounter E_2, we integrate $\frac{L_1 \cdot 255 \cdot 300}{\Delta t_{ref}} = 0$ intensity units over 300 ticks. As before, we set the new latent log value $\widetilde{L_2} = \widetilde{L_1} + 0.15 = 0.15$, such that $L_2 = \exp(\widetilde{L_2}) - 1.0 = 0.1618$. When we encounter E_3, then, we integrate $\frac{L_2 \cdot 255 \cdot 400}{\Delta t_{ref}} = 41.27$ intensity units over 400 ticks.

Although the clamping mechanism causes a worse visual appearance than mode (i) (Sec. 5.2.1), as shown in the white and black pixels of moving edges in Fig. 7b, this method preserves the temporal resolution of the source events. By contrast, intensity timings in mode (i) are quantized to the EDI reconstruction frame rate.

5.2.3 Mode (iii): DVS events → ADΔER. Finally, we can choose to ignore the APS data, or use a DVS sensor alone, and integrate the DVS events directly. In this case, our EDI program functions solely as the camera driver or event file decoder. At sub-second intervals, we reset the latent log image $\widetilde{L}$ in the ADΔER transcoder to $\ln(0.5)$, representing a mid-level intensity. Then, we calculate the absolute intensity of the following DVS events relative to this mid-level intensity. As we do not deblur APS frames in this mode, we cannot find the optimal θ value by which to interpret these DVS events. Therefore, we use a constant $\theta = 0.15$ to calculate the changes in log intensity. As shown in Fig. 5, we only gather intensity information for pixels that change greatly enough to trigger DVS events, and unchanging pixels will carry a mid-level gray value.



(a) (b) (c)

Figure 6: Quantitative results of ADΔER on event camera sources. (a) Effect of Δt_{ref} on the latency of ADΔER transcoding. (b) Proportion of recovered DVS events after ADΔER transcoding, as compared to the source `.aedat4` DVS event rates. (c) ADΔER event rates for each transcode mode and each video of our data set, where $\Delta t_{ref} = \Delta t_s/500$ and $M = 40$. Mode (iii) yields dramatically higher ADΔER rates, despite not encompassing absolute intensities in slowly-changing or static regions of the scene.

5.3 Evaluation

As our chief aim was to enable the development of fast, practical event-based systems, we needed to test our method on a real-world source representation. However, as discussed in Sec. 2.2.3, most event vision data sets use an intermediate representation that is slower to ingest. Thus, we employ the DVSNOISE20 data set [1], which provides raw `.aedat4` files in the same format that the DAVIS 346 camera produces. This data set provides 3 recordings for each of 16 scenes, though we used only one recording per scene in our evaluation. We also add to the data set 6 of our own recordings with varied APS exposure lengths and more extreme lighting conditions.

We transcoded each DAVIS video to ADΔER using mode (ii) with ADΔER threshold parameter $M \in \{0, 10, 20, 30, 40\}$, $\Delta t_{ref} \in \{100, 1000\}$, $\Delta t_s = 1 \times 10^6$ (microsecond resolution), and $\Delta t_{max} = \Delta t_s \cdot 4 = 4 \times 10^6$. At $M = 10$ for both choices of Δt_{ref}, we observed a median 49-50% decrease in the ADΔER event rate compared to $M = 0$. At $M = 40$, we observed a median 66% decrease in ADΔER event rate compared to $M = 0$, demonstrating the utility of our temporal average scheme.

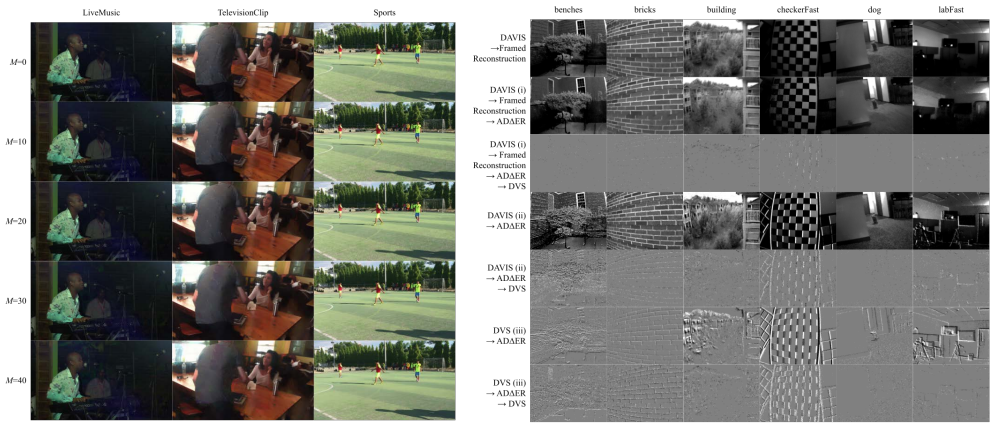
Secondly, we transcoded each DAVIS video to ADΔER under modes (i) and (ii) with a fixed $M = 40$, and $\Delta t_{ref} \in \{\Delta t_s/1000, \Delta t_s/500, \Delta t_s/250, \Delta t_s/100, \Delta t_s/75, \Delta t_s/50\}$ ticks. Additionally, we ran direct framed reconstructions of DAVIS in EDI and we transcoded each video once with mode (iii). We recorded the maximum latency between reading a packet from the `.aedat4` source video in our EDI program and completing the transcode of that packet's data. In Fig. 6a, we observe that both framed reconstruction and transcode mode (i) incur dramatically more latency as the effective frame rate increases, which corroborates our findings in Sec. 4.3. Furthermore, we observe worse latency for sequences with high DVS event rates. By contrast, mode (ii) incurs relatively constant 2.1 second latency across all values of Δt_{ref}, yet has less variance as Δt_{ref} increases. The speed of mode (iii) does not depend on Δt_{ref}, and we see that we can achieve a median transcode latency of only 0.4 seconds.

These results support our argument that it is computationally expensive to employ a high-rate, framed intensity representation for event data. While such methods unavoidably quantize the temporal components of the event sources, modes (ii) and (iii) of our scheme can transcode the DAVIS/DVS event data to an intensity representation, ADΔER, with a temporal resolution matching that of the source.

We see direct evidence for this claim in Fig. 6b, which shows the proportion of DVS events reconstructed from the ADΔER representations after transcoding with $M = 40$ and $\Delta t_{ref} = \Delta t_s/500$ ticks, compared to the number of DVS events in the *source* DAVIS videos. We can quickly reconstruct a DVS stream from ADΔER, since the intensity changes to trigger DVS events will occur only at the temporal boundaries of our sparse ADΔER events. However, we must assume a constant θ, as in transcode mode (iii) (Sec. 5.2.3). Mode (i), which quantizes events into fixed-duration frames, can recover only a small fraction of the DVS events in the source. On the contrary, modes (ii) and (iii) have high proportions of recoverable DVS events, even with a high M. Fig. 6c illustrates that transcode mode (ii) generally necessitates far fewer ADΔER events than mode (iii) with these same settings. Thus, our ADΔER codec with mode (ii) can simultaneously utilize the APS frames to compress the DVS stream and utilize the DVS stream to deblur the APS frames, while fusing both streams into a unified, asynchronous intensity representation.

Fig. 7b shows qualitative results on six scenes from our data set[2]. The ADΔER transcodes shown use $M = 40$ and $\Delta t_{ref} = \Delta t_s/500$, as in our latency experiments. Since mode (i) of our event transcoder simply uses our framed source transcoder (Sec. 4.1) after reconstructing a frame sequence with EDI, we note that the artifacts in the ADΔER representation here stem from artifacts in EDI's *framed reconstruction*, or from the temporal averaging in ADΔER. In particular, we point to the `building` images, which show the limitations of EDI in deblurring APS frames under fast motion and long exposures. We see that we recover few DVS events

[2]We provide the framed reconstructions of these videos as supplementary material at bit.ly/41onrGs.



(a) (b)

Figure 7: Qualitative results of the ADΔER transcoder. (a) Instantaneous samples of framed videos transcoded to ADΔER at various M. High M yields more smearing and ghosting artifacts in low-contrast regions. (b) Event videos transcoded to framed video (first row) and to ADΔER under our three transcode modes (rows 2, 4, and 6) with $\Delta t_{ref} = \Delta t_s/500$ and $M = 40$. We transcode our ADΔER representations back to DVS (rows 3, 5, and 7) and see that modes (ii) and (iii) preserve much more of the temporal contrast detail of the DVS streams. Mode (ii) has a worse visual appearance than mode (i), due to DVS intensity drift.

from mode (i), due to its temporal quantization of the source events. However, mode (ii) exhibits a similar quality of recovered DVS events compared to mode (iii), despite also conveying the absolute intensities for pixels without DVS source events.

6 FUTURE WORK

6.1 Extending this Paper

We note the difficulty in empirically evaluating the quality of our transcoded representation for DAVIS/DVS sources. Since APS frames are blurred, they are not reliable references for framed quality metrics (e.g., PSNR, SSIM, and VMAF). Furthermore, existing metrics for DVS stream quality focus on noise filtering tasks and can only process a few hundred events per second [25], while real-world DVS sequences encompass several millions of events per second. Therefore, when an end-to-end system is in place, we will evaluate application performance as a metric for the representational quality of streams transcoded from event-based sources. Finally, we will also work to expand our framework with support for additional event camera data types, such as those from Prophesee [32].

6.2 Bigger Picture

Our video acquisition and representation pipeline is a necessary first step toward the development of a real-time, source-agnostic event video ecosystem. The next steps lie in the compression, streaming, and applications of ADΔER data.

As we have previously noted, ADΔER is amenable to lossy compression, yet incompatible with traditional video codecs. Therefore, it requires novel techniques for inducing loss according to metrics of data distortion, connection bandwidth, and application performance. ADΔER's sparse representation naturally produces low data rates where pixels are the least dynamic, but future work will focus on inducing loss on the Δt components without causing temporal decoherence between independent pixel streams, incorporate block-based coding, support stream scrubbing and drop-in, and rate adaptation. Lossy compression schemes for ADΔER are likely to produce temporal artifacts akin to increasing M during transcode, rather than the spatial "blocking" artifacts of traditional video. Since a streaming client will only receive data updates for dynamic video regions, we naturally avoid sending redundant data, and we can allocate more bits to pixels of interest in a bandwidth-constrained scenario.

Finally, connected client applications can take a two-pronged approach to ADΔER data: they may be bespoke event-based applications ingesting ADΔER streams directly, or they may be classical frame-based applications seeking to benefit from our sparse representation and high temporal precision. In the latter case, the client can utilize our open-source library to reconstruct framed images from received ADΔER data, though the maximum incurred reconstruction latency will scale linearly with the choice of Δt_{max} in the case of real-time streaming. Computational overhead will also limit the spatiotemporal resolution that may be processed in real time, though robust compression schemes can help alleviate this barrier.

7 CONCLUSION

This paper proposes the ADΔER video representation for both framed and event sources. We show that, through temporal averaging of similar intensities, we can increase the intensity precision and greatly reduce the number of samples per pixel compared to grid-based representations. We demonstrate extremely low-latency DAVIS fusion, and we argue that our method provides the computational efficiency and temporal granularity necessary for building real-time intensity-based applications for event cameras.

ACKNOWLEDGMENTS

This work is partially supported by a grant from the United States Department of Defense.

REFERENCES

[1] R. Baldwin, M. Almatrafi, V. Asari, and K. Hirakawa. 2020. Event Probability Mask (EPM) and Event Denoising Convolutional Neural Network (EDnCNN) for Neuromorphic Cameras. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, Los Alamitos, CA, USA, 1698–1707. https://doi.org/10.1109/CVPR42600.2020.00177

[2] Raymond Baldwin, Ruixu Liu, Mohammed Mutlaq Almatrafi, Vijayan K Asari, and Keigo Hirakawa. 2022. Time-Ordered Recent Event (TORE) Volumes for Event Cameras. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2022), 1–1. https://doi.org/10.1109/TPAMI.2022.3172212

[3] Srutarshi Banerjee, Henry H. Chopp, Jianping Zhang, Zihao W. Wang, Peng Kang, Oliver Cossairt, and Aggelos Katsaggelos. 2022. A Joint Intensity-Neuromorphic Event Imaging System With Bandwidth-Limited Communication Channel. *IEEE Transactions on Neural Networks and Learning Systems* (2022), 1–15. https://doi.org/10.1109/TNNLS.2022.3214779

[4] Srutarshi Banerjee, Zihao W. Wang, Henry H. Chopp, Oliver Cossairt, and Aggelos K. Katsaggelos. 2021. Lossy Event Compression Based On Image-Derived Quad Trees And Poisson Disk Sampling. In *2021 IEEE International Conference on Image Processing (ICIP)*. IEEE. https://doi.org/10.1109/icip42928.2021.9506546

[5] Thomas Barbier, Celine Teuliere, and Jochen Triesch. 2021. Spike Timing-Based Unsupervised Learning of Orientation, Disparity, and Motion Representations in a Spiking Neural Network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. 1377–1386.

[6] Ryad Benosman, Charles Clercq, Xavier Lagorce, Sio-Hoi Ieng, and Chiara Bartolozzi. 2014. Event-Based Visual Flow. *IEEE Transactions on Neural Networks and Learning Systems* 25, 2 (2014), 407–417. https://doi.org/10.1109/TNNLS.2013.2273537

[7] Raphael Berner, Christian Brändli, and Massimo Zannoni. 2018. Data Rate Control for Event-Based Vision Sensor.

[8] G. Bradski. 2000. The OpenCV Library. *Dr. Dobb's Journal of Software Tools* (2000).

[9] Christian Brandli, Lorenz Muller, and Tobi Delbruck. 2014. Real-time, high-speed video decompression using a frame- and event-based DAVIS sensor. In *2014 IEEE International Symposium on Circuits and Systems (ISCAS)*. 686–689. https://doi.org/10.1109/ISCAS.2014.6865228

[10] 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. https://doi.org/10.1109/TCSVT.2021.3101953

[11] T. Delbruck, R. Graca, and M. Paluch. 2021. Feedback control of event cameras. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. IEEE Computer Society, Los Alamitos, CA, USA, 1324–1332. https://doi.org/10.1109/CVPRW53098.2021.00146

[12] Georgi Dikov, Mohsen Firouzi, Florian Röhrbein, Jörg Conradt, and Christoph Richter. 2017. Spiking Cooperative Stereo-Matching at 2 ms Latency with Neuromorphic Hardware. In *Biomimetic and Biohybrid Systems*, Michael Mangan, Mark Cutkosky, Anna Mura, Paul F.M.J. Verschure, Tony Prescott, and Nathan Lepora (Eds.). Springer International Publishing, Cham, 119–137.

[13] Hadar Cohen Duwek, Albert Shalumov, and Elishai Ezra Tsur. 2021. Image Reconstruction From Neuromorphic Event Cameras Using Laplacian-Prediction and Poisson Integration With Spiking and Artificial Neural Networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. 1333–1341.

[14] FFmpeg Project. 2021. *FFmpeg*. https://ffmpeg.org/

[15] Andrew C. Freeman, Chris Burgess, and Ketan Mayer-Patel. 2021. Motion Segmentation and Tracking for Integrating Event Cameras. In *Proceedings of the 12th ACM Multimedia Systems Conference* (Istanbul, Turkey) *(MMSys '21)*. Association for Computing Machinery, New York, NY, USA, 1–11. https://doi.org/10.1145/3458305.3463373

[16] Andrew C. Freeman and Ketan Mayer-Patel. 2020. Integrating Event Camera Sensor Emulator. In *Proceedings of the 28th ACM International Conference on Multimedia* (Seattle, WA, USA) *(MM '20)*. Association for Computing Machinery, New York, NY, USA, 4503–4505. https://doi.org/10.1145/3394171.3414394

[17] Andrew C. Freeman and Ketan Mayer-Patel. 2021. Lossy Compression for Integrating Event Cameras. In *2021 Data Compression Conference (DCC)*. 53–62. https://doi.org/10.1109/DCC50243.2021.00013

[18] G. Gallego, T. Delbruck, G. M. Orchard, C. Bartolozzi, B. Taba, A. Censi, S. Leutenegger, A. Davison, J. Conradt, K. Daniilidis, and D. Scaramuzza. 2020. Event-based Vision: A Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2020), 1–1. https://doi.org/10.1109/TPAMI.2020.3008413

[19] Daniel Gehrig, Antonio Loquercio, Konstantinos G. Derpanis, and Davide Scaramuzza. 2019. End-to-End Learning of Representations for Asynchronous Event-Based Data. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*.

[20] Arren Glover, Valentina Vasco, and Chiara Bartolozzi. 2018. A Controlled-Delay Event Camera Framework for On-Line Robotics. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*. 2178–2183. https://doi.org/10.1109/ICRA.2018.8460541

[21] Yuhuang Hu, Jonathan Binas, Daniel Neil, Shih-Chii Liu, and Tobi Delbruck. 2020. DDD20 End-to-End Event Camera Driving Dataset: Fusing Frames and Events with Deep Learning for Improved Steering Prediction. In *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*. 1–6. https://doi.org/10.1109/ITSC45102.2020.9294515

[22] Yuhuang Hu, Shih-Chii Liu, and Tobi Delbruck. 2021. v2e: From Video Frames to Realistic DVS Events. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 1312–1321. https://doi.org/10.1109/CVPRW53098.2021.00144

[23] Alex Junho Lee, Younggun Cho, Young-sik Shin, Ayoung Kim, and Hyun Myung. 2022. ViViD++: Vision for Visibility Dataset. *IEEE Robotics and Automation Letters* 7, 3 (2022), 6282–6289.

[24] Lucien Lenzen, Rolf Hedtke, and Mike Christmann. 2018. How Tone Mapping Influences the Bit Rate and the Bit Depth of Coded Sequences. *SMPTE Motion Imaging Journal* 127, 5 (2018), 38–43. https://doi.org/10.5594/JMI.2018.2810021

[25] Jianing Li, Yihua Fu, Siwei Dong, Zhaofei Yu, Tiejun Huang, and Yonghong Tian. 2021. Asynchronous Spatiotemporal Spike Metric for Event Cameras. *IEEE Transactions on Neural Networks and Learning Systems* (2021), 1–12. https://doi.org/10.1109/TNNLS.2021.3061122

[26] Zhi Li, Anne Aaron, Ioannis Katsavounidis, Anush Moorthy, and Megha Manohara. 2016. Toward a practical perceptual video quality metric. *The Netflix Tech Blog* 6, 2 (2016).

[27] Ana Maqueda, Antonio Loquercio, Guillermo Gallego, Narciso Garcia, and Davide Scaramuzza. 2018. Event-Based Vision Meets Deep Learning on Steering Prediction for Self-Driving Cars. 5419–5427. https://doi.org/10.1109/CVPR.2018.00568

[28] Elias Mueggler, Henri Rebecq, Guillermo Gallego, Tobi Delbruck, and Davide Scaramuzza. 2017. The event-camera dataset and simulator: Event-based data for pose estimation, visual odometry, and SLAM. *The International Journal of Robotics Research* 36, 2 (2017), 142–149. https://doi.org/10.1177/0278364917691115 arXiv:https://doi.org/10.1177/0278364917691115

[29] Yuya Omori, Takayuki Onishi, Hiroe Iwasaki, and Atsushi Shimizu. 2017. A 120 fps high frame rate real-time HEVC video encoder with parallel configuration scalable to 4K. In *2017 IEEE Symposium in Low-Power and High-Speed Chips (COOL CHIPS)*. 1–3. https://doi.org/10.1109/CoolChips.2017.7946382

[30] Liyuan Pan, Richard Hartley, Cedric Scheerlinck, Miaomiao Liu, Xin Yu, and Yuchao Dai. 2020. High Frame Rate Video Reconstruction based on an Event Camera. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2020), 1–1. https://doi.org/10.1109/TPAMI.2020.3036667

[31] Liyuan Pan, Cedric Scheerlinck, Xin Yu, Richard Hartley, Miaomiao Liu, and Yuchao Dai. 2019. Bringing a Blurry Frame Alive at High Frame-Rate With an Event Camera. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

[32] Etienne Perot, Pierre de Tournemire, Davide Nitti, Jonathan Masci, and Amos Sironi. 2020. Learning to Detect Objects with a 1 Megapixel Event Camera. In *Proceedings of the 34th International Conference on Neural Information Processing Systems* (Vancouver, BC, Canada) *(NIPS'20)*. Curran Associates Inc., Red Hook, NY, USA, Article 1396, 14 pages.

[33] Christoph Posch, Daniel Matolin, and Rainer Wohlgenannt. 2010. A QVGA 143dB dynamic range asynchronous address-event PWM dynamic image sensor with lossless pixel-level video compression. In *2010 IEEE International Solid-State Circuits Conference - (ISSCC)*. 400–401. https://doi.org/10.1109/ISSCC.2010.5433973

[34] Henri Rebecq, Timo Horstschaefer, and Davide Scaramuzza. 2017. Real-time Visual-Inertial Odometry for Event Cameras using Keyframe-based Nonlinear Optimization. https://doi.org/10.5244/C.31.16

[35] Henri Rebecq, René Ranftl, Vladlen Koltun, and Davide Scaramuzza. 2019. Events-to-Video: Bringing Modern Computer Vision to Event Cameras. *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)* (2019).

[36] Henri Rebecq, René Ranftl, Vladlen Koltun, and Davide Scaramuzza. 2019. High Speed and High Dynamic Range Video with an Event Camera. *IEEE Trans. Pattern Anal. Mach. Intell. (T-PAMI)* (2019). http://rpg.ifi.uzh.ch/docs/TPAMI19_Rebecq.pdf

[37] Christian Reinbacher, Gottfried Munda, and Thomas Pock. 2016. Real-Time Intensity-Image Reconstruction for Event Cameras Using Manifold Regularisation. 9.1–9.12. https://doi.org/10.5244/C.30.9

[38] Sourav Dey Roy and Mrinal Kanti Bhowmik. 2020. A Comprehensive Survey on Computer Vision Based Approaches for Moving Object Detection. In *2020 IEEE Region 10 Symposium (TENSYMP)*. 1531–1534. https://doi.org/10.1109/TENSYMP50017.2020.9230869

[39] Cedric Scheerlinck, Nick Barnes, and Robert Mahony. 2019. Continuous-Time Intensity Estimation Using Event Cameras. In *Computer Vision – ACCV 2018*, C.V. Jawahar, Hongdong Li, Greg Mori, and Konrad Schindler (Eds.). Springer International Publishing, Cham, 308–324.

[40] Cedric Scheerlinck, Henri Rebecq, Daniel Gehrig, Nick Barnes, Robert Mahony, and Davide Scaramuzza. 2020. Fast Image Reconstruction with an Event Camera. In *IEEE Winter Conf. Appl. Comput. Vis. (WACV)*. 156–163. https://doi.org/10.1109/WACV45572.2020.9093366

[41] M. Singh, P. Zhang, A. Vitkus, K. Mayer-Patel, and L. Vicci. 2017. A Frameless Imaging Sensor with Asynchronous Pixels: An Architectural Evaluation. In *2017 23rd IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC)*. 110–117. https://doi.org/10.1109/ASYNC.2017.19

[42] Aaron J. Smith, Montek Singh, and Ketan Mayer-Patel. 2017. A System Model For Frameless Asynchronous High Dynamic Range Sensors. In *NOSSDAV'17*.

[43] Chen Song, Qixing Huang, and Chandrajit Bajaj. 2022. E-CIR: Event-Enhanced Continuous Intensity Recovery. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 7793–7802. https://doi.org/10.1109/CVPR52688.2022.00765

[44] Bishan Wang, Jingwei He, Lei Yu, Gui-Song Xia, and Wen Yang. 2020. Event Enhanced High-Quality Image Recovery. In *European Conference on Computer Vision*. Springer.

[45] Hao Wang, Youhui Huo, and Hongbo Zhang. 2021. Research on Dynamic Range Analysis and Improvement of Imaging Equipment. In *2021 Workshop on Algorithm and Big Data* (Fuzhou, China) *(WABD 2021)*. Association for Computing Machinery, New York, NY, USA, 40–44. https://doi.org/10.1145/3456389.3456392

[46] Yilin Wang, Sasi Inguva, and Balu Adsumilli. 2019. YouTube UGC Dataset for Video Compression Research. In *2019 IEEE 21st International Workshop on Multimedia Signal Processing (MMSP)*.

[47] Ziwei Wang, Yonhon Ng, Cedric Scheerlinck, and Robert Mahony. 2021. An Asynchronous Kalman Filter for Hybrid Event Cameras. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 448–457.

[48] Alex Zhu, Liangzhe Yuan, Kenneth Chaney, and Kostas Daniilidis. 2018. EV-FlowNet: Self-Supervised Optical Flow Estimation for Event-based Cameras. https://doi.org/10.15607/RSS.2018.XIV.062

[49] Bin Zhu, Shan Liu, Yuan Liu, Yi Luo, Jing Ye, Haiyan Xu, Ying Huang, Hualong Jiao, Xiaozhong Xu, Xianguo Zhang, and Chenchen Gu. 2021. A Real-Time H.266/VVC Software Decoder. In *2021 IEEE International Conference on Multimedia and Expo (ICME)*. 1–6. https://doi.org/10.1109/ICME51207.2021.9428470

Received 18 November 2022; revised 27 January 2023; accepted 3 March 2023