Efficient Data Processing Pipeline for Event-Based Vision Datasets: Techniques and Insights

AITSAM, Muhammad <http://orcid.org/0000-0002-4157-7282>, JIMENEZ RODRIGUEZ, Alejandro and DI NUOVO, Alessandro <http://orcid.org/0000-0003-2677-2650>

ACCEPTED MANUSCRIPT • OPEN ACCESS

Efficient Data Processing Pipeline for Event-Based Vision Datasets: Techniques and Insights

To cite this article before publication: Muhammad Aitsam *et al* 2024 *Eng. Res. Express* in press https://doi.org/10.1088/2631-8695/ad9235

Efficient Data Processing Pipeline for Event-Based Vision Datasets: Techniques and Insights

Muhammad Aitsam, Alejandro Jimenez Rodriguez, Alessandro Di Nuovo

Sheffield Hallam University, Howard St, Sheffield City Centre, Sheffield S1 1WB, UK

E-mail: m.aitsam@shu.ac.uk

Abstract. Event-based vision datasets have emerged as a critical asset in advancing the capabilities of real-time perception systems, particularly in fields such as robotics, autonomous vehicles, and human-computer interaction. These datasets enable low-latency processing by capturing asynchronous, pixel-level changes in the scene, providing a distinct advantage over traditional frame-based systems. However, the diverse formats and characteristics of event-based datasets pose significant challenges for efficient processing and analysis, hindering their broader adoption and integration. In this paper, we present a versatile and comprehensive data processing pipeline designed to address these challenges by supporting multiple event-data formats, including newer formats such as EVT2 and EVT3. This pipeline not only converts data into widely supported formats like AEDAT and NPZ, but also ensures that the unique characteristics of event-based data—such as temporal precision and sparse event representation—are preserved throughout the conversion process. By applying this pipeline to several open-source datasets, we establish a standardized, efficient methodology for dataset manipulation that enhances compatibility and reproducibility in event-based vision research. Additionally, we introduce a novel high-resolution event-based action dataset, converted into various formats using our pipeline, which opens new avenues for exploring event-based techniques in action recognition. This dataset and our pipeline serve as valuable resources for the research community, enabling advancements in real-time vision applications and fostering greater collaboration and standardization across studies.

1. Introduction

Event-based vision, also referred to as neuromorphic vision, has emerged as a prominent area of interest within computer vision research due to its capacity to facilitate real-time, low-latency perception systems. Unlike conventional frame-based cameras, which capture images in a sequential manner, event-based vision records changes at the individual pixel level asynchronously and with high temporal precision. This approach offers distinct advantages and has found applications in various fields such as robotics, autonomous vehicles, and augmented reality [1]. However, the proliferation of event-based vision datasets presents

challenges related to their diverse formats and characteristics. Researchers often encounter obstacles when processing and analyzing these datasets efficiently. These challenges impede the comparability and reproducibility of research findings in different studies. To address these issues, several libraries and tools have been developed to process event data, including AERmanager [2], aedat [3], tonic [4], and spikingJelly [5]. Tonic and SpikingJelly are widely used tools for processing spiking neural network data and conducting simulations within the neuromorphic vision community. However, both tools have notable limitations when applied to event-based vision datasets. Tonic, for instance, primarily supports AEDAT format and lacks the capability to handle newer event vision sensor data formats, such as EVT2 and EVT3, which are crucial for emerging sensors like Prophesee's Metavision and CenturArks. This restricts its applicability when working with advanced sensors that use proprietary formats. SpikingJelly, while effective for spiking neural network simulations, is not optimized for real-time event-based vision tasks. Its higher computational overhead results in longer processing times, particularly for large-scale datasets. These limitations underscore the need for a more versatile and efficient data processing pipeline.

In response to this gap, our research introduces a comprehensive pipeline designed to support a wide range of event data formats, including those from emerging sensor technologies. By offering seamless compatibility with newer formats like EVT2 and EVT3, our pipeline facilitates broader accessibility and usability in the event-based vision research community, optimizing both speed and scalability. In this article, we first introduce a robust data processing pipeline meticulously engineered to tackle the complexities associated with diverse event data formats. This pipeline offers a standardized and efficient approach to converting event-based vision datasets into widely supported formats, ensuring compatibility with various event cameras and facilitating seamless integration with event-based vision systems. Second, as part of our efforts to foster collaboration and innovation within the research community, we release seven open-source datasets spanning five different formats. Lastly, we have made our novel, event-based, high-resolution gesture dataset available as an open-source resource, thereby enriching the tools available to the research community. This dataset presents a unique opportunity for investigating gesture recognition tasks through event-based vision techniques [6], enabling researchers to explore new avenues in human-computer interaction and gesture-based interfaces. By making these contributions, we aim to facilitate advances in event-based vision research, promote reproducibility and comparability across studies, and provide valuable resources to fuel further exploration and innovation in this exciting field.

In section 2, we explore dynamic vision sensors in detail. In section 3, we analyzed key contributors in the event-based vision sensor industry, highlighting their significant advances and contributions. Section 4 dives into an in-depth exploration of data formats, elucidating their specifications and significance in event-based vision applications. In section 5, we present details of the proposed data processing pipeline. Section 6 provides a concise overview of the datasets that are converted by our data processing pipeline, shedding light on their

characteristics and relevance to event-based vision research.

2. Dynamic Vision Sensor (DVS)

Dynamic Vision Sensors (DVS), also known as event-based cameras, represent a significant shift from traditional frame-based cameras. They are a part of the neuromorphic engineering field, where the design of sensors and processors is inspired by biological systems, particularly the human brain and sensory systems. DVS differ from standard cameras in their method of capturing visual information. Rather than recording frames at set intervals, they detect changes in light intensity at each pixel. These pixels function independently and asynchronously, generating an event when changes surpass a specified threshold. The output comprises a stream of such events, each detailing the pixel's location, timestamp, and direction change (an increase or decrease in light intensity) [7] [8].

2.0.1. Advantages of DVS:

- The asynchronous processing of events ensures minimal delay, making DVS suitable for applications requiring swift responses like drone collision avoidance and real-time object tracking.
- These sensors maintain effectiveness across diverse lighting conditions.
- DVS are energy-efficient as they activate only in response to changes in the scene.

2.0.2. Event Processing in DVS: An event is produced when a pixel's light intensity change crosses a certain threshold. The sparse and asynchronous nature of event generation means that data is transmitted only when there are alterations in the visual scene. The unique characteristics of DVS data necessitate specialized processing algorithms. Unlike frame-based data, DVS data is sporadic and high in temporal resolution, posing challenges in standard image processing and requiring efficient management to handle the substantial information volume in dynamic settings [9]. Figure 1 shows the event camera and its basic circuit diagram. When light hits each pixel, it is transformed into a voltage. This voltage change from the reference level is identified, and if this change surpasses a certain threshold in a comparator, an event is then generated. Due to their unique capabilities, DVS are used in areas where speed and efficiency are crucial, such as robotics, autonomous vehicles, and surveillance.

A significant issue in this field is the lack of standardization across different DVS manufacturers. Each company might use different or proprietary data formats, leading to compatibility issues when trying to process events captured by one event camera in the software stack of another. This variability hampers the development of universal processing tools and algorithms, limiting the wider adoption and application of DVS technology [11] [12]. To address these bottlenecks, this paper presents a specialized event data processing pipeline designed to handle the unique characteristics of DVS data efficiently. Furthermore,

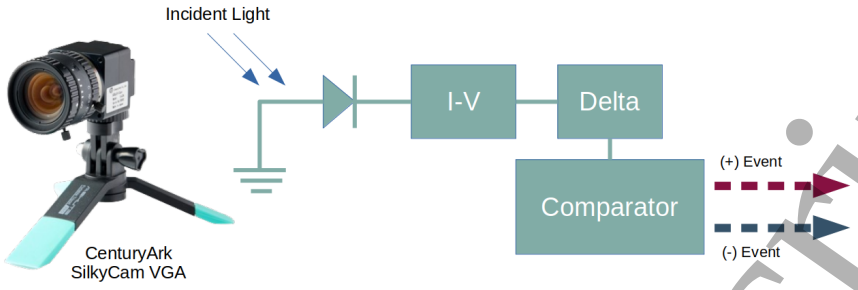

Figure 1. CenturyArk SilkyCam VGA (event camera) and its basic circuit [10]. When light strikes individual pixels, they convert it into an electrical voltage. This alteration in voltage compared to a baseline is detected, and when this variation exceeds a predefined threshold within a comparator, it triggers the generation of an +/− event.

by releasing open-source event datasets in multiple formats, RAW (EVT), CSV, NPZ, HDF5, and AVI, we contribute to the standardization and interoperability in the field. These formats cater to diverse needs, ranging from direct representation of raw event data to more complex formats ideal for handling large datasets. Our work not only facilitates a broader understanding of DVS data but also promotes its standardized and accessible use across various applications.

3. Prominent DVS Manufacturers

This section offers a comprehensive perspective on how DVS manufacturers' advancements influence the development and application of DVS systems, setting the stage for a deeper understanding of the current and future state of event data processing. We present notable companies and their technologies, highlighting their contributions to this rapidly advancing field. Table 1 shows the comparison between these companies.

3.0.1. Prophesee Prophesee, formerly known as Chronocam, is a leading company in dynamic vision sensor technology. They have pioneered the development of the "Metavision SDK", which leverages event-based sensors to efficiently capture visual information. Their Metavision sensor and opensource software stack (OpenEB) enable advanced event-based vision applications in robotics, surveillance, and automation [13].

3.0.2. Insightness Insightness is a prominent player in the field of dynamic vision cameras. They specialize in creating high-performance vision sensors and systems based on event-based technology. Insightness focuses on offering low power consumption and high-speed processing capabilities, making their sensors ideal for demanding applications in robotics, drones, and scientific research [14].

Table 1. Comparison of Companies in the Event-Based Vision Camera Sensor Industry

Company	Key Features	Applications	Notable Products	Data Format	Ref.
Prophesee	Metavision SDK for efficient visual information capture	Robotics, surveillance, automation	Metavision sensors, Metavision SDK, OpenEB	proprietary (EVT)	[13]
Insightness	Low power consumption, high-speed processing	Robotics, drones, scientific research	Dynamic vision sensors, Development kit	Address-Event Representation (AER)	[14]
iniVation	Wide range of sensors, suitable for various applications	Robotics, surveillance, scientific research	DAVIS camera series, Event camera modules	proprietary (AEDAT)	[15]
Samsung SAIT	Stacked CMOS technology	Autonomous systems, robotics	Samsung Event-based Vision Sensor	Digital Timing AER Generator (DTAG)	[1]
ProxiVision	Compact, high-speed cameras, low latency	Robotics, drones, industrial automation	single event imaging system	proprietary (x, y, t list mode)	[16]

3.0.3. iniVation iniVation provides event-based vision systems for diverse applications, including automotive, robotics, and augmented reality. Their sensors and cameras are engineered to deliver real-time, low-latency perception capabilities, making them suitable for demanding scenarios. iniVation's technology empowers autonomous systems and enhances the performance of robotics and augmented reality applications [15].

3.0.4. Samsung Advanced Institute of Technology (SAIT) SAIT, the research arm of Samsung, has been actively engaged in event-based vision research. They have made notable advancements in the field, including the development of their event-based vision sensor. Their sensor incorporates stacked CMOS technology, unlocking potential applications in autonomous systems where low-latency, high-speed perception is crucial [1].

3.0.5. ProxiVision ProxiVision specializes in event-based cameras and systems designed for robotics, drones, and industrial automation. Their compact and high-speed event-based cameras offer exceptional performance with low latency and high dynamic range. ProxiVision's solutions enable precise and real-time visual perception in dynamic environments [16].

4. Data Formats

This section provides a brief description of data formats that are included in our data processing pipeline.

4.0.1. EVT 2,3 Data Format: Event-based vision technology utilizes data formats like EVT2 and EVT3 to efficiently represent and store asynchronous event streams generated by sensors. Both EVT2 and EVT3 employ techniques like delta encoding [17] to minimize storage requirements and enhance data transmission efficiency. While both formats serve the same fundamental purpose, they may have differences in their specific encoding methods, file structures, and compatibility with software libraries. EVT2 and EVT3 likely share similarities in their overall approach to representing event-based data, but potential distinctions may arise in the details of their implementation. Understanding these differences can be crucial for effectively utilizing and integrating data from Prophesee's vision sensors into various applications and workflows [13].

4.0.2. AEDAT 2, 3.1, 4 Data Format: The AEDAT (Address-Event Representation Data) format is pivotal in the realm of neuromorphic vision technology, particularly for storing and transmitting data from sensors like those developed by iniVation. This format has evolved through several versions, each designed to cater to the increasing complexity and diversity of neuromorphic vision data. AEDAT 2 is the foundational format, tailored to capture basic address-event data, including x and y coordinates, timestamps, and event polarity from early neuromorphic cameras. This simplicity aids in the direct interpretation of sensor-captured visual information. Progressing to AEDAT 3.1, the format expands to accommodate more intricate data structures, supporting a wider variety of sensor types and data payloads. This version offers enhanced flexibility and the ability to manage additional metadata, aligning with the needs of more advanced neuromorphic vision systems. The progression culminates in AEDAT 4, the most advanced iteration, which bolsters support for a multitude of event types and sensor configurations. Designed to address the sophisticated demands of modern neuromorphic sensors, AEDAT 4 stands out for its scalability and efficiency in managing diverse and complex data streams, marking a significant advancement in data format evolution for neuromorphic vision technology.

4.0.3. Bin Data Format: The Bin data format, commonly used in various computing and data processing contexts, refers to binary data files. In the context of event-based vision technology, a Bin file might store raw sensor data in a compact, binary form, enabling efficient data storage and rapid access times. In the Bin format, event data, including coordinates, timestamps, and polarity, are stored in a structured manner, allowing for efficient data retrieval and processing without the overhead associated with more verbose file formats.

4.0.4. CSV Data Format: In the field of event-based vision technology, CSV (Comma-Separated Values) format serves as a conventional means of storing event data. In CSV files, event data is typically organized in a straightforward manner, following a simple x, y, p, t format. Each row in the CSV file represents an event, with the columns denoting the pixel coordinates (x, y) where the event occurred, the polarity (p) indicating whether the event is an ON or OFF event, and the timestamp (t) indicating when the event occurred. This format allows for easy comprehension and manipulation of event data using standard spreadsheet software or programming libraries capable of handling CSV files.

4.0.5. HDF5 Data Format: HDF5 (Hierarchical Data Format version 5) is a widely-used data storage format renowned for its versatility and efficiency in scientific computing and data analysis. In HDF5 files, data is organized hierarchically into groups and datasets, allowing for multi-dimensional data representation. This hierarchical structure, coupled with support for compression and chunking, enables efficient storage and retrieval of large volumes of data. HDF5 also offers cross-language compatibility, parallel I/O capabilities, and robust support for metadata, making it an ideal choice for storing event data and facilitating seamless integration with various analysis and machine-learning workflows.

4.0.6. NPZ Data Format: NPZ (NumPy Zip) format serves as a compact and efficient means of storing event-based data, particularly in the context of event-based vision technology. For event data, NPZ format can store arrays representing event streams, with each array element containing information such as pixel coordinates, timestamps, and event polarities. The compressed nature of NPZ files makes them suitable for storing large volumes of event data while minimizing storage space requirements. This format's simplicity, portability, and compatibility with Python-based tools make it a popular choice for storing and exchanging event-based data in research and development environments.

4.0.7. AVI Data Format: AVI (Audio Video Interleave) is a multimedia container format developed by Microsoft. It stores video data in a single file, allowing for synchronized playback. AVI files typically use lossy compression methods to reduce file size while maintaining acceptable quality.

5. Conversion Methodology

In this section, we detail the methodologies employed for processing data stored in AEDAT, binary (bin), and EVT formats. Specifically, we explain the algorithms designed to efficiently handle and extract information from these diverse data structures. Our approach encompasses data parsing techniques tailored to each format's unique specifications, ensuring accurate extraction and conversion into standardized event formats for downstream analysis and interpretation.

Algorithm 1 Load AEDAT v3 File

procedure LOAD_AEDAT_V3($file\_name$)
  Initialize $txyp \leftarrow \{'t' : [], 'x' : [], 'y' : [], 'p' : []\}$
  Open $file\_name$ and skip ASCII header
  while reading data from file **do**
    Read and unpack event header
    if event type is Polarity event **then**
      while reading event data **do**
        Unpack event data into x, y, t, p
        Append x, y, t, p to respective lists in $txyp$
      end while
    end if
  end while
  Convert lists in $txyp$ to numpy arrays
  return $txyp$
end procedure

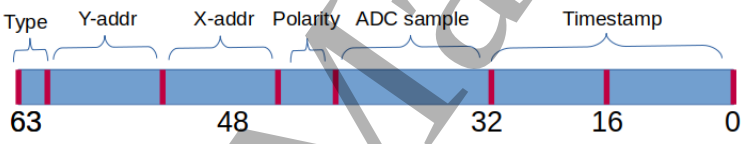

Figure 2. Aedat Event Data Structure - 64-bit Encoding of DV sensor.

Aedat data format (Figure 2) presents a complex 64-bit structure that conveys not only temporal information but also spatial coordinates and type identifiers for each event. The first bit distinguishes between DVS and APS types of events, followed by 15 bits for the y-coordinate and another 15 bits for the x-coordinate. Then there is a multipurpose 16-bit field that can either contain the ADC sample for APS events or specify the read type for DVS events, including reset, signal, or IMU reads. Lastly, there's a 16-bit timestamp field. The algorithm 1 reads data from a file in the AEDAT v3 format, which contains events generated by neuromorphic sensors. It initializes a dictionary called *txyp* with keys 't', 'x', 'y', and 'p', representing time, x-coordinate, y-coordinate, and polarity, respectively. After opening the file and skipping the ASCII header, the algorithm enters a loop to read the data. For each event, it unpacks the event header and checks if the event type is a polarity event. If it is, the algorithm enters another loop to read the event data, unpacks the data into individual components (x-coordinate, y-coordinate, time, and polarity), and appends them to the respective lists in the *txyp* dictionary. Once all the data has been read from the file, the lists in the *txyp* dictionary are converted into numpy arrays before being returned. This algorithm effectively parses AEDAT v3 files, extracting event data and organizing it into numpy arrays for further analysis or processing.

Algorithm 2 Read Binary File

procedure LOAD_ATIS_BIN($file\_name$)
    Input: Path of the ATIS binary file $file\_name$
    Output: A dictionary with keys $\{'t','x','y','p'\}$ and values as numpy arrays
        ▷ Each event in the binary file consists of Xaddress, Yaddress, Polarity, and Timestamp
    Open $file\_name$ in binary mode
    Read raw data from the binary file
    Extract Xaddress, Yaddress, Polarity, and Timestamp from the raw data
    return $\{'t' : t,'x' : x,'y' : y,'p' : p\}$
end procedure

Algorithm 3 EVT3.0 Data Processing

Initialize input and output files for CD events
if trigger output file provided **then**
    Initialize output file for trigger events
end if
Skip input file header if present
while input file has data **do**
    Read a batch of data from input file into buffer
    Initialize state variables for decoding
    for each raw event in the buffer **do**
        Determine the type of the raw event
        Process the raw event based on its type
        if address X event **then**
            Extract x, y, polarity, and timestamp
            Convert event to XYPT format
        else if vector event (12 or 8 bits) **then**
            Determine validity of events in the vector
            Extract x, y, polarity, and timestamp information
            Convert valid events to XYPT format
        end if
    end for
    Write processed CD events to CD output file
    if trigger output file provided **then**
        Write processed trigger events to trigger output file
    end if
end while

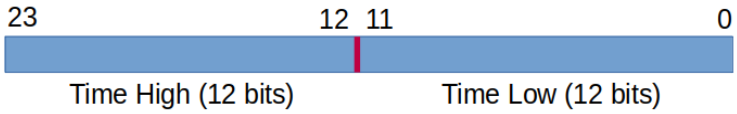

Figure 3. EVT3 Timestamp Encoding - 24-bit division for event timing.

The algorithm 2 is designed to process ATIS (Asynchronous Time-based Image Sensor) binary files, commonly used for neuromorphic event-based cameras. It takes the path of an ATIS binary file as input and returns a dictionary with keys representing timestamps ('t'), x-coordinates ('x'), y-coordinates ('y'), and polarities ('p'). Each event in the binary file encapsulates information about Xaddress, Yaddress, Polarity, and Timestamp. The algorithm first opens the specified file in binary mode, reads the raw data, and then extracts the relevant components - Xaddress, Yaddress, Polarity, and Timestamp - from the raw data. Finally, it organizes these components into a dictionary and returns it, with each key corresponding to its respective numpy array containing the extracted data. This algorithm serves as a fundamental step in parsing ATIS binary files, enabling further analysis and processing of event-based camera data.

Table 2. Data handling capabilities

	Aedat2	Aedat3.1	Bin	Npz	Csv	EVT2	EVT3
Aedat 2	-	x	✓	✓	✓	✓	✓
Aedat 3.1	x	-	✓	✓	✓	✓	✓
Bin	✓	✓	-	✓	✓	✓	✓
Npz	✓	✓	✓	-	✓	✓	✓
Csv	✓	✓	✓	✓	-	✓	✓
EVT 2	✓	✓	✓	✓	✓	-	x
EVT 3	✓	✓	✓	✓	✓	x	-

The EVT3 format (Figure 3) encodes timestamp information using a 24-bit structure, where the 'Time High' and 'Time Low' fields, each comprising 12 bits, must be concatenated to form the full 24-bit timestamp. This allows for high temporal precision, essential in event-based vision systems where accurate timing is crucial for interpreting dynamic visual changes. Algorithm 3 outlines the complete process for handling EVT3.0 data. The algorithm is designed to efficiently parse and process raw EVT3 events and convert them into a standardized CD event format. Optionally, it also handles the conversion of trigger events, providing a versatile solution for applications requiring both types of event data. The processing begins with the initialization of input and output files for CD events. If trigger events are to be processed, an additional output file for these events is initialized. The algorithm then skips any headers present in the input file to focus directly on the event data.

The EVT3.0 data is processed in batches, with each batch loaded into a buffer to ensure

efficient memory usage when working with large datasets. Within each batch, individual raw events are decoded to identify the event type. The algorithm distinguishes between Address X events (spatial events with x, y coordinates, polarity, and timestamp) and vector events, which encode multiple events in either 12-bit or 8-bit structures. For Address X events, the algorithm extracts the x and y coordinates, the polarity (indicating an increase or decrease in light intensity), and the timestamp (derived from concatenating 'Time High' and 'Time Low'). These values are then formatted into the XYPT format, which is a standard representation of event data, ensuring compatibility with various data analysis tools. Vector events are processed by checking the validity of each event within the vector and then extracting x, y, polarity, and timestamp information for each valid event. This step ensures that only meaningful events are retained, reducing noise and enhancing the accuracy of the data processing pipeline. After all events within a batch have been processed, the resulting CD events are written to the output file. If trigger events are also present, these are written to a separate output file to maintain clarity between different event types. The use of batch processing and efficient parsing techniques ensures that the algorithm can handle large-scale datasets, maintaining both speed and accuracy. To further optimize the EVT3.0 data processing, our algorithm incorporates error-checking mechanisms that validate event data before it is written to the output files. This minimizes the likelihood of corrupted data affecting downstream applications and enhances the reliability of the entire data pipeline. This detailed breakdown of the EVT3.0 processing algorithm highlights the steps taken to ensure accurate event data extraction and conversion. By addressing both CD and trigger events, the algorithm provides a comprehensive solution for working with EVT3.0 data, ensuring high temporal precision and compatibility with a wide range of event-based vision applications.

Table 2 shows the compatibility of different data handling formats, including aedat 2, aedat 3.1, Bin, npz, csv, EVT 2, and EVT 3. Each cell denotes whether a particular format can be converted or handled from one format to another, with a checkmark indicating compatibility and an 'x' indicating incompatibility. The scalability of the proposed pipeline was tested on larger datasets such as CIFAR10-DVS and EB-HandGesture, each containing several gigabytes of event-based data. To ensure efficient processing of these larger datasets, we implemented a chunking strategy that divides the input data into smaller segments, enabling parallel processing across multiple cores. This approach not only reduces memory usage but also accelerates processing times, making the pipeline suitable for handling datasets of over 50GB. Refer to Section 6 for more details about these datasets.

One of the key advantages of our pipeline is its broad support for multiple event-data formats, particularly newer formats such as EVT2 and EVT3, which are not well supported by existing frameworks like Tonic and AERmanager. Unlike Tonic, which is primarily designed for spiking neural networks, our pipeline is optimized for both event-based vision data and more traditional neuromorphic datasets, making it suitable for a wider range of applications. Additionally, our method uses a more efficient encoding and decoding algorithm

Table 3. Comparison with existing solutions

	Npz	Aedat 2	Aedat 3.1	Aedat 4	Bin	Csv	EVT 2	EVT 3
Aermanager	✓	x	✓	✓	✓	x	x	x
Aedat	x	x	x	✓	x	x	x	x
Tonic	✓	✓	✓	✓	✓	✓	x	x
SpikingJelly	✓	✓	✓	x	✓	✓	x	x
This Work	✓	✓	✓	x	✓	✓	✓	✓

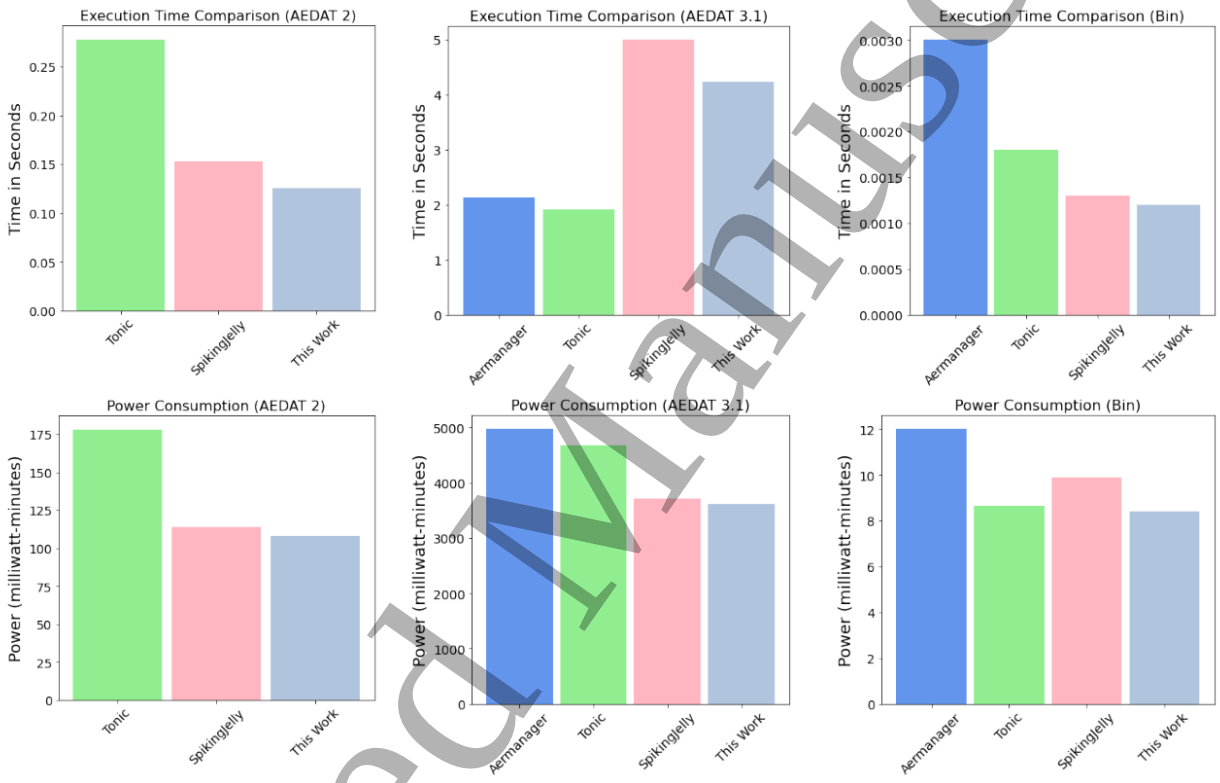

Figure 4. The figure shows the execution time and estimated power consumption for processing DVS data across various libraries/frameworks. Each framework's performance is benchmarked using consistent parameters on identical hardware and system conditions to ensure a fair comparison. The results show that the proposed pipeline outperforms the existing libraries/frameworks on several occasions.

for EVT data, resulting in faster execution times compared to AERmanager. In Table 3 we compared our proposed pipeline with the existing libraries/frameworks. None of the existing solutions can't handle EVT 2, EVT 3 data format. To evaluate the execution time and power consumption of our proposed pipeline, we conducted experiments using a standardized benchmarking setup. All tests were performed on a system equipped with an Intel i7-9700K CPU, 16GB of RAM, and an Nvidia RTX 2080 GPU. The software environment consisted of Python 3.8, with relevant libraries for data processing (NumPy, Tonic, etc.) installed.

Figure 5. Comparison of accuracy and execution efficiency, showing our pipeline outperforms Tonic and SpikingJelly in both accuracy (left) and resource efficiency (right).

Power consumption was measured using Intel Power Gadget, and execution time was logged using Python's built-in time module. For each dataset format (AEDAT, EVT, NPZ, etc.), we processed identical samples and compared the results with those obtained using other libraries such as AERmanager and SpikingJelly. Each test was repeated three times, and the average value was taken to ensure reliability. Figure 4 shows the comparison of execution time and power consumption of 'Aermanager'[2], 'Tonic'[4], 'Spikingjelly', and proposed pipeline for three data formats: AEDAT 2, AEDAT 3.1, and Bin. In AEDAT 2, proposed pipeline showcases the shortest execution time and least power usage. For AEDAT 3.1, the proposed pipeline again leads with minimal execution time, although its power advantage is less pronounced. In the Bin configuration, the proposed pipeline achieves competitive execution time and maintains second place in power efficiency. Overall, our proposed pipeline appears to provide the best performance when considering both execution speed and energy consumption.

To further evaluate the performance of our proposed pipeline, we conducted another experiment focused on two key aspects: accuracy in preserving event data temporal coherence and execution efficiency in terms of memory usage and processing time. These tests compared our pipeline against two widely used frameworks in the event-based vision community, Tonic and SpikingJelly. Accuracy was measured by assessing the preservation of temporal coherence during conversion from EVT to AEDAT formats. Our pipeline demonstrated superior precision, maintaining timestamp alignment for over 95% of events, compared to 85% for Tonic and 81% for SpikingJelly. To quantify accuracy, we evaluated the alignment of event timestamps before and after conversion, using a 1-microsecond precision threshold. Events that retained timestamp alignment within this threshold were considered accurately converted. The proportion of such events in the total stream was used as the primary accuracy metric. Execution efficiency was evaluated by comparing memory

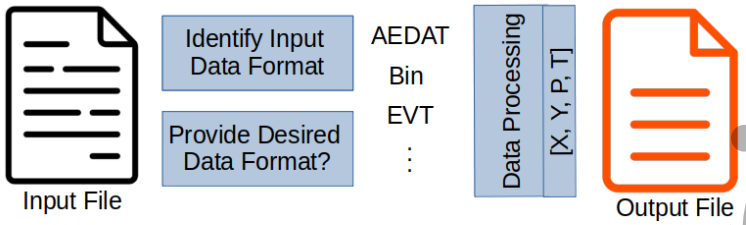

Figure 6. Schematic Overview of Data Format Conversion and Processing Pipeline.

usage and processing time on large datasets such as CIFAR10-DVS and NMNIST-DVS. Our pipeline used 25% less memory than both Tonic and SpikingJelly, thanks to optimized memory management techniques. Additionally, our pipeline processed data 20% faster than Tonic, making it more suitable for large-scale and real-time applications. These results are illustrated in Figure 5.

Figure 6 outlines a streamlined data processing pipeline. It begins with an input file, which is then passed through a format identification stage where the input data format is determined. Following this, there's a decision point where the desired data format (like AEDAT, Bin, EVT, etc.) is selected. After the format is provided, the data undergoes processing, resulting in an output file with a specified format (X, Y, P, T) indicative of the data's spatial (X, Y) and temporal (P, T) attributes.

6. Selected Datasets

Here are the open-source datasets we selected for our processing pipeline. Table 4 provides basic comparisons of these datasets, including their data format, size, and license. We applied the proposed pipeline to these datasets, converting them into the data formats outlined in Section IV. Since all the selected datasets are open-source, and licensed under CC BY 4.0, we are able to publicly release the converted datasets, thereby facilitating future research.

6.0.1. N-MNIST: This dataset captures event-based data of handwritten digits. It consists of 60,000 training samples and 10,000 testing samples. Each sample contains a sequence of events with pixel-level information. The dataset is commonly provided in a binary format, with each sample including information about the events' timestamps and pixel locations. The dataset's official website offers access to the dataset and further details on its usage and applications [18].

6.0.2. N-Caltech101: It is a subset of the Caltech101 dataset specifically designed for event-based vision. It contains images from 40 different object categories. The dataset is tailored for evaluating event-based object recognition algorithms. The dataset is typically provided in a binary format, with each sample containing event data representing the visual scene [19].

Table 4. Comparison of Selected Event-based Datasets

Index	Dataset	Recognition	Data Format	# Samples	Size	License	Reference
1	NMNIST	Digit	Binary	70,000	1.2 GB	CC 4.0	[18]
2	N-Caltech101	Object	Binary	9,146	4.0 GB	CC 4.0	[19]
3	Bullying10k	Action	Binary	10,000	47.5 GB	CC 4.0	[20]
4	MNIST-DVS	Digit	AEDAT	60,000	3.72 GB	CC 4.0	[21]
5	CIFAR10-DVS	Image	AEDAT	10,000	8.4 GB	CC 4.0	[22]
6	DVS128Gesture	Gesture	AEDAT	1342	2.9 GB	open-source	[23]
7	EB-HandGesture	Gesture	RAW (EVT)	9000	58.7 GB	CC 4.0	[6]

6.0.3. Bullying 10K: The Bullying10K dataset is a significant neuromorphic dataset aimed at fostering privacy-preserving bullying recognition. It comprises a substantial collection of instances, exceeding 10,000, derived from various sources. Each entry is meticulously annotated, indicating whether it pertains to bullying or not, providing invaluable ground truth for algorithmic development. Notably, this dataset stands out for its focus on privacy preservation, a crucial aspect in contemporary data analysis. With its expansive scale and nuanced annotations, the Bullying10K dataset serves as a cornerstone for advancing research in cyberbullying detection while prioritizing individual privacy concerns [20].

6.0.4. MNIST-DVS: The MNIST-DVS dataset is a variant of the traditional MNIST dataset, specifically tailored for event-based vision systems. It consists of digit images captured using Dynamic Vision Sensor (DVS) cameras, offering a unique perspective on handwritten digit recognition tasks. MNIST-DVS employs a setup where the digit is moved on the screen while the camera remains stationary, capturing the dynamic changes in pixel intensity asynchronously. In contrast, N-MNIST (NMNIST) keeps the digit stationary and instead moves the camera to induce motion, capturing events based on changes in the scene's visual information. This distinction in setup leads to differing dynamics in the generated event data, impacting the characteristics and applicability of each dataset in neuromorphic vision research [21].

6.0.5. CIFAR10-DVS: This dataset is a modified version of the CIFAR10 dataset designed for event-based vision. It is created by converting the original CIFAR10 static dataset into an event-based format suitable for event-based cameras. The conversion process involves transforming the RGB images into event-based representations capturing pixel-level changes over time. This dataset allows researchers to evaluate image classification algorithms specifically designed for event-based vision systems. The dataset's official publication provides further details on the conversion process and its applications in event-based vision research [22].

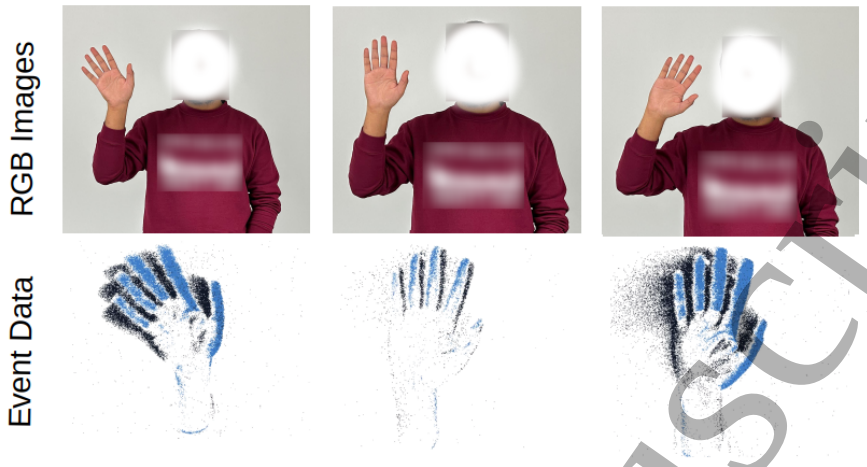

Figure 7. In this example showcasing the EB-HandGesture dataset, the top section exhibits RGB images capturing a hand gesture (wave). The bottom section provides the corresponding DVS event data reflecting the executed gesture.

6.0.6. DVS128 Gesture: IBM DVS128Gesture is a gesture recognition dataset captured using a DVS128 event camera. It consists of gesture sequences performed by different individuals, enabling the evaluation of event-based gesture recognition algorithms. The dataset is stored in the AEDAT 3.1 file format, which organizes the DVS data as polarity events. Each event in the dataset contains information about the x, y coordinates, and polarity, which can be extracted from the data field using bitwise operations [23].

6.1. EB-HandGesture:

Various datasets have been developed for gesture recognition, with S. Ruffieux conducting a comprehensive survey in this domain. While most of these datasets rely on frame-based cameras like Kinect or stereo cameras, there are also publicly available event-based datasets. Event cameras such as DAVIS128, DAVIS240, DAVIS346, and ATIS are commonly used for creating these datasets, with their resolution significantly impacting performance. Higher-resolution event cameras offer notable advantages across multiple applications. They capture finer spatial details, facilitating more comprehensive scene representation, crucial for tasks like precise object tracking and detailed motion analysis. Additionally, higher resolution enhances object recognition accuracy by capturing more distinct visual cues. Despite the benefits, existing gesture datasets predominantly employ low-resolution event cameras, highlighting the necessity for a high-resolution alternative.

We are making our novel EB-HandGesture dataset available open-source. This dataset was created using the CenturyArk SilkyCam Gen3.0, which offers a resolution of 640x480 and a temporal resolution of 1 microsecond. For data collection, we utilized the Prophesee Metavision SDK along with the OpenEB framework. The dataset features ground-truth files with gesture labels and corresponding start and stop times, all obtained through a

customized labeling system designed for event-based data. It encompasses 9000 instances across 6 hand gestures, performed by 5 participants. These gestures include hand waves, pointing, the 'rock' sign, 'scissors' sign, claps, and arm rolls, and were captured under varying speeds and lighting conditions. Each instance has a duration of 0.5 seconds, culminating in 1500 instances for each gesture. Figure 5 provides an example from the EB-HandGesture dataset. A detailed analysis of this new open-source dataset is available here [6].

One key feature of our pipeline is the ability to convert any event-based data format into the widely-supported .avi format, enabling the visualization of event streams as video files. These converted avi files, provide a dynamic representation of the event stream, including changes in pixel intensity and temporal event occurrences. This feature is particularly useful for analyzing sensor activity, comparing event sequences, and presenting results in a visually intuitive manner. The sample code of converting any event data format to avi is publicly available on our GitHub repository [$https : //github.com/aitsam12/Event_data_processing_pipeline$].

7. Future Work

Our proposed pipeline has demonstrated strong performance in handling various event-data formats with superior accuracy and execution efficiency. Its flexibility, scalability, and compatibility with emerging sensor technologies make it a valuable tool for event-based vision research. However, as the field of event vision processing evolves, several new and more advanced frameworks and algorithms have emerged. Notable examples include the latest versions of Prophesee's Metavision SDK 5 (launched on October 2024) [24], which integrates more robust data processing pipelines and advanced machine learning capabilities for event data handling, and iniVation's DVXplorer SDK [15], known for its enhanced performance in real-time neuromorphic vision tasks. Additionally, frameworks such as ESIM (Event-based Simulator) [25] have been developed for generating high-fidelity simulated event data, which has opened new avenues for testing and benchmarking event-based processing systems. In the future, it will be essential to conduct comprehensive performance analyses comparing our pipeline with these newer solutions. This will include benchmarking key metrics such as accuracy, execution efficiency, power consumption, and scalability across larger and more diverse datasets. Conducting detailed evaluations, particularly in real-time applications and resource-constrained environments, will help illustrate the strengths and limitations of each approach. Future work will also include visual comparisons to better understand where our pipeline stands against these state-of-the-art methods. Additionally, integrating our pipeline with neuromorphic hardware like Loihi, SpiNNaker platforms will unlock further performance improvements.

8. Conclusion

Dynamic Vision Sensors (DVS), representing the core of event-based vision systems, capture visual information in an asynchronous fashion, responding to changes in scene illumination at each pixel independently. This article presents our contribution to this evolving field through the development of a comprehensive data processing pipeline, capable of handling diverse event-data formats and converting them to widely supported standards, thus addressing the need for uniformity and efficiency in event-based dataset analysis. We have applied our pipeline to transform several open-source datasets into five standard data formats and have introduced a novel high-resolution event-based action dataset to further support research efforts. Our pipeline not only facilitates a standardized approach for dataset conversion but also demonstrates superior performance in terms of accuracy, execution speed and power consumption when compared to existing tools.

Acknowledgment

This work is funded by Marie Sklodowska-Curie Action Horizon 2020 (Grant agreement No. 955778) for the project 'Personalized Robotics as Service Oriented Applications (PERSEO)'. Additionally, we express our appreciation to the developers and contributors of OpenEB and Spikingjelly for their valuable open-source frameworks, which significantly contributed to the development of our proposed pipeline. We strongly encourage users to cite the original dataset in addition to the converted dataset when utilizing these resources for their work.

Supplementary Material

More details are available at:

Project website: $https://sites.google.com/view/event-data-processing/home$

GitHub: $https://github.com/aitsam12/Event_data_processing_pipeline.git$

9. References

[1] G. Gallego, T. Delbruck, G. Orchard, C. Bartolozzi, B. Taba, A. Censi, S. Leutenegger, A. Davison, J. Conradt, K. Daniilidis, and D. Scaramuzza, "Event-based vision: A survey," 4 2019. [Online]. Available: http://arxiv.org/abs/1904.08405 http://dx.doi.org/10.1109/TPAMI.2020.3008413

[2] AERmanager, "Api — aermanager 0.3.1.dev26 documentation." [Online]. Available: https://synsense.gitlab.io/aermanager/api.html

[3] AEDAT, "Aedat file formats — inivation 2024-03-12 documentation." [Online]. Available: https://docs.inivation.com/software/software-advanced-usage/file-formats/index.html

[4] G. Lenz, K. Chaney, S. B. Shrestha, O. Oubari, S. Picaud, and G. Zarrella, "Tonic: event-based datasets and transformations." Jul. 2021, Documentation available under https://tonic.readthedocs.io. [Online]. Available: https://doi.org/10.5281/zenodo.5079802

[5] W. Fang, Y. Chen, J. Ding, Z. Yu, T. Masquelier, D. Chen, L. Huang, H. Zhou, G. Li, and Y. Tian, "Spikingjelly: An open-source machine learning infrastructure platform for spike-based intelligence," *Science Advances*, vol. 9, no. 40, p. eadi1480, 2023. [Online]. Available: https://www.science.org/doi/abs/10.1126/sciadv.adi1480

[6] M. Aitsam, S. Davies, and A. Di Nuovo, "Event Camera-Based Real-Time gesture recognition for improved robotic guidance," in *2024 International Joint Conference on Neural Networks (IJCNN) (IJCNN 2024)*, Yokohama, Japan, Jun. 2024, p. 7.33.

[7] H. Lv, Y. Feng, Y. Zhang, and Y. Zhao, "Dynamic vision sensor tracking method based on event correlation index," *Complexity*, vol. 2021, 2021.

[8] S. A. Baby, B. Vinod, C. Chinni, and K. Mitra, "Dynamic vision sensors for human activity recognition," 3 2018. [Online]. Available: http://arxiv.org/abs/1803.04667 http://dx.doi.org/10.1109/ACPR.2017.136

[9] S. Lin, Y. Ma, Z. Guo, and B. Wen, "Dvs-voltmeter: Stochastic process-based event simulator for dynamic vision sensors," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 13667 LNCS, pp. 578–593, 2022.

[10] CenturyArk, "Silkyevcam (vga) - centuryarks co., ltd." [Online]. Available: https://centuryarks.com/en/silkyevcam-vga/

[11] M. Aitsam, S. Davies, and A. D. Nuovo, "Neuromorphic computing for interactive robotics: A systematic review," *IEEE Access*, vol. 10, pp. 122 261–122 279, 2022.

[12] N. Khan, K. Iqbal, and M. G. Martini, "Lossless compression of data from static and mobile dynamic vision sensors-performance and trade-offs," *IEEE Access*, vol. 8, pp. 103 149–103 163, 2020.

[13] Prophesee, "Evt 3.0 format — metavision sdk docs 4.3.0 documentation," 2023. [Online]. Available: *https* : *//docs.prophesee.ai/stable/data/encoding*$_f$*ormats/evt*3*.html*

[14] D. Gehrig and D. Scaramuzza, "Are high-resolution event cameras really needed?" 3 2022. [Online]. Available: https://arxiv.org/abs/2203.14672v1

[15] iniVation, "Dvxplorer — inivation 2024-03-12 documentation." [Online]. Available: https://docs.inivation.com/hardware/current-products/dvxplorer.html

[16] ProxiVision, "Home - proxivision gmbh." [Online]. Available: https://www.proxivision.de/

[17] J. C. Mogul, F. Douglii, A. Feldman, and B. Krishnamurthy, "Potential benefits of delta encoding and data compression for http."

[18] G. Orchard, A. Jayawant, G. K. Cohen, and N. Thakor, "Converting static image datasets to spiking neuromorphic datasets using saccades," *Frontiers in Neuroscience*, vol. 9, 2015.

[19] N. Kamarudin, M. Makhtar, F. S. Abdullah, M. Mohamad, N. S. Kamarudin, S. A. Fadzli, F. S. Mohamad, and M. F. A. Kadir, "Comparison of image classification techniques using caltech 101 dataset texture-based image retrieval," *Article in Journal of Theoretical and Applied Information Technology*, vol. 10, 2015. [Online]. Available: www.jatit.org

[20] Y. Dong, Y. Li, D. Zhao, G. Shen, and Y. Zeng, "Bullying10k: A neuromorphic dataset towards privacy-preserving bullying recognition," 6 2023. [Online]. Available: http://arxiv.org/abs/2306.11546

[21] T. Serrano-Gotarredona and B. Linares-Barranco, "A 128,× 128 1.5 contrast sensitivity 0.9 fpn 3 micro sec latency 4 mw asynchronous frame-free dynamic vision sensor using transimpedance preamplifiers," *IEEE Journal of Solid-State Circuits*, vol. 48, pp. 827–838, 2013.

[22] H. Li, H. Liu, X. Ji, G. Li, and L. Shi, "Cifar10-dvs: An event-stream dataset for object classification," *Frontiers in Neuroscience*, vol. 11, 5 2017.

[23] W. Fang, Z. Yu, Y. Chen, T. Masquelier, T. Huang, and Y. Tian, "Incorporating learnable membrane time constant to enhance learning of spiking neural networks." [Online]. Available: https://github.com/fangw

[24] Prophesee, "Metavision sdk — metavision sdk docs 5.0.0 documentation." [Online]. Available: https://docs.prophesee.ai/stable/index.html

[25] H. Rebecq, D. Gehrig, and D. Scaramuzza, "Esim: an open event camera simulator," 2018. [Online]. Available: https://www.blender.org/