IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received March 18, 2020, accepted April 14, 2020, date of publication May 22, 2020, date of current version June 12, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2996661

Lossless Compression of Data From Static and Mobile Dynamic Vision Sensors-Performance and Trade-Offs

NABEEL KHAN, KHURRAM IQBAL, AND MARIA G. MARTINI, (Senior Member, IEEE)
Wireless Multimedia and Networking Research Group, Kingston University, London KT1 2EE, U.K.

Corresponding author: Nabeel Khan (n.khan@kingston.ac.uk)

This work was supported by the EPSRC via Internet of Silicon Retinas: Machine to Machine Communications for Neuromorphic Vision Sensing Data (IoSiRe) under Grant EP/P022715/1.

ABSTRACT Dynamic Vision Sensors (DVS) are emerging retinomorphic visual capturing devices, with great advantages over conventional vision sensors in terms of wide dynamic range, low power consumption, and high temporal resolution. The bio-inspired approach of the DVS results in lower data rates than conventional vision sensors. Still, such data can be further compressed. Compression of DVS data is an emerging research area and a detailed performance comparison of different compression strategies for these data is still missing. This paper addresses lossless compression strategies for data output by neuromorphic visual sensors. We compare the performance of a number of strategies, including the only strategy developed specifically for such data and other more generic data compression strategies, tailored here to the case of neuromorphic data. We perform the comparison in terms of compression ratio, as well as compression and decompression speed and latency. Moreover, the compression performance analysis is performed under diverse scenarios including stationary and mobile DVS. According to the detailed experimental analysis, Lempel-Ziv-Markov chain algorithm (LZMA) achieves the best compression ratios among all the considered strategies for the case when the DVS is static. On the other hand, Spike coding achieves the best compression ratios under the scenario when spike events are produced by a sensor in motion. However, both strategies result in low compression speed and high latency which restrict the applications of these strategies in real-time scenarios. The Brotli strategy achieves the best trade-off between compression ratio, speed and latency under static as well as mobile scenarios. We also observe a significant decrease in compression and decompression performance (in terms of ratio, speed and latency) of all the strategies under mobile DVS scenarios.

INDEX TERMS Dynamic vision senor, neuromorphic computing, computer vision, spike coding, data compression, dictionary based compression, integer compression, IoT specifc compression, entropy coding, fast integer compression.

I. INTRODUCTION

Dynamic Vision Sensors (DVS) [1], [2] mimic the visual processing characteristics of living organisms, *i.e.,* they capture only changes in scene reflectance. Conventional videos are captured and stored in the form of synchronous frames, whereas DVS respond to the temporal luminance changes with an asynchronous stream of spike events, fired independently. The Address Event Representation (AER) protocol is utilized to output the spike event stream. A spike event consists of four essential elements, which are represented by a tuple $< X, Y, t, p >$: the spatial addresses X and Y, the time-stamp t and the polarity flag p. Spike events occur whenever there is either movement/change of lights conditions in the scene or motion of the sensor or both. In other words, there is no spike event output for static scenes and motionless sensors. The unique spike firing mechanism enables DVS to satisfy low-bandwidth, low-power and low-latency requirements. Figure 1 shows an example of the conventional camera and DVS output. The first row shows the DVS output, where spatio-temporal coordinates $\{X, Y, t\}$ and polarity p of the neuromorphic sequence are rendered as frames. In each of the

The associate editor coordinating the review of this manuscript and approving it for publication was Paolo Napoletano.

rendered DVS frames in the figure, spike events fired owing to the increase in luminance intensity (polarity $p = 1$) are represented by the ''green'' color. On the other hand, spike events associated to decrease in brightness (polarity $p = 0$) are represented by the ''red'' color.

The spike event rate of the DVS is dependent on scene complexity [3] and relative motion of the camera *w.r.t* scene objects [4]. Because the output of the DVS is quite different from conventional frame-based image sequences, existing computer vision techniques cannot be directly applied to the series of neuromorphic spike events. To address this issue, many algorithms have been specifically tailored to leverage spike events data for a diverse range of applications (object detection, classifications, optical flow estimation, etc.). A good survey of these algorithms in a wide range of applications is provided in [5].

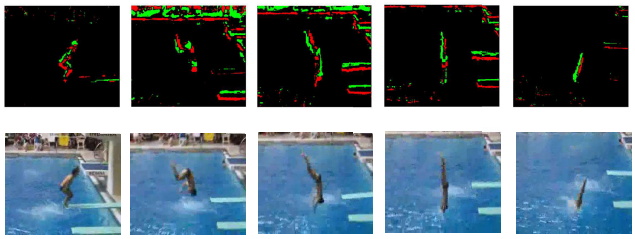

FIGURE 1. Rendered frames from DVS data (above) and video frames acquired via conventional camera (below).

A. MOTIVATION

The unique characteristics of the DVS offer advantages over conventional vision sensors in real-time interaction systems such as robotics [6], [7], drones [8] and autonomous driving [9], [10]. In the near future, most of the envisaged services performing object/gesture recognition or classification will be performed via cloud / edge computing. Therefore, these services would require the transmission of the spike events to cloud / edge servers for the processing of visual data [11]. Furthermore many scenarios, such as the collaboration of multiple intelligent cars or drones forming an Internet of Intelligent Vehicles (IoV) also require real-time transmission. Even if the data acquisition of DVS yields an inherent compression, further compression of the spike data would be highly desirable in the aforementioned scenarios.

A preliminary version of this paper has been presented at IEEE ICASSP 2020 [12]. The main extensions of this work are summarized in the subsequent sections.

B. CONTRIBUTIONS

1) COMPRESSION PERFORMANCE ANALYSIS OF DIVERSE STRATEGIES

This paper addresses lossless compression strategies, i.e., strategies where compression does not involve loss of information. The output of the DVS is a multivariate stream of integers (X and Y spatial addresses, time-stamp field, and the polarity flag). Hence, it is worth investigating the performance on DVS data of general purpose lossless compression strategies, IoT specific compression algorithms, integer based compression approaches, that we tailored for DVS data. We also investigate the performance of DVS data specific compression approach.

2) PERFORMANCE ANALYSIS UNDER DIVERSE SCENARIOS

We analyse existing lossless data compression strategies and their suitability for the compression of neuromorphic data under diverse scenarios. These scenarios include stationary and mobile sensor for indoor and outdoor conditions. Compression performance evaluation under these diverse scenarios is very important since these represent many emerging applications; for instance drones [8], self-driving cars [9], [10], and stereo vision [13]–[15] demonstrate the case of mobile DVS sensor in outdoor conditions, whereas surveillance and monitoring [16], [17] highlight the application of stationary DVS sensors, also in outdoor conditions. The work in [18], [19] explores the application of DVS in a mobile robot for the indoor environments. Furthermore, tactile sensing [7], [20] and automated fall detection monitoring [21] highlight the potential of DVS under static indoor environments.

3) PERFORMANCE ANALYSIS IN TERMS OF COMPRESSION RATIO, COMPRESSION AND DECOMPRESSION SPEED, AND LATENCY

We perform comparisons in terms of compression ratio, as well as compression and decompression speed and latency. Compression speed is an important performance indicator as compression needs to be fast enough to keep up with the rate of data ingestion, in particular if compression is performed onboard, at the sensor location. The majority of the potential applications of DVS involve complex visual processing, such as diverse types of machine and deep learning strategies [7], [9], [13], [14], [20], [21]. Therefore, the majority of DVS applications would require computation offloading, *i.e.*, transfer of resource intensive computation tasks to a cluster, grid or a cloud. Computation offloading of the compressed data over a network can overcome resource limitations of a device. In computation offloading, the decompression speed of an encoding strategy is of paramount importance since high decompression speed would reduce cloud service latency. Therefore, it is important to have an algorithm that could function efficiently in a central database or cloud storage.

Existing data compression strategies that can be used for neuromorphic visual sensor data are reviewed in Section II. We introduce the datasets considered for the tests and the simulation set-up in Section III, while comparative simulation results are reported in Sections IV and V, where the trade-offs to be addressed are also discussed. In particular, we analyse the compression performance for spike events emitted from static DVS in Section IV, whereas the compression performance of mobile DVS data is analysed in Section V. Finally, the concluding remarks appear in Section VI.

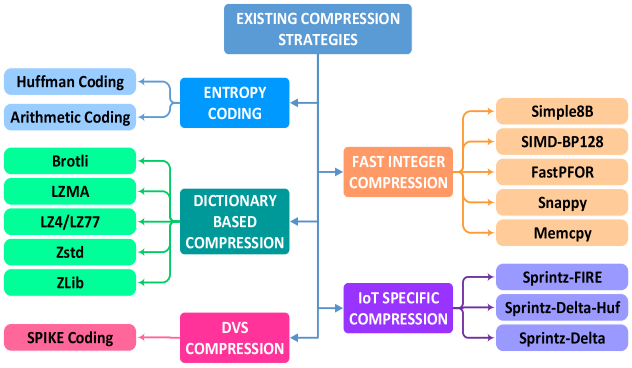

FIGURE 2. Potential compression strategies applicable to DVS Data.

II. RELEVANT DATA COMPRESSION STRATEGIES

The potential strategies to compress DVS data are reported in Figure 2, clustered in five classes, and described in detail below. Section II-A discusses the spike coding strategy specifically designed for DVS data, whereas general purpose, IoT specific, and integer based compression approaches are discussed in Section II-B.

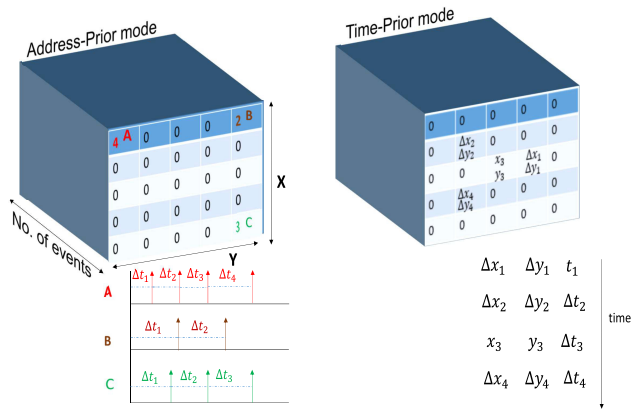

FIGURE 3. Two different modes of the spike coding strategy. Left: AP mode; Right: TP mode.

A. DVS SPECIFIC COMPRESSION APPROACH

The authors in [22] proposed the first lossless coding strategy for DVS data. According to the proposed strategy, the event sequence is partitioned into multiple three dimensional macrocubes, as shown in Figure 3, where the X and Y dimensions of the macrocube span the full spatial resolution of the pixel array (for instance, $X_{max} = 240$ and $Y_{max} = 180$ for DVS240B). The third dimension of the macrocube is the time dimension (in terms of number of spike events). Each macrocube coding has two modes which are Address-Prior (AP) and Time-Prior (TP) modes, as shown in Figure 3. The AP mode is designed for spatially decentralized macrocubes, *i.e.,* for the case where events are distributed over the entire spatial resolution. On the other hand, the TP mode is designed for spatially centralized macrocubes, *i.e.,* for the case where events occur among neighboring pixels. For each macrocube, both the modes are tried and the one achieving the best compression is chosen. The prediction residuals of the chosen mode are fed into a CABAC (Context-adaptive binary arithmetic coding) entropy coder.

The coding strategy is based on the spike firing mechanism of the DVS, which indicates the presence of temporal and spatial redundancy.

- Temporal redundancy: According to the spike firing model [22], when the luminance intensity is stably changing, linearly increasing or decreasing, the time intervals between consecutive spikes at the same pixel appear to be nearly equal, which may induce temporal correlations.
- Spatial redundancy: The adjacent pixels receive almost the same luminance concurrently, indicating the presence of spatial correlation.

1) TIME-PRIOR MODE

The TP mode of the strategy, exemplified in the right part of Figure 3, exploits spatial redundancy when the majority of the neighboring pixels elicit spike events. This mode finds a centre point (x_3, y_3 in Figure 3) and projects all other spike events *w.r.t* the centre point. In this mode, the events are projected in an increasing order of their time-stamps and delta coding is applied to the time-stamp field as shown in Figure 3. Since the time-stamp field comprises a stream of incremental integers, delta coding results in an efficient compression of this field. Spike events are centralized in the spatial field, therefore, the offsets of the spike events to the centre point result in compression of the spatial address field. The residuals of both the fields (spatial address and time-stamp) are fed to the CABAC for further compression.

2) ADDRESS-PRIOR MODE

The AP mode exploits temporal redundancy of the spike firing mechanism. Since events are spread over the entire spatial resolution, the TP mode cannot achieve high compression gains for the spatial address field, *i.e.,* the offsets of the events to the centre point cannot efficiently reduce the size of the spatial addresses. In AP mode, the number of spikes at each pixel is recorded, as shown in the left part of Figure 3. Since the time intervals between consecutive events at a pixel are approximately equal (as shown below in the left part of Figure 3), differencing (delta coding) between successive events time-stamps results in a series of approximately similar integers which is exploited by the entropy coder (CABAC). Therefore, efficient compression gains can be achieved for the spatially decentralized macrocubes by projecting the spike event count in a macrocube to the *XY* plane, and delta coding the time-stamps of the successive events on each pixel.

3) SPIKE EVENT POLARITY

The spike polarity field is separately fed to the entropy coder. Luminance increases (or decreases) in a steady state [22], therefore, the temporal correlation of the polarity on a pixel

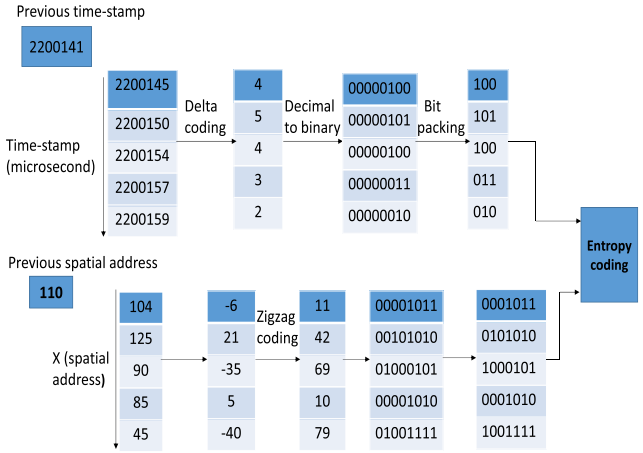

FIGURE 4. IoT specific compression to the time-stamp and the spatial address fields.

is high. In other words, if the polarity of the previous spike event is one (or zero) on a pixel, there is a high probability that the next event polarity will be one (or zero) on the same pixel. Hence, the encoding of the current spike event polarity exploits the polarity of the previous spike event, *i.e.*, previous spike polarity becomes the context of the current spike event and fed to the context adaptive binary coder (CABAC).

B. GENERAL PURPOSE, IoT AND INTEGER BASED COMPRESSION APPROACHES

1) ENTROPY CODING

Entropy coding is a lossless compression technique applicable to all types of data, hence the most fundamental building block of many compression strategies. Entropy coders are quite versatile, *i.e.,* they can compress any type of data. Therefore, most of the complex compression strategies have an entropy coder as their final compression step. In entropy coding, the length of the codeword that the encoder associates to each input symbol is directly proportional to the negative logarithm of the probability of each symbol. The most common entropy coding strategies are Huffman [23] and Arithmetic [24]. Huffman is a prefix encoder that uses variable length codewords for encoding the input data symbols. On the other hand, Arithmetic coding encodes a group of symbols into a single number.

2) DICTIONARY BASED COMPRESSION

Dictionary coding strategies operate by searching for matches between the data to be compressed and a set of strings contained in a dictionary. The dictionary is a data structure (maintained by the encoder), which is either static or dynamic, containing the set of strings. The main goal of a dictionary encoder is to find a match between the content of the dictionary and the data to be compressed. When an exact match is found, the encoder substitutes a short reference to the position of the string in the dictionary. The majority of advanced encoders use a dynamic dictionary whose content changes according to the content of the data to be compressed. For instance, LZ77 [25] holds the last N bytes of the processed data in a sliding window buffer. This sliding buffer serves as the dictionary storing every substring of N bytes of data as dictionary entries. There are many advanced level dictionary-based compression strategies, such as LZMA [26], Zstandard (Zstd) [27], Zlib [28] and Brotli [29], having sophisticated data structures. These strategies have huge dictionary sizes and use multi-level encoding to achieve high compression ratios. For instance, LZMA uses a multi-level compression where the dictionary based output is further compressed by an adaptive binary range coder. On the other hand, Zstd combines a dictionary based coding strategy with a hybrid entropy coder. The hybrid encoder achieves fast compression and comprises a combination of Finite State Entropy (FSE) [30] and Huffman encoding. Similarly Brotli combines dictionary based encoding with Huffman and second order context modelling. Another well known dictionary based encoder is Zlib which utilises the combination of LZ77 and Huffman encoding where a sliding window buffer can be adjusted to achieve a trade-off between compression ratio and speed.

3) IoT SPECIFIC COMPRESSION

The authors in [31] proposed a time-series compression algorithm, called Sprintz, for resource constrained devices. The strategy achieves state-of-the-art compression ratios with very low memory requirements (less than 1 KB memory) and virtually adds no latency. The main goal of the strategy is to exploit correlation among successive samples in multivariate time series data. The first step of the strategy is based on a forecaster, also known as Fast Integer Regression (FIRE), which encodes the difference between the actual and the predicted sample. The next step is to apply a bit packing algorithm to the prediction residuals. In the final step, Huffman coding is applied to the residuals. The strategy combining the predictive coder (FIRE) and Huffman coding is called Sprintz-FIRE. In order to achieve a trade-off between speed and compression, Sprintz has several variants. For instance, Sprintz-Delta skips the Huffman coding step and utilizes delta coding, instead of forecasting (FIRE), for a better compression speed. On the other hand, Sprintz-Delta-Huf combines delta coding and Huffman encoding to achieve better compression at the expense of lower compression speed.

Encoding steps of Sprintz-Delta-Huf on time-stamp and spatial address fields are shown in Figure 4, where delta coding and bit packing greatly reduce the size of the time-stamp field (32 bits to only 3 bits). Since spatial address differencing results in negative integers, an additional zigzag encoding step is applied to the signed residuals as shown in Figure 4.

4) FAST INTEGER COMPRESSION

Another class of potential candidates for compressing neuromorphic data are the Fast Integer compressors like SIMDBP128 [32], Simple8B [33], Memcpy, FastPFOR [32] and SNAPPY [34]. These algorithms are known for

their superlative compression speed as they are specifically designed for encoding and decoding billions of arrays of integers for search engines and relational database applications. The fastest known method of compressing integers is SIMDBP128 [32] and Memcpy, whereas FastPFOR [32] achieves better compression ratios with a little compromise on compression speed. Another well known fast integer compression algorithm is SNAPPY [34], developed by Google and used by various distributed file systems such as Hadoop, InfluxDB, KairosDB, etc.

TABLE 1. Dataset information for static DVS scenarios [35].

	Sequence	Event Rate (kev/s)	Sequence Duration (s)
Indoor	Waterdrop	3042.96	3.80
	Lighter	1329.59	2.10
	Football	1351.61	7.21
	Jump	724.09	3.28
	Game	618.42	9.57
	Pendulum	21.17	5.37
Outdoor	Pedestrians	71.66	355.21
	Daytime-Traffic1	129.70	109.84
	Daytime-Traffic2	18.32	301.60
	Night-Roadside	268.57	63.37
	Night-Traffic	350.591	15.47

III. EXPERIMENT SETUP

A. DATASETS

In order to compare the different compression strategies over a variety of scenes and sensor speeds, we consider two datasets, for fixed and moving visual sensor. The datasets are described below.

1) DATASET FOR STATIONARY DVS

To evaluate the performance of the considered strategies for static DVS, we utilized the PKU-DVS dataset for spike coding [35]. For static DVS scenarios, spike events are produced due to object movements (change of light conditions) in the scene, whereas the sensor remains stationary. The considered dataset is divided into indoor and outdoor scenarios. These scenarios are captured under diverse circumstances; for instance the outdoor sequences include daytime, night, slow speed movement (pedestrian scenarios) and high speed movement (vehicular speed movements). Similarly the indoor scenarios include diverse types of object movements ranging from pendulum motion to a jumping movement of a person. The spike event rate, in Kilo-event per second (Kev/s), along with the sequence duration of all the indoor and outdoor scenarios is reported in Table 1.

2) DATASET FOR MOBILE DVS

In order to assess the compression performance on mobile DVS data, we consider the Dynamic and Active-pixel Vision Sensor (DAVIS) dataset [36]. DAVIS incorporates a conventional global-shutter camera and an event-based sensor in the same pixel array. The compression performance of the considered strategies is evaluated on the asynchronous spike event stream. The DAVIS dataset is specifically designed for high-speed and high-dynamic-range robotics and computer-vision applications for indoor and outdoor scenarios. In addition to the visual output (spike event stream and intensity images), the dataset also includes the speed of the sensor. The indoor and outdoor scenarios are captured under widely varied motion (angular and linear motion of the sensor) and dynamic range. The inclusion of intensity images also makes it possible to compute scene complexity [4] of different outdoor and indoor scenes. Figure 5 shows some of the considered indoor and outdoor scenes [36]. The extracted sequences considered for compression performance evaluation, with different content complexity and sensor speed, are shown in Table2. The higher the scene complexity and sensor speed, the higher the event rate [4]. For instance, the *Boxes* sequence exhibits an event rate of 4288.65 Kev/s, whereas the bandwidth of the *Shapes* sequence is only 245.61 Kev/s owing to the low content complexity and slow speed of the sensor. The scene complexity and motion speed information of the dataset is reported in [4].

FIGURE 5. Shapes, Boxes, Poster and Outdoor scenes of the the DAVIS dataset [36].

TABLE 2. Dataset information for mobile DVS scenarios, extracted from [36].

	Sequence	Event Rate (kev/s)	Sequence Duration (s)	Scene Complexity	Speed
Indoor	Boxes	4288.65	5 (45-50)	High	High
	Poster	4021.1	5 (45-50)	High	High
	Dynamic	1077.73	20 (1-20)	Medium	Medium
	Slider	336.78	3 (1-3)	Medium	Low
	Shapes	245.61	20 (1-20)	Low	Low
Outdoor	Running3	1525.5	20 (40-60)	Medium	High
	Running2	1229.4	20 (20-40)	Medium	Medium
	Running1	713.8	20 (1-20)	Medium	Medium
	Urban	503.04	10 (1-10)	High	Low
	Walking	342.2	20 (1-20)	Medium	Low

B. MULTIVARIATE REPRESENTATION OF THE AER (AER: DATA OUTPUT FORMAT OF THE DVS CAMERA)

Figure 6 shows the recorded data format of spike events in terms of AER: a series of 8-byte words, where time-stamp

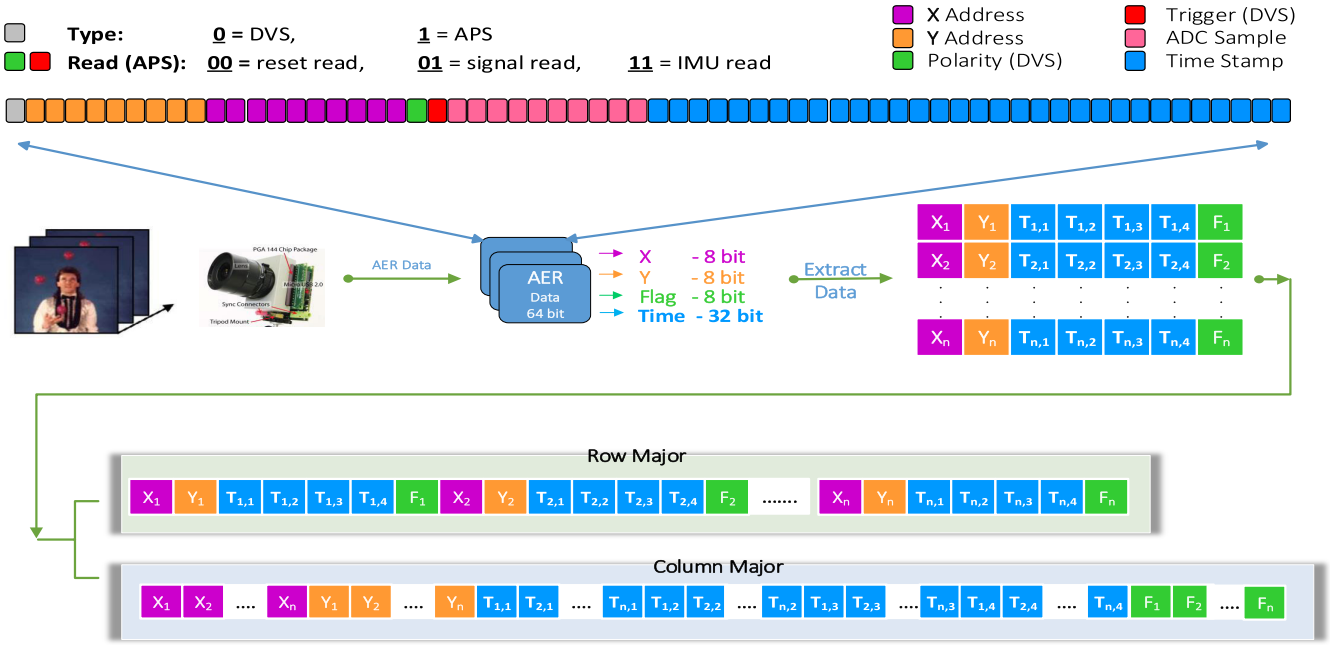

FIGURE 6. Simulation Setup, including in the top part the AER data format.

information is represented by the first 32 bits. One bit is reserved for the neuromorphic vision sensor type (DVS or Asynchronous Time-based Image Sensor (ATIS)). The spatial address information in terms of X and Y is represented by 10 and 9 bits respectively.

The ATIS has built in acceleration, temperature and gyroscope sensors, therefore 10 bits are reserved for Analog to Digital Converter (ADC) samples of these sensors, as shown in Figure 6. One bit (44^{th} bit) represents the DVS polarity field, whereas DVS trigger information is also represented by one bit (43^{rd} bit). There are several ways to compress the AER data, the most common approach consisting of compressing the data as a single variable vector, *i.e.,* as a single stream of 8 byte integers. According to our study [12], the most effective approach is to convert the data into a multivariate stream by extracting the relevant information (time-stamps, spatial addresses and the polarity bit) from the 64-bit AER data. This approach is shown in Figure 6, where all the four relevant fields are extracted from the AER data format. It is important to note that we divide the 32-bit time-stamp field into four 8-bit integers. The multivariate stream now consists of seven columns (each 1 byte long) where time-stamp information consists of four columns, whereas the spatial addresses and the polarity flag fields comprise three columns. The multivariate data stream is then converted into row-major and column-major formats. In row-major order, consecutive elements of the rows of the array are contiguous in memory, whereas in column-major order the consecutive elements of the columns are contiguous, as shown in Figure 6. The row major format is primarily used for the IoT specific compression strategies, whereas Entropy, Dictionary and Fast integer compression strategies utilize the column major format.

In order to compress the DVS data by utilizing the spike coding strategy, the spatial resolution of the macrocube is the same as that of the DAVIS, *i.e.,* $X_{max} = 240$, and $Y_{max} = 180$. Furthermore, the third dimension (number of spikes) of the macrocube is set to 32768, for optimum compression performance recommended by [37].

C. KEY PERFORMANCE INDICATORS

1) COMPRESSION RATIO

The performance of the considered and benchmark strategies is evaluated by computing the compression ratio (CR):

$$\mathrm{CR} = \frac{N_{events} \times 8}{\gamma} \tag{1}$$

where γ is the size (in bytes) of the compressed output stream and N_{events} is the total number of spike events, with each event equal to 8 bytes.

2) COMPRESSION AND DECOMPRESSION SPEED

In order to evaluate data compression algorithms, speed (compression and decompression) is measured in terms of uncompressed data handled per second. The compression speed (CS) and decompression speed (DS) is evaluated as:

$$\mathrm{CS} = \frac{N_{events} \times 8}{T_c} \; [bytes/s] \tag{2}$$

and

$$\mathrm{DS} = \frac{N_{events} \times 8}{T_d} \; [bytes/s] \tag{3}$$

where T_c and T_d are the time (seconds) required to compress and decompress the spike events data respectively. All experiments use a single thread on a 2015 Macbook Pro with a 2.2GHz Quad-Core Intel Core i7 processor and 16 GB, 1600 MHz DDR3 RAM.

3) COMPRESSION AND DECOMPRESSION DELAY FACTOR

We define the compression (and decompression) delay factor as the ratio between the bit rate of the source and the compression (and decompression) speed. This is evaluated as reported below:

$$\Lambda_{\mathrm{c}} = \frac{\left(\frac{N_{events} \times 8}{T_s}\right)}{\mathrm{CS}} = \frac{T_c}{T_s} \tag{4}$$

and

$$\Lambda_{\mathrm{d}} = \frac{\left(\frac{N_{events} \times 8}{T_s}\right)}{\mathrm{DS}} = \frac{T_d}{T_s} \tag{5}$$

where T_s is the time duration (in seconds) of the DVS sequence.

In order to analyse the latency performance of each of the considered algorithms, first we compute the bitrate of a single DVS sequence followed by the computation of the average bitrate of all the video sequences. The compression delay factor, for instance, of an algorithm is computed as the ratio between the average bit rate of all the sequences and the average compression speed. This is equivalent to the ratio between the time required to compress the DVS data sequence and the time duration of the DVS sequence.

In the following, the compression performance analysis of the static and mobile DVS data is discussed in Section IV and V respectively.

IV. COMPRESSION PERFORMANCE ON DVS DATA GENERATED BY STATIC SENSOR FOR OUTDOOR AND INDOOR SCENARIOS

In this section we first analyse the different spike event fields of the DVS data for static scenarios, followed by a detailed analysis of the compression performance of all the considered strategies.

A. HISTOGRAM ANALYSIS OF DIFFERENT FIELDS OF THE DVS DATA UNDER STATIC SCENARIOS

In order to analyse the compression performance of all the considered strategies on the DVS data for the static scenario, we computed the histogram of different sequences (*Night-traffic*, *Pendulum* and *Daytime-traffic2*), as shown in Figure 7. The figure shows the histogram of the spatial address fields (X and Y) in the first two columns and of the the delta-coded time-stamp field in the third column. The histogram of the spatial address highlights the locations receiving spike events (change of luminance intensity). For static sensor scenarios, objects' movements usually lead to spike events firing over adjacent pixels (neighbouring spatial address). In other words, the spatial address integer field in the spike event stream (due to objects movements) is highly correlated. For instance if we analyse the histogram of the *Night-traffic* sequence (first row of Figure 7), the majority of the the spike events are elicited over a specific range of spatial addresses (X = 180 to 220 and Y = 130 to 170). Similarly, for the other two sequences spike events are fired at a specific range of pixels. For instance, the motion of pendulum elicits luminance variations at a handful of pixels location, as shown by the histogram bins of the *Pendulum* sequence (second row of Figure 7). On the other hand, the delta coding of the time-stamp field reduces the range of the 32 bit integer down to fewer bits. For instance, in the time-stamp histogram of the *Night-traffic* sequence (top right in Figure 7) there are only two bins. Therefore, only one bit (0 or 1) is required to represent the delta coded time-stamp field of the *Night-traffic* sequence. Similarly the number of bins for the *Pendulum* and *Daytime-traffic2* sequences is less than 200. These sequences require only 8 bits to represent the delta coded time-stamp field ($2^8 = 256 > 200$).

The histogram analysis of the static DVS data reveals a highly correlated spike event sequence, *i.e.,* the stream of multivariate integers exhibits a high degree of correlation.

B. COMPRESSION PERFORMANCE ANALYSIS

In the following, we analyse the compression performance of the considered strategies. As a summary, Table 3 reports the compression ratios of all the considered strategies, whereas the compression and decompression speed, in Mega-byte per second (MB/s), is reported in Table 4. The best strategy for each scene is highlighted in bold. According to Table 3, spike coding and dictionary based strategies show impressive lossless compression ratios.

Details on the compression performance of each strategy are reported below:

1) COMPRESSION PERFORMANCE OF DICTIONARY BASED STRATEGIES

Dictionary based compression strategies yield excellent compression performance, as reported in Table 3. The histogram analysis of the static DVS data shows a highly correlated multivariate stream of integers. The main design goal of dictionary based strategies is to exploit redundancy in the data. The higher the correlation in the data, the higher the compression performance of these strategies.

LZMA shows the best performance in term of compression ratio among the considered strategies. LZMA utilizes a huge dictionary size as compared to LZ4 and has a range encoder as the final compression step. The range encoder utilizes a complex mathematical model to make probability predictions on a bit basis (as compared to byte predictions models in other dictionary based strategies). Therefore, encoding data on a bit by bit bases allows LZMA to efficiently exploit correlation between consecutive spike events. The encoding of single bit by the range encoder allows many encoding possibilities, therefore, LZMA uses Dynamic Programming

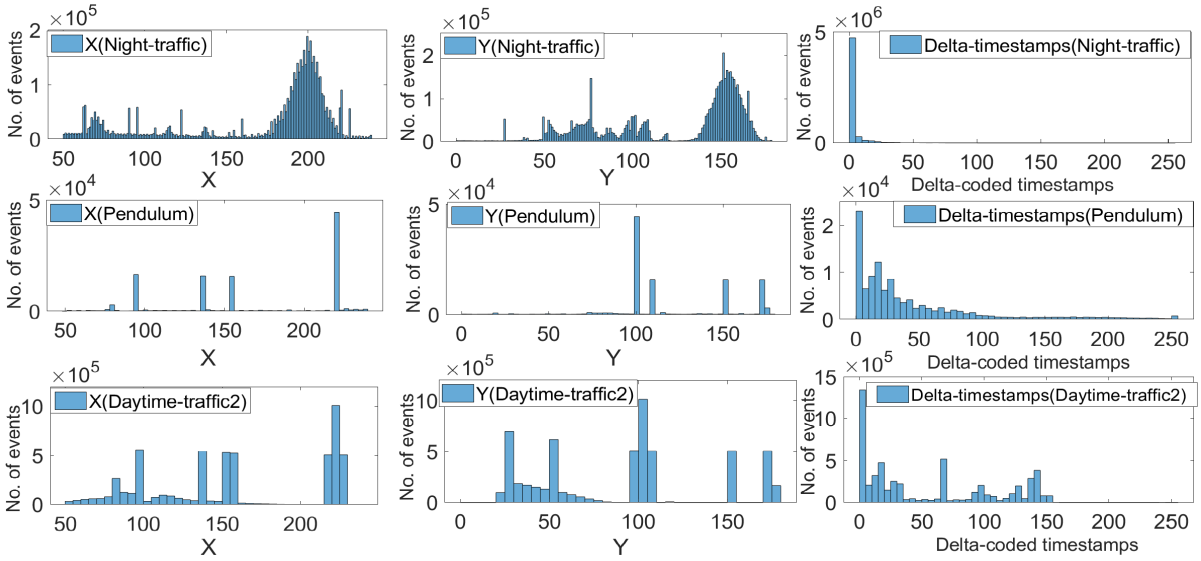

FIGURE 7. Histogram of different fields of the spike events data. First row: Night-traffic, Second row: Pendulum and Third row: Daytime-traffic2.

TABLE 3. Compression ratio comparison of the considered strategies on the static PKU-DVS dataset.

	Sequence	Spike Coding	LZMA	Brotli	Zlib	Zstd	LZ4	Sprintz Delta-Huf	Sprintz FIRE	Sprintz Delta	Huffman	Snappy	FastPFOR	SIMDBP128	Simple8B	Memcpy
Indoor	Waterdrop	59.59	**82.28**	71.07	67.61	65.76	30.74	24.35	22.91	5.32	3.20	14.63	1.39	1.38	1.24	1.12
	Lighter	20.39	**22.19**	17.98	17.47	16.60	9.73	4.29	3.28	2.40	2.65	7.42	1.38	1.38	1.23	1.12
	Football	**18.82**	18.70	15.46	14.47	14.85	8.84	6.41	5.04	3.18	2.88	6.81	1.37	1.37	1.23	1.12
	Jump	**8.97**	8.84	7.51	7.21	7.19	4.66	3.38	2.90	2.33	2.38	4.32	1.37	1.37	1.22	1.12
	Game	4.50	**5.45**	4.74	4.66	4.50	3.32	3.59	3.32	2.53	2.01	3.21	1.40	1.36	1.24	1.12
	Pendulum	**4.40**	4.25	3.69	3.68	3.49	2.68	2.37	2.11	1.80	1.87	2.54	1.34	1.34	1.20	1.12
Outdoor	Pedestrians	14.25	**16.53**	13.55	13.21	12.74	8.54	5.42	4.15	2.76	2.64	6.23	1.18	1.18	1.06	1.12
	Daytime-Traffic1	11.13	**12.32**	10.10	10.04	9.62	6.06	3.52	2.85	2.20	2.46	5.36	1.19	1.19	1.07	1.12
	Daytime-Traffic2	**6.49**	5.75	4.64	4.54	4.46	3.01	2.45	2.17	1.82	1.91	2.98	1.17	1.17	1.05	1.12
	Night-Roadside	5.44	**6.80**	5.82	5.64	5.57	3.79	3.36	3.01	2.36	2.14	3.72	1.24	1.22	1.11	1.12
	Night-Traffic	3.87	**4.92**	4.31	4.18	4.08	2.93	3.77	3.65	2.66	1.93	2.90	1.42	1.35	1.26	1.12
	Total Average	**14.35**	**17.09**	**14.44**	**13.88**	**13.53**	**7.66**	**5.72**	**5.04**	**2.67**	**2.37**	**5.47**	**1.31**	**1.30**	**1.17**	**1.12**

algorithm to select an optimal encoding mode. The utilization of range encoder and dynamic programming for encoding mode selection is computationally expensive, therefore the compression and decompression speed for LZMA is the slowest among all the considered dictionary based strategies. Zstd also performs well in terms of compression ratios as it also uses an entropy coder as the final compression step. However, instead of using the computationally expensive binary range encoder, it utilizes a combination of the finite state entropy and Huffman encoding strategy. This results in fast compression and decompression speed with a slight reduction in compression ratio. The use of Finite State Entropy as the final encoding step achieves very high decompression speed for Zstd as shown in Figure 4. Brotli achieves the best trade-off between compression ratio and speed (compression and decompression) as shown in Table 3 and Table 4. The better compression and decompression speed for Brotli is mainly due to the utilization of Huffman encoding as the final compression step. Furthermore, second order context modelling achieves superior compression ratios for Brotli as compared to Zstd and Zlib.

2) SPIKE CODING COMPRESSION PERFORMANCE

The spike coding strategy encodes spike events by projecting them over a series of macrocubes. A limited number of spikes are projected within each macrocube. In this study, we consider 32678 number of spike events per macrocube as recommended by [37]. In order to fix the size of the entropy coder, the spike coding strategy limits the number of spike events per macrocube. The histogram analysis of static DVS data shows the firing of spike events among a group of adjacent pixels. For such scenarios, spike coding utilizes the TP mode, where spike events are projected based on their firing time as shown in Figure 3. The TP mode exploits spatial redundancy with in each macrocube. In dictionary based strategies, correlation among millions of consecutive spike events (multivariate stream of integers) is exploited because of the huge dictionary size. On the other hand, the spike coding exploits spatial redundancy among 32678 spike events, *i.e.*, with in each macrocube. Therefore, dictionary based compression strategies such as Brotli and LZMA outperform the spike coding strategy. The cube-based spike coding mechanism achieves good compression ratios

TABLE 4. Static sensors compression and decompression speed [MB/s] comparison.

	Sequence	Spike Coding	LZ4	Zstd	Brotli	Zlib	LZMA	Sprintz Delta	Sprintz Delta-Huf	Sprintz FIRE	Huffman	Memcpy	SIMDBP128	FastPFOR	Simple8B	Snappy
Indoor	Waterdrop	43.40	4155.13	1262.44	830.34	262.51	86.80	404.82	367.98	292.84	1387.83	**12598.39**	9454.41	1830.35	2265.70	3024.70
	Lighter	17.36	1920.82	609.08	459.04	149.08	34.72	410.63	317.71	253.42	1190.02	**10839.13**	9118.57	2511.07	2110.19	1404.30
	Football	14.94	1475.34	513.50	378.88	128.14	29.88	428.60	344.59	251.63	1256.63	**11989.02**	9296.95	2193.03	2141.36	1188.01
	Jump	9.63	1112.53	390.05	264.64	83.86	19.25	456.53	346.77	265.06	1299.85	**13498.58**	10210.34	2603.61	2102.18	933.49
	Game	6.13	1082.56	351.09	218.04	59.62	12.26	450.69	354.07	252.16	1265.59	**14196.59**	10123.36	2367.31	2058.03	903.19
	Pendulum	5.28	857.72	292.24	164.12	48.80	10.55	306.73	233.08	213.60	1196.92	11343.83	**12643.18**	2560.07	1920.45	704.29
Outdoor	Pedestrians	14.34	2281.10	656.12	508.21	124.31	28.68	422.82	352.39	263.67	1375.60	**12828.17**	9523.91	2376.74	1790.39	1449.13
	Daytime-Traffic1	11.87	1463.52	467.80	336.02	97.86	23.73	427.74	323.43	244.55	1310.09	**13626.17**	9731.51	2359.53	1818.22	1114.58
	Daytime-Traffic2	5.50	886.30	294.40	165.77	54.67	11.00	409.18	311.64	224.11	1252.53	**12158.18**	9920.32	2614.51	1612.74	751.85
	Night-Roadside	7.32	1022.34	359.82	218.94	71.28	14.64	459.25	354.30	254.55	1281.39	**13906.93**	10025.77	2428.76	1834.05	829.61
	Night-Traffic	5.23	961.01	329.86	175.83	55.74	10.46	440.38	329.64	227.93	1320.83	**13218.38**	9597.34	2108.26	1985.87	802.24
Average Compression speed		**12.82**	**1565.31**	**502.40**	**338.17**	**103.26**	**25.63**	**419.76**	**330.51**	**249.41**	**1285.21**	**12745.76**	**9967.79**	**2359.39**	**1967.20**	**1191.40**
Indoor	Waterdrop	389.27	4271.75	3101.06	171.03	884.48	778.54	1485.09	1060.81	867.80	2278.89	**12003.80**	10751.29	8718.68	2949.99	3109.71
	Lighter	140.89	2825.23	1601.07	689.89	694.96	281.78	1374.61	757.46	616.23	1730.72	10951.34	10170.15	**11056.96**	2713.58	2288.02
	Football	130.87	2646.72	1492.22	756.09	601.59	261.74	1480.42	875.12	701.19	1730.78	**12155.79**	11994.67	11732.09	2845.22	1987.79
	Jump	75.71	2459.24	1275.96	599.74	495.23	151.42	1525.09	768.90	688.72	1604.15	**12022.92**	10789.04	11508.99	3012.76	1907.66
	Game	47.82	2240.32	965.15	487.32	393.17	95.64	1427.66	837.60	690.84	1460.77	**13136.61**	10644.95	10381.20	2969.51	1679.37
	Pendulum	34.78	533.58	702.98	208.85	272.06	69.55	414.64	272.98	423.36	1321.32	**5280.15**	1856.54	3483.92	1580.67	1111.35
Outdoor	Pedestrians	130.56	3645.15	1968.43	863.18	661.56	261.12	1498.81	868.73	729.33	1969.85	**13071.37**	11967.33	11269.57	2565.37	2349.91
	Daytime-Traffic1	94.82	2716.77	1523.84	681.48	554.76	189.63	1425.02	756.93	634.73	1893.70	**12367.86**	10188.84	9927.30	2277.80	2073.96
	Daytime-Traffic2	49.61	2111.75	952.56	405.50	361.80	99.22	1487.46	681.98	632.43	1492.49	**12644.63**	10691.23	10581.86	2459.83	1877.82
	Night-Roadside	61.25	2454.75	1186.16	545.99	451.69	122.49	1619.22	838.64	706.98	1665.61	**13650.18**	12633.30	11824.77	2844.39	1773.65
	Night-Traffic	45.00	2337.06	984.51	403.97	338.86	90.00	1517.69	812.86	683.28	1690.66	**12393.57**	12269.23	11463.54	2576.02	1693.62
Average Decompression speed		**109.14**	**2567.48**	**1432.18**	**528.46**	**519.11**	**218.28**	**1386.88**	**775.64**	**670.44**	**1712.63**	**11788.93**	**10359.69**	**10177.17**	**2617.74**	**1986.62**

by exploiting the spatial redundancies in the DVS data. The compression ratios achieved for the datasets *Jump*, *Game*, *Pendulum* and *Daytime-traffic2* are the best among all the considered strategies. It is important to note that the spatial and temporal prediction residuals are further compressed by CABAC. However, CABAC is computationally expensive, therefore the compression and decompression speed achieved by spike coding is the lowest among all the considered strategies, as reported in Table 4. Moreover, the process of trying both the modes, AP and TP, and the selection of the best mode (yielding maximum compression) further reduces the compression speed.

3) IoT SPECIFIC COMPRESSION PERFORMANCE

Table 3 shows that IoT specific compression strategies yield low compression ratios. These strategies consider only 8 samples per block, i.e., 8 consecutive samples for each spike's field. These strategies are mostly designed for sensor generated time-series data exhibiting slow time-varying characteristics, *i.e.* when consecutive samples in sensor generated data have high correlation. However, the correlation among the 8 spatial address integers can be low when the spike events are fired at 8 different spatial locations. Sprintz-Delta-Huff shows the best compression ratio among the IoT specific compression strategies, which shows that delta coding yields better compression than Sprintz forecasting algorithm FIRE. This is mainly because it is difficult to forecast the firing location of the spike events.

The time-stamp field of the spike events is the most compressible field since it is an incremental integer. Delta coding followed by bitpacking greatly reduces redundancy, as shown in Figure 4, where the 32 bits time-stamp field is reduced to 3 bits. It is important to note that Huffman coding as the final compression step improves the compression ratio of the Sprintz strategy as shown in Table 3, where Sprintz-Delta-Huf outperforms Sprintz-Delta at the expense of a higher compression speed. The main advantage of the Sprintz strategy is its low memory consumption (less than 1 kB), which makes it suitable for resource constrained devices.

4) HUFFMAN CODING COMPRESSION PERFORMANCE

Huffman encoding as the only compression strategy achieves low compression ratio. However, the combination of Huffman coding with the Sprintz and Dictionary based strategies yields promising results. For instance, the compression ratio is more than doubled (from 2.67 to 5.72) by employing Huffman encoding to further compress the residuals of Sprintz-Delta encoding. Similarly the performance of the classic dictionary based strategy (LZ4, yielding compression ratio of 7.63) is enhanced by using Huffman encoding as the final compression step in the Zlib strategy, which increases the compression ratio to 13.53 as shown in Table 3. However, the improved compression ratio comes at the cost of decrease in the compression and decompression speed, as shown in Table 4.

5) FAST INTEGER COMPRESSION PERFORMANCE

Fast compression strategies such as Memcpy, Simple8B, FastPFOR and SIMDBP128 hardly compress the DVS data. These strategies do not aim at maximum compression, but rather at high compression and decompression speed. For instance, Memcpy and SIMDBP128 achieve compression speed of 12745.76 and 9967.79 MB/s respectively, as reported in Table 4. Snappy achieves the best compression ratio among the fast integer based compression approaches.

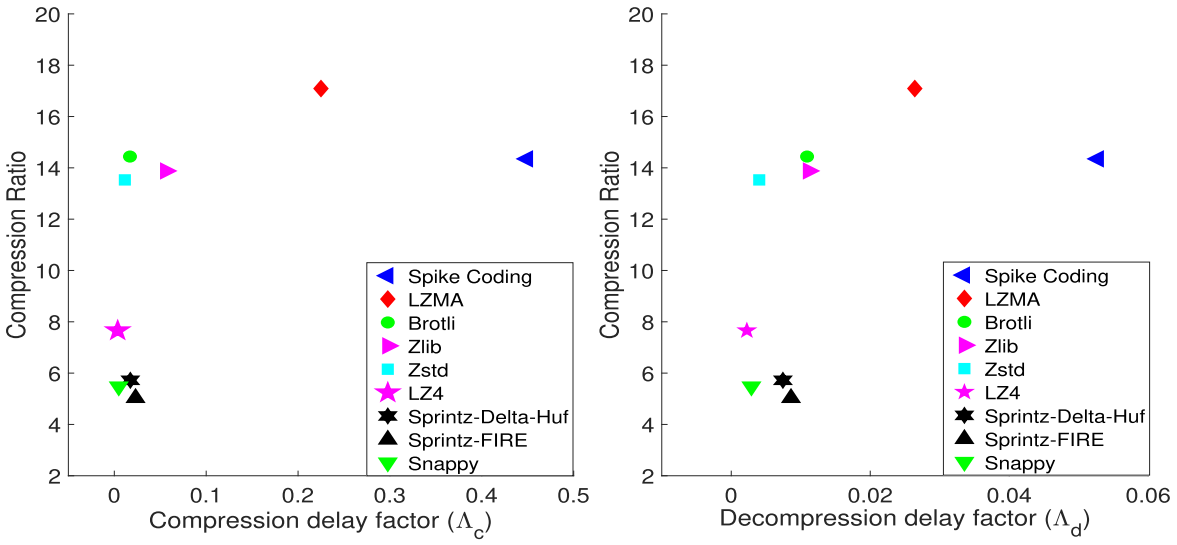

FIGURE 8. Average compression and decompression delay factor (time needed to compress and decompress 1s of data) for static sensor scenario.

C. AVERAGE COMPRESSION AND DECOMPRESSION DELAY FACTOR ANALYSIS FOR STATIC DVS

Figure 8 reports the average compression and decompression delay factor *w.r.t* compression and decompression ratios respectively. The average compression delay factor is a function of input data rate and compression speed. The higher the compression speed and lower the data rate, the lower the delay factor. Spike coding and LZMA exhibit a higher compression delay factor of approximately 0.45 and 0.25 respectively. This implies that spike coding and LZMA need 0.45 and 0.25 seconds respectively to compress one second of data on average. The higher latency performance of both the strategies is mainly because of the utilization of computationally expensive binary arithmetic encoding. Furthermore, the process of finding the best compression mode (TP or AP) for each macrocube increases the average compression delay factor (to approximately 0.45) for the spike coding strategy. On the other hand, LZ4 achieves the lowest compression delay factor of less than 0.005 (less than 5 ms of latency in compressing one second of data) because this strategy skips the entropy coding step, unlike other dictionary based strategies. Snappy also shows impressive compression delay factor performance because the main design goal of the strategy is the high compression speed with a reasonable compression ratio. IoT specific Sprintz strategies also incur low compression latency (less than 20 ms) owing to the encoding of only 8 spike events per block. Dictionary based strategies such as Brotli, Zstd and Zlib are also suitable for real-time compression mainly because of the delay factor of less than 0.06, *i.e*, less than than 60 ms of compression latency is required by these strategies.

All the considered strategies show impressive decompression delay factor (less than 60 ms needed to decompress one second of data on average) performance as shown in Figure 8. This is mainly because of the higher decompression speed of all the considered strategies.

In the following, we analyse the compression and decompression performance of the different strategies on data acquired by a DVS sensor in motion.

V. COMPRESSION PERFORMANCE ON DVS DATA GENERATED BY MOBILE SENSOR FOR OUTDOOR AND INDOOR SCENARIOS

The compression ratios of mobile DVS sensor data for the outdoor and indoor scenarios are shown in Table 5, whereas the compression and decompression speed performance is shown in Table 6. For mobile sensor scenarios, spike coding achieves the best compression performance followed by the dictionary based compression strategies.

A summary of the key observations from the results is reported subsequently.

A. LOW COMPRESSION RATIO PERFORMANCE FOR MOBILE DVS DATA

The compression gains achieved by different strategies are lower compared to the scenarios of the static sensor. Table 5 also reports the percentage decrease in compression gains *w.r.t* static sensor. For instance, the gains achieved by the spike coding strategy decrease by 72.2 % (14.35 to 3.99), whereas the compression performance of LZMA decreases to 3.57 (79.14 % decrease). There is also a sharp decline in compression performance of other dictionary based strategies, for instance the performance of Zstd, Zlib, Brotli and LZ4 decreases by 76.66 %, 76.63 %, 77.18 % and 67.40 % respectively. The decrease in the compression performance of different Sprintz strategies is lower as compared to the dictionary based strategies. For instance Sprintz-Delta-Huff, Sprintz-Fire and Sprintz-Delta show a decline of 50.21 %, 44.38 % and 9.40 % respectively. The compression performance of the fast integer compression strategies remains approximately the same.

TABLE 5. Compression ratio comparison of the considered strategies on the mobile DAVIS dataset.

	Sequence	Spike Coding	LZMA	Brotli	Zlib	Zstd	LZ4	Sprintz Delta-Huf	Sprintz FIRE	Sprintz Delta	Huffman	Snappy	FastPFOR	SIMDBP128	Simple8B	Memcpy
Indoor	Boxes	**4.95**	4.92	4.38	4.21	4.13	3.03	3.83	3.72	2.83	1.96	2.98	1.4	1.38	1.25	1.12
	Dynamic	**3.85**	3.34	3.19	3.13	3.07	2.46	2.68	2.63	2.33	1.89	2.44	1.31	1.28	1.15	1.12
	Poster	**4.88**	4.77	4.26	4.12	4.02	2.97	3.7	3.6	2.76	1.96	2.92	1.4	1.37	1.24	1.12
	Slider	**3.84**	3.19	2.85	2.89	2.76	2.3	2.65	2.62	2.36	1.79	2.26	1.41	1.36	1.23	1.12
	Shapes	**3.78**	3.04	2.78	2.8	2.67	2.19	2.46	2.43	2.26	1.79	2.21	1.34	1.31	1.17	1.12
Outdoor	Running 1	**3.68**	3.25	3.09	3.05	2.96	2.39	2.6	2.58	2.33	1.87	2.37	1.35	1.32	1.18	1.12
	Running 2	**3.92**	3.41	3.26	3.19	3.13	2.51	2.7	2.67	2.35	1.89	2.48	1.3	1.27	1.15	1.12
	Running 3	**3.97**	3.49	3.32	3.26	3.21	2.56	2.75	2.72	2.36	1.9	2.53	1.29	1.26	1.14	1.12
	Urban	**3.45**	3.13	2.93	2.91	2.83	2.32	2.58	2.54	2.31	1.83	2.31	1.36	1.35	1.22	1.12
	Walking	**3.54**	3.11	2.89	2.88	2.8	2.24	2.53	2.52	2.3	1.84	2.24	1.35	1.31	1.18	1.12
Total Average		**3.99**	**3.57**	**3.30**	**3.24**	**3.16**	**2.50**	**2.85**	**2.80**	**2.42**	**1.87**	**2.47**	**1.35**	**1.32**	**1.19**	**1.12**
% Decrease w.r.t Static		**72.22**	**79.14**	**77.18**	**76.63**	**76.66**	**67.40**	**50.21**	**44.38**	**9.40**	**21.01**	**54.77**	**-3.13**	**-1.62**	**-1.79**	**0.00**

TABLE 6. Mobile sensors compression and Decompression speed [MB/s] comparison.

	Sequence	Spike Coding	LZ4	Zstd	Brotli	Zlib	LZMA	Sprintz Delta	Sprintz Delta-Huf	Sprintz FIRE	Huffman	Memcpy	SIMDBP128	FastPFOR	Simple8B	Snappy
Indoor	Boxes	9.45	903	298.02	196.4	48.34	9.87	471.08	352.22	241.86	1219.36	**8990.29**	8423.1	1444.03	2548.75	706.52
	Dynamic	8.21	1182.4	330.07	207.1	39.33	8.35	437.07	318.02	233.9	1024.64	**9012.52**	8407.45	1494.12	2316.07	921.86
	Poster	9.38	924.41	294.33	187.5	47.66	9.72	466.7	348.09	235.12	1159.37	**9196.34**	8672.56	1448.96	2535.98	715.51
	Slider	9.25	1206.2	378.07	219.7	37.59	9.9	404.23	308.08	214.3	955.02	9713.75	**10677.14**	1554.21	2330.08	1148.5
	Shapes	7.92	1709	441.87	250.7	36.94	8.23	429.44	331.11	233.35	1076.27	9121.42	**9447.7**	1460.57	2252.63	1482.3
Outdoor	Running 1	8.05	1252.5	345.92	196.1	37.51	8.21	469.25	332.65	231.43	1058.51	**8049.93**	7814.26	1507.46	2219.14	962.18
	Running 2	8.25	1188.7	334.71	213.1	40.34	8.52	473.13	343.57	235.5	1029.74	**9083.12**	7946.89	1514.55	2128.07	942.67
	Running 3	8.38	1195.2	307.08	216	40.56	8.37	470.27	340.16	235.35	1023.53	**8541.42**	8286.39	1449.09	2246.89	934.03
	Urban	7.87	1333.5	348.58	208.6	38.15	8.5	441.08	324.55	234.33	1012.84	7849.92	**8604.35**	1695.93	2314.57	1063.7
	Walking	7.75	1516.9	399.41	217	36.61	8.36	424.04	313.21	226.2	1015.22	6954.21	**8518.76**	1588.84	2313.59	1441.5
Average Compression speed		**8.45**	**1241.18**	**347.81**	**211.22**	**40.30**	**8.80**	**448.63**	**331.17**	**232.13**	**1057.45**	**8651.29**	**8679.86**	**1515.78**	**2320.58**	**1031.88**
% Decrease w.r.t Static		**34.09**	**20.71**	**30.77**	**37.54**	**60.97**	**65.67**	**-6.88**	**-0.20**	**6.93**	**17.72**	**32.12**	**12.92**	**35.76**	**-17.96**	**13.39**
Indoor	Boxes	74.25	1880.07	1013.86	436.02	374.41	77.30	1471.65	771.71	634.79	1748.35	**7502.87**	3278.72	6343.22	6115.13	1366.07
	Dynamic	52.25	2313.92	1018.19	359.20	334.22	54.70	1519.74	806.63	655.48	1668.19	**9087.91**	6572.97	6720.85	6228.00	1650.52
	Poster	73.11	1894.10	1020.29	407.22	365.94	75.12	1472.85	771.50	625.15	1678.48	**9464.12**	7003.73	6360.19	6523.91	1398.62
	Slider	49.47	1044.28	688.03	261.84	274.94	50.69	1085.69	635.98	612.67	1223.51	**5839.27**	3467.50	4121.36	5180.18	1070.93
	Shapes	47.33	2075.39	1023.76	333.62	324.40	49.83	1258.35	783.06	616.03	1689.79	**7884.98**	6421.31	6148.89	6261.32	2250.10
Outdoor	Running 1	48.65	2249.61	996.18	348.71	324.79	53.65	1503.93	584.76	629.42	1614.11	**7844.53**	6599.31	6503.42	5642.34	1690.38
	Running 2	54.28	2498.51	1053.36	360.96	339.33	56.19	1556.53	790.14	662.85	1559.41	**9275.16**	7017.97	6878.13	6467.32	1748.66
	Running 3	55.18	2330.73	977.77	373.24	343.79	56.54	1541.85	756.15	662.38	1531.47	**8790.65**	6859.40	6320.75	5862.20	1762.01
	Urban	47.89	2072.53	929.82	324.16	315.42	51.08	1263.24	710.22	630.35	1544.63	**7829.54**	6122.97	6631.65	5456.52	1648.09
	Walking	47.97	1634.49	867.84	329.68	305.22	51.37	1333.79	757.63	615.96	1419.66	**7530.89**	6101.14	6050.99	5506.71	2190.58
Average Decompression speed		**55.04**	**1999.36**	**958.91**	**353.47**	**330.25**	**57.65**	**1400.76**	**736.78**	**634.51**	**1567.76**	**8104.99**	**5944.50**	**6207.95**	**5924.36**	**1677.60**
% Decrease w.r.t Static		**49.57**	**22.13**	**33.05**	**33.11**	**36.38**	**73.59**	**-1.00**	**5.01**	**5.36**	**8.46**	**31.25**	**42.62**	**39.00**	**-126.32**	**15.55**

B. THE COMPRESSION RATIO INCREASES WITH THE EVENT RATE

Another important observation is that higher event rate sequences result in better compression performance, *i.e.*, all the considered strategies compress higher event rate sequences more than low event rate sequences. For instance, among the considered sequences, *Poster* and *Boxes* yield the highest compression gains. On the other hand, the compression performance of the *Shapes* sequence is the lowest.

In the following, we find the reasoning behind the aforementioned observations of higher event rate sequences yielding better compression, and the low compression ratios for the mobile DVS data.

1) HISTOGRAM AND ENTROPY COMPUTATION

Figure 9 shows the frequency distribution (histogram) of the X, Y, and delta coded time-stamp fields of the *Poster*, high rate sequence, and *Shapes*, low rate sequence. Furthermore, the figure also reports the entropy of these fields.

The entropy is evaluated as:

$$H(X) = \sum_{x=1,\ldots,X_{max}} p(x) \log_2 \frac{1}{p(x)} \tag{6}$$

$$H(Y) = \sum_{y=1,\ldots,Y_{max}} p(y) \log_2 \frac{1}{p(y)}. \tag{7}$$

The corresponding source with the same alphabet and uniform probability distribution, i.e., with equiprobable symbols (incompressible source in the memoryless case if we do not accept losses) would result in the maximum entropy:

$$H_{\max}(X) = \log_2 X_{max} \tag{8}$$

$$H_{\max}(Y) = \log_2 Y_{max}. \tag{9}$$

The maximum entropy of spatial address X is hence $\log_2 240 = 7.9069$, and for Y it is $\log_2 180 = 7.4919$.

The entropy of both the spatial address fields is close to that of the uncompressible source for both the sequences, as shown in Figure 9, where the entropy of the spatial addresses of both the sequences is reported. In other words, the spatial address fields exhibit a stream of nearly equally distributed random integers for mobile DVS scenarios. It is important to note that both, high and low complexity sequences, show similar entropy for the spatial address field. The other considered outdoor and indoor scenes also exhibit approximately the same level of entropy for the spatial address fields (not shown here for brevity). The proximity

to an equiprobable distribution in the spatial address fields of the spike events (produced by the moving DVS) is the main reason for the low lossless compression performance as analysed in Section V-A.

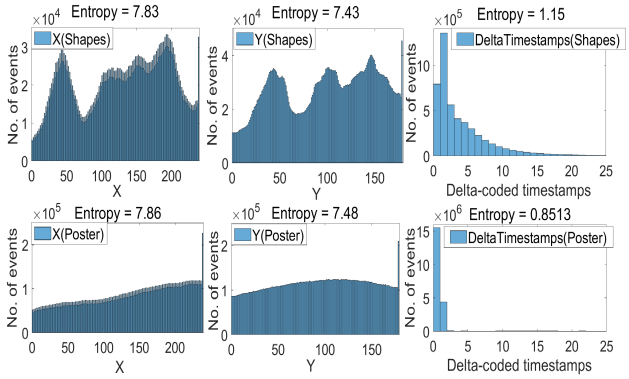

FIGURE 9. First Row: Frequency distribution and entropy of the X, Y and delta-coded time-stamp fields of the *Shapes* sequence. Second Row: Frequency distribution and entropy of the X, Y and delta-coded time-stamp fields of the *Poster* sequence.

Figure 9 also shows the histogram and entropy of the delta coded time-stamp field. The entropy of the delta coded (differences between the consecutive integers) time-stamps is very low. Furthermore, the frequency distribution of the delta coded time-stamp field ranges from 0 to 1, and 0 to 20 for the *Poster* and *Shapes* sequences respectively. In other words, delta coding of time-stamp integers produces a series of zeros and ones for the high rate *Poster* sequence and a stream of integers with the range 0 to 20 for the low rate *Shapes* sequence. This implies that the time-stamp information for the *Poster* and *Shapes* sequences require only one and five bits respectively.

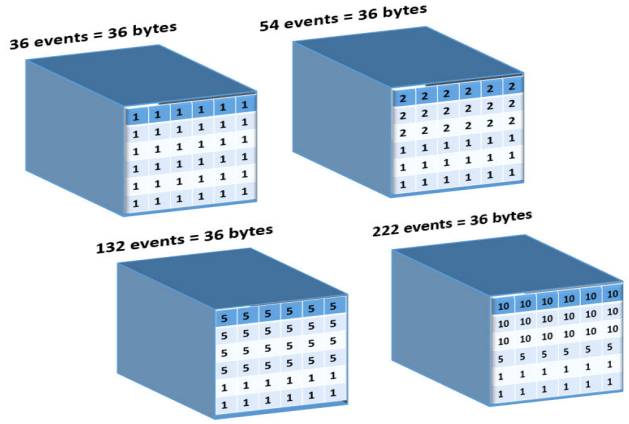

FIGURE 10. Macrocubes containing the location histogram count (AP mode). The higher the average number of events per macrocube, the higher the compression.

Contrary to the spatial address field, the time-stamp field is highly compressible as shown (Figure 9) by the lower entropy computation of this field. Furthermore, the higher the event rate of the mobile DVS, the lower the number of bins of the delta coded time-stamp. Since the spatial address field is nearly incompressible, the delta coded time-stamp field determines the compression performance of a sequence. Therefore, the higher the event rate, the higher the compression ratio as analysed in Section V-B.

Another key observation is the decrease in compression and decompression speed of the mobile DVS scenarios as compared to the scenarios of the static DVS.

C. DECREASE IN COMPRESSION AND DECOMPRESSION SPEED FOR MOBILE DVS SCENARIOS

The majority of the considered strategies (especially dictionary based strategies) shows a considerable decrease in compression and decompression speed for mobile sensor data vs. fixed sensor data. For instance the compression speed of LZMA shows a decline of 65.67 % (25.63 to 8.84). Similarly Zlib, Zstd, Brotli and LZ4 exhibit a decrease of 60.97 %, 30.77 %, 37.54 % and 20.71 % respectively. This is mainly because of the lack of patterns (randomness) in the stream of consecutive spike events. DVS motion fires spike events across the entire pixel array and the spatial address fields (X and Y) in the spike events represent a random stream of integers as depicted by the histogram in Figure 9. The compression and decompression speed depend upon the compressibility of the data. For instance, in dictionary based strategies, repeatable items are stored in a dictionary and a code is assigned as a substitute. The higher the repeatable items in the data to be compressed, the smaller the size of the dictionary and vice versa. Randomness in the data causes the dictionary to fill more quickly. When the dictionary becomes full, a new one is created on the fly and new codes are assigned to the repeatable items in the data. This process requires a lot of computation which results in a decrease of compression and decompression speed.

For mobile sensor data, the decrease in the compression and decompression speed of the Sprintz strategy is negligible (comparing Table 4 and Table 6). This is mainly because the Sprintz strategy encodes only a block of eight consecutive spike events. Therefore, the randomness in the data only affects the compression ratio but does not impact the compression and decompression speed. Furthermore, the Sprintz strategy virtually adds no latency and requires less than 1 KB of memory. The performance difference in the compression ratio between the Sprintz and the best performing strategies is between 10 % to 40 %. Hence, the Sprintz strategy becomes a good candidate to compress mobile DVS data for IoT scenarios where computation resources are limited.

In the subsequent section, we analyse how the spike coding strategy achieves the best compression gains for the mobile DVS data.

D. SPIKE CODING ACHIEVES THE BEST COMPRESSION RATIOS FOR MOBILE DVS SCENARIOS

The spike event distribution in space is either centralized or decentralized. In static sensor scenarios, the motion of

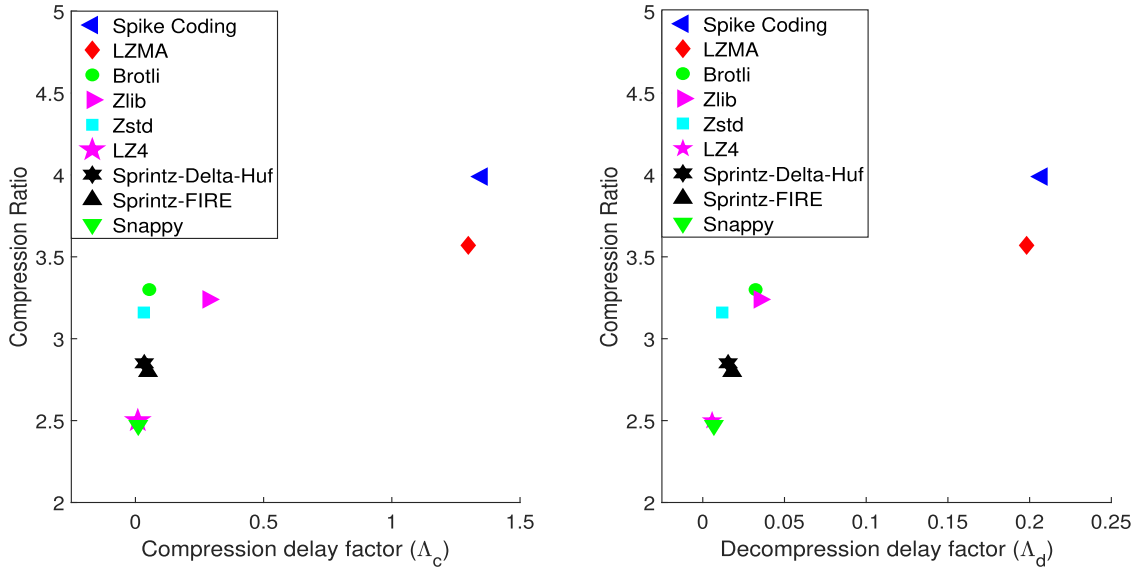

FIGURE 11. Average compression and decompression delay factor (time needed to compress and decompress 1s of data) for mobile sensor scenario.

objects leads to a number of neighbouring pixels to elicit spikes (spatial-centralized). On other hand, the motion of the DVS sensor causes spike firing across the full resolution of the pixel array (spatial-decentralized). However, the firing of spike events across the entire pixel array produces a random stream of integers (X and Y), shown in Figure 9, which are hardly compressible. The AP mode of the spike coding strategy is specifically designed for such scenarios. For instance, consider Figure 10, which shows spike firing across the entire pixel array of the macrocubes (the four macrocubes contain the location histogram event count). The first macrocube contains 36 events, which are represented by 36 bytes, *i.e.,* each pixel has an event and each event count is represented by a byte. In the second macrocube, the event count per pixel is doubled but the number of bytes required to represent spike event count remains the same. Thus, the higher the average event count per macrocube, the higher the compression performance of the spike coding strategy. Furthermore, the temporal correlation between the successive macrocubes' spike event count is also exploited, which results in a better compression performance for the spike coding strategy as compared to the dictionary based strategies.

E. INCREASE IN COMPRESSION AND DECOMPRESSION LATENCY UNDER MOBILE DVS SCENARIOS

The delay factor performance (*w.r.t* compression ratio) for mobile scenarios is reported in Figure 11. A significant increase in compression and decompression delay factors is observed as compared to the latency performance under static DVS scenarios reported in Figure 8. This is mainly because of the high event rate of mobile DVS and low compression and decompression speed. The comparison of the delay factors between the static and mobile scenarios is reported in Figure 12. According to the figure, the average one second compression latency (delay factor) for spike coding is increased from 450 ms to 1352 ms (an increase of 200.44 %). For LZMA, the average compression latency per second is increased by 476.89 % (225 ms to 1298 ms). LZ4 and Snappy achieve the lowest compression latency (less than 10 ms) as shown in Figure 11. Brotli and Zstd are the other two low latency dictionary based compression strategies, incurring a delay of 54 and 33 ms respectively (for the mobile DVS scenarios). It is important to note that the compression speed of the Sprintz strategies remains the same for the static and mobile DVS scenarios. Therefore, the Sprintz strategies exhibit an impressive compression latency of less than 50 ms (1 s of DVS data compressed in less than 50 ms) for mobile scenarios.

Apart from spike coding and LZMA, all the strategies exhibit a decompression latency of less than 50 ms. The impact of DVS mobility on decompression latency is the highest for LZMA and spike coding algorithms, where the latency is approximately 200 ms.

VI. CONCLUSION

Spike event cameras are revolutionary sensors, that offer many advantages over conventional frame-based vision sensors, such as low power, wide dynamic range, low latency and high temporal resolution. The pixels in DVS respond to luminance changes by firing an asynchronous stream of spike events. The spike stream comprises the X and Y spatial addresses, firing time, and the polarity of each spike event. The compression of the DVS output is important for efficient transmission, storage, on-board detection, monitoring and recording. Since the output of the DVS comprises a multivariate stream of integers, it is worth investigating the performance of integer based compression approaches,

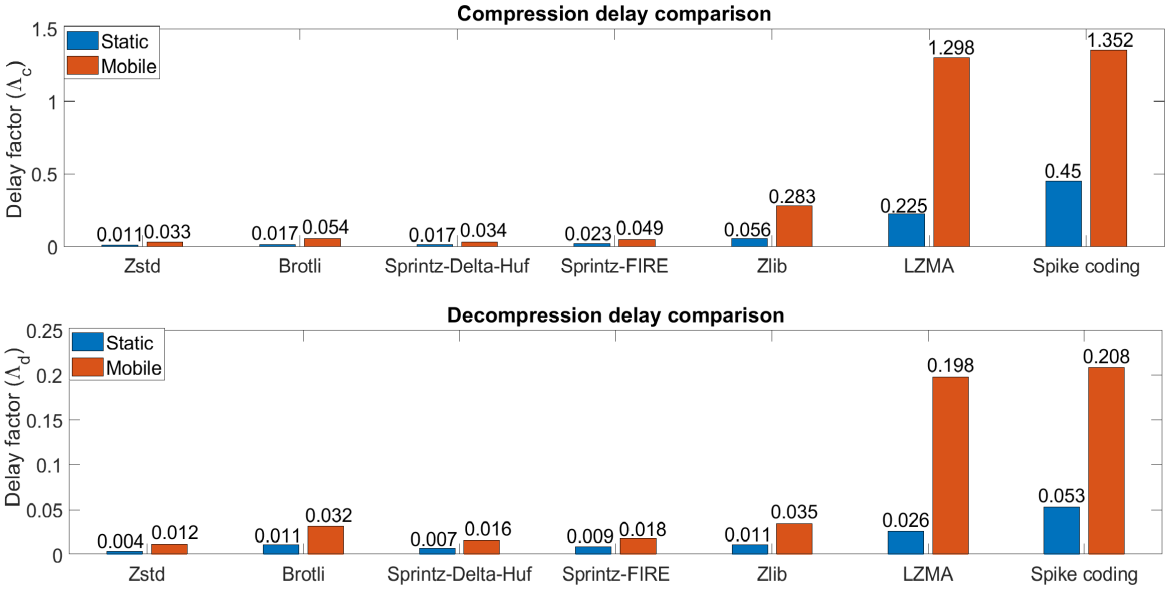

FIGURE 12. Comparison of compression and decompression delay factors between static and mobile sensor scenarios.

IoT specific compression strategies, general purpose lossless compression algorithms, and many other similar compression approaches. This work provides a detailed lossless compression performance analysis of diverse strategies under diverse scenarios including outdoor static DVS, outdoor mobile DVS, indoor static DVS and outdoor static DVS.

According to the detailed experimental analysis provided, LZMA achieves the best compression ratios among all the considered strategies for the case when the DVS is static. On the other hand, Spike coding achieves the best compression ratios when spike events are produced by a visual sensor in motion. In terms of compression and decompression speed and latency, LZ4 and Snappy provide the best performance, while the Brotli algorithm represents a good compromise between compression ratio, compression and decompression speed and latency. We observe a significant compression performance degradation, in terms of ratio, speed and latency, for mobile sensor scenarios vs. fixed sensor scenarios. The detailed analysis presented in this paper provides a benchmark in the selection of a lossless compression approach for a particular application. Furthermore, such a detailed study also provides a benchmark for an efficient design of spike coding algorithms.

REFERENCES

[1] P. Lichtsteiner, C. Posch, and T. Delbruck, "A 128×128 120 dB 15 μs latency asynchronous temporal contrast vision sensor," *IEEE J. Solid-State Circuits*, vol. 43, no. 2, pp. 566–576, Jan. 2008.

[2] P. Lichtsteiner, C. Posch, and T. Delbruck, "A 128×128 120db 30mw asynchronous vision sensor that responds to relative intensity change," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, San Francisco, CA, USA, Feb. 2006, pp. 2060–2069.

[3] N. Khan and M. G. Martini, "Data rate estimation based on scene complexity for dynamic vision sensors on unmanned vehicles," in *Proc. IEEE 29th Annu. Int. Symp. Pers., Indoor Mobile Radio Commun. (PIMRC)*, Bologna, Italy, Sep. 2018, pp. 1174–1178.

[4] N. Khan and M. G. Martini, "Bandwidth modeling of silicon retinas for next generation visual sensor networks," *Sensors*, vol. 19, no. 8, p. 1751, Apr. 2019.

[5] M. Cannici, M. Ciccone, A. Romanoni, and M. Matteucci, "Attention mechanisms for object recognition with event-based cameras," 2018, *arXiv:1807.09480*. [Online]. Available: http://arxiv.org/abs/1807.09480

[6] D. Tedaldi, G. Gallego, E. Mueggler, and D. Scaramuzza, "Feature detection and tracking with the dynamic and active-pixel vision sensor (DAVIS)," in *Proc. 2nd Int. Conf. Event-Based Control, Commun., Signal Process. (EBCCSP)*, Krakow, Poland, Jun. 2016, pp. 1–7.

[7] A. Rigi, F. B. Naeini, D. Makris, and Y. Zweiri, "A novel event-based incipient slip detection using dynamic active-pixel vision sensor (DAVIS)," *Sensors*, vol. 18, no. 2, pp. 1–17, 2018.

[8] E. Mueggler, B. Huber, and D. Scaramuzza, "Event-based, 6-DOF pose tracking for high-speed maneuvers," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Chicago, IL, USA, Sep. 2014, pp. 2761–2768.

[9] A. I. Maqueda, A. Loquercio, G. Gallego, N. Garcia, and D. Scaramuzza, "Event-based vision meets deep learning on steering prediction for self-driving cars," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Salt Lake city, UT, USA, Jun. 2018, pp. 5419–5427.

[10] J. Li, S. Dong, Z. Yu, Y. Tian, and T. Huang, "Event-based vision enhanced: A joint detection framework in autonomous driving," in *Proc. IEEE Int. Conf. Multimedia Expo (ICME)*, Shanghai, China, Jul. 2019, pp. 1396–1401.

[11] M. Martini, N. Khan, Y. Bi, Y. Andreopoulos, H. Saki, and M. Shikh-Bahaei, "Challenges and perspectives in neuromorphic-based visual IoT systems and networks," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Barcelona, Spain, May 2020, pp. 8539–8543.

[12] K. Iqbal, N. Khan, and M. G. Martini, "Performance comparison of lossless compression strategies for dynamic vision sensor data," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Barcelona, Spain, May 2020, pp. 4427–4431.

[13] Z. Xie, S. Chen, and G. Orchard, "Event-based stereo depth estimation using belief propagation," *Frontiers Neurosci.*, vol. 11, pp. 535–549, Oct. 2017.

[14] P. Rogister, R. Benosman, S.-H. Ieng, P. Lichtsteiner, and T. Delbruck, "Asynchronous event-based binocular stereo matching," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 23, no. 2, pp. 347–353, Feb. 2012.

[15] G. Gallego, H. Rebecq, and D. Scaramuzza, "A unifying contrast maximization framework for event cameras, with applications to motion, depth, and optical flow estimation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Salt Lake City, USA, Jun. 2018, pp. 3867–3876.

[16] M. Litzenberger, B. Kohn, A. N. Belbachir, N. Donath, G. Gritsch, H. Garn, C. Posch, and S. Schraml, "Estimation of vehicle speed based on asynchronous data from a silicon retina optical sensor," in *Proc. IEEE Intell. Transp. Syst. Conf.*, Toronto, ON, Canada, Sep. 2006, pp. 653–658.
[17] E. Piatkowska, A. N. Belbachir, S. Schraml, and M. Gelautz, "Spatiotemporal multiple persons tracking using dynamic vision sensor," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. Workshops*, Providence, RI, USA, Jun. 2012, pp. 35–40.
[18] A. Glover and C. Bartolozzi, "Robust visual tracking with a freely-moving event camera," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Vancouver, ON, Canada, Sep. 2017, pp. 3769–3776.
[19] D. P. Moeys, F. Corradi, E. Kerr, P. Vance, G. Das, D. Neil, D. Kerr, and T. Delbruck, "Steering a predator robot using a mixed frame/event-driven convolutional neural network," in *Proc. 2nd Int. Conf. Event-Based Control, Commun., Signal Process. (EBCCSP)*, Krakow, Poland, Jun. 2016, pp. 1–8.
[20] F. Baghaei Naeini, A. M. Alali, R. Al-Husari, A. Rigi, M. K. Al-Sharman, D. Makris, and Y. Zweiri, "A novel dynamic-vision-based approach for tactile sensing applications," *IEEE Trans. Instrum. Meas.*, vol. 69, no. 5, pp. 1881–1893, May 2020.
[21] E. Torti, A. Fontanella, M. Musci, N. Blago, D. Pau, F. Leporati, and M. Piastra, "Embedded real-time fall detection with deep learning on wearable devices," in *Proc. 21st Euromicro Conf. Digit. Syst. Design (DSD)*, Prague, Czech Republic, Aug. 2018, pp. 405–412.
[22] Z. Bi, S. Dong, Y. Tian, and T. Huang, "Spike coding for dynamic vision sensors," in *Proc. Data Compress. Conf.*, Snowbird, UT, USA: IEEE, Mar. 2018, pp. 117–126.
[23] D. Huffman, "A method for the construction of minimum-redundancy codes," *Proc. IRE*, vol. 40, no. 9, pp. 1098–1101, Sep. 1952.
[24] I. H. Witten, R. M. Neal, and J. G. Cleary, "Arithmetic coding for data compression," *Commun. ACM*, vol. 30, no. 6, pp. 520–540, Jun. 1987.
[25] T. Nishimoto and Y. Tabei, "LZRR: LZ77 parsing with right reference," in *Proc. Data Compress. Conf. (DCC)*, Mar. 2019, pp. 211–220.
[26] I. Pavlov. *LZMA SDK (Software Development Kit)*. Accessed: Oct. 1, 2019. [Online]. Available: https://www.7-zip.org/sdk.html
[27] Y. Collet and E. M. Kucherawy. (Jul. 2018). *Zstandard-Real-Time Data Compression Algorithm*. [Online]. Available: http://facebook.github.io/zstd/
[28] P. Deutsch and J.-L. Gailly, *Zlib Compressed Data Format Specification Version 3.3*, document RFC 1950, May 1996.
[29] J. Alakuijala and Z. Szabadka, *Brotli Compressed Data Format*, Internet Engineering Task Force, document RFC 7932, 2016.
[30] Y. Collet. *Finite State Entropy*. [Online]. Available: https://github.com/Cyan4973/FiniteStateEntropy
[31] D. Blalock, S. Madden, and J. Guttag, "Sprintz: Time series compression for the Internet of Things," in *Proc. ACM Interact., Mobile, Wearable Ubiquitous Technol.*, 2018, vol. 2, no. 3, p. 93.
[32] D. Lemire and L. Boytsov, "Decoding billions of integers per second through vectorization," *Softw., Pract. Exper.*, vol. 45, no. 1, pp. 1–29, Jan. 2015.
[33] V. N. Anh and A. Moffat, "Index compression using 64-bit words," *Softw., Pract. Exper.*, vol. 40, no. 2, pp. 131–147, 2010.
[34] S. H. Gunderson. (Apr. 2015). *Snappy: A Fast Compressor/Decompressor*. [Online]. Available: https://github.com/google/snappy
[35] National Engineering Laboratory for Video Technology (NELVT) and Institute of Digital Media (Peking University). *PKU-DVS Dataset*. Accessed: Oct. 1, 2019. [Online]. Available: https://pkuml.org/resources/pku-dvs.html
[36] E. Mueggler, H. Rebecq, G. Gallego, T. Delbruck, and D. Scaramuzza, "The event-camera dataset and simulator: Event-based data for pose estimation, visual odometry, and SLAM," *Int. J. Robot. Res.*, vol. 36, no. 2, pp. 142–149, 2017.
[37] S. Dong, Z. Bi, Y. Tian, and T. Huang, "Spike coding for dynamic vision sensor in intelligent driving," *IEEE Internet Things J.*, vol. 6, no. 1, pp. 60–71, Feb. 2019.

NABEEL KHAN received the B.S. degree in electronics engineering from the Sir Syed University of Engineering and Technology (SSUET), Karachi, Pakistan, in 2007, and the M.Sc. degree in data communication from Kingston University, London, in 2009, and the Ph.D. degree in video optimization over LTE networks. He works as a Research Fellow with the Wireless Multimedia and Networking (WMN) Research Group, Kingston University, where he is working on Internet of Silicon retina (IoSIRE) Project. His research interest includes data communication, computer vision, data compression and modeling, and the Internet of Things.

KHURRAM IQBAL received the B.Sc. degree from the University of Engineering and Technology, Taxila, Pakistan, in 2003, and the M.Sc. degree from King's College London, in 2006. He is currently pursuing the Ph.D. degree in data compression algorithms for dynamic video sensor with the Department of Science, Engineering and Computing, Kingston University. He is also an experienced software test developer with full system development lifecycle experience.

MARIA G. MARTINI (Senior Member, IEEE) received the Laurea in electronic engineering *(summa cum laude)* from the University of Perugia, Italy, in 1998, and the Ph.D. degree in electronics and computer science from the University of Bologna, Italy, in 2002. She is currently a Professor with the Faculty of Science, Engineering and Computing, Kingston University, London, where she also leads the Wireless Multimedia Networking Research Group. She has led the KU team in a number of national and international research projects, funded by the European Commission, such as OPTIMIX, CONCERTO, QoE-NET, Qualinet, U.K. research councils, U.K. Technology Strategy Board/Innovate, U.K., and International Industries. She was an Associate Editor of the *IEEE Signal processing Magazine*, from 2018 to 2020, and the IEEE TRANSACTIONS ON MULTIMEDIA, from 2014 to 2018. She has also been Lead Guest Editor for the IEEE JSAC special issue on QoE-aware wireless multimedia systems and the Guest Editor of the IEEE JOURNAL OF BIOMEDICAL AND HEALTH INFORMATICS, the *IEEE Multimedia*, and the *International Journal on Telemedicine and Applications*, among others. She chaired/organized a number of conferences and workshops. She is part of international committees and expert groups, including the NetWorld2020 European Technology Platform expert advisory Group, the Video Quality Expert Group (VQEG) and the IEEE Multimedia Communications Technical Committee, where she has served as Vice-Chair, from 2014 to 2016, as a Chair, from 2012 to 2014, of the 3D Rendering, Processing, and Communications Interest Group, and as a Key Member of the QoE and multimedia streaming IG. She is an Expert Evaluator for the European Commission and EPSRC among others. Her research interests include QoE-driven wireless multimedia communications, decision theory, video quality assessment, and medical applications. She has authored about 200 scientific articles, contributions to standardization groups (IEEE, ITU), and several patents on wireless video.