Entropy Coding-based Lossless Compression of Asynchronous Event Sequences

Ionut Schiopu and Radu Ciprian Bilcu
Huawei Technologies Oy (Finland) Co. Ltd, Tampere Handset Camera Innovation Lab
Korkeakoulunkatu 7, 33720 Tampere, Finland
ionut.schiopu@huawei.com and radu.ciprian.bilcu@huawei.com

Abstract

The event sensor acquires large amounts of data as events are triggered at microsecond time resolution. In this paper, a novel entropy coding-based method is proposed for encoding asynchronous event sequences. The proposed method employs the event coding framework, where: (i) the input sequence is rearranged as a set of same-timestamp subsequences, where each subsequence is represented of a set of data structures (DSs); and (ii) each DS is encoded by a specific version of the triple threshold partition (TTP) algorithm, where a bitstream collects the binary representation of a set of data elements. A first contribution consists in improving the low-complexity algorithm, LLC-ARES, by modifying the TTP algorithm to employ entropy coding-based techniques to efficient encode the set of data elements. An adaptive Markov model encodes each data element by modelling its symbol probability distribution. Six different types of data elements are distinguished, each having a different support symbol alphabet. Another contribution consists in exploring novel prediction strategies, for the unsorted spatial dimension, and parameter initialization, for the new error distributions. The experimental evaluation demonstrates that the proposed method achieves an improved average coding performance of $28.03\%, 35.27\%$, *and* 64.54% *compared with the state-of-the-art data compression codecs Bzip2, LZMA, and ZLIB, respectively, and* 21.4% *compared with LLC-ARES.*

1. Introduction

Recently, the neuromorphic engineering domain proposed a new type of sensor, bio-inspired by the human brain, called the event sensor [15]. In contrast to the conventional camera, where the incoming light intensity is captured by a video sequence, the event camera outputs a sequence of asynchronous events triggered at specific pixel position, at up to one microsecond times resolution, and only to report either an increase or a decrease of the incoming light intensity. Since the event camera captures only the dynamic information and removes the unnecessary static information, it is now widely used in computer vision where the RGB and event-based solutions already provide the state-of-the-art performance for many applications, such as feature detection and tracking [35], optical flow estimation [23, 24, 31], object segmentation [29, 32], stereo depth estimation [27, 30], and many others. For a comprehensive literature review see [10]. Due to the high temporal resolution, the event camera generates large amounts of data as high bit-rate levels are achieved using the raw representation of 8 bytes per event. Hence, efficient event coding solutions are required to reduce the high bitrate levels.

The event data compression domain is understudied as recently more efficient high resolution event sensors, e.g., Prophesee's Gen4 [9], Samsung's dynamic vision sensor (DVS) [25], are available to consumers. Several solutions are proposed to compress the asynchronous event sequences without any information loss. In [4], the codec is removing the redundancy of the spatial and temporal information using three strategies: adaptive macro-cube partitioning structure, the address-prior mode, and time-prior mode. In [7], the authors propose to further extend the lossless codec by using octree-based cube partition and a flexible inter-cube prediction. This strategy was evaluated on low-resolution event datasets and coding performance comparison is not possible as the used dataset is partly made available only for academic purpose and on a case-by-case basis. In [21], the Low-complexity Lossless Compression of AsynchRonous Event Sequence (LLC-ARES) method was proposed by first rearranging the input event sequence using the Same-Timestamp (ST) representation and then employing the triple threshold partition (TTP) algorithm to generate a set of data elements, which are represented using a reduced number of bits and simply collected in a bitstream as LLC-ARES is designed to be suitable for hardware implementation into event signal-processing (ESP) chips. Here, the proposed method employs the event coding framework [21] and improves LLC-ARES by modifying the TTP-based algorithm to employ entropy coding-based techniques to encode the set of data elements, therefore, the codec is suitable for hardware implementation into system-on-a-chip (SoC).

Another approach consist in employing different strategies, such as [2, 16, 33], to generate a sequence of synchronous Event Frames (EFs), so that the event data can be consumed as "intensity-like" frames/images. In [13], a time aggregation-based lossless video encoding method based on an event-accumulation process is used to create two event frames of positive and negative polarity counts, which are further encoded using HEVC [26]. In [19], a context-based lossless compression method is proposed to encode (sum-accumulation) EFs sequences, where the event spatial information and the event polarity information are encoded separately by employing two different algorithms. In [20], a low-complexity lossless compression method is proposed to encode the EFs sequences using run-length encoding and Elias coding [8] so that the proposed solution is suitable for hardware implementation into ESP chips. In [17], the asynchronous event sequences are treated as a point cloud representation, so the proposed method employs a point cloud compression based strategy.

Traditional data compression strategies can also be employed to encode the event data. In [12], the authors studied the performance of data compression codecs and show that the dictionary-based methods usually offer the best performance, e.g., the Zeta Library (ZLIB) [6], the Lempel–Ziv–Markov chain algorithm (LZMA) [18] (used by 7-Zip), and the Bzip2 algorithm [22] (based on the well-known Burrows–Wheeler transform [5]).

In the lossy event data compression domain some type of information loss was accepted with the goal of achieving very low bitrates. In [34], the authors propose a macro cuboids partition of the raw event data and they employ a novel spike coding framework, inspired from video coding, to encode spike segments. In [3], the authors propose a lossy coding method based on a quad tree segmentation map derived from the adjacent intensity images, which requires access to the a hybrid event sensor that comprises of a pair of a DVS sensor and an active pixel sensor (APS).

The goal of this work is to propose a novel lossless compression method for asynchronous event sequences. The novel contributions of this work are summarized as follows: **(1)** a novel entropy coding-based codec is proposed for raw event data; **(2)** the TTP algorithm [21] was modified to additionally employ entropy coding-based techniques by allocating a set of corresponding adaptive Markov models (AMM) to encode each generated data element instead of representing it on a number of bits; **(3)** novel prediction strategies are proposed for the unsorted spatial dimension; **(4)** a novel parameter initialization procedure is introduced to model the new prediction error distribution.

The remainder of the paper is structured as follows. Sec. 2 details the proposed method. Sec. 3 presents the experimental evaluation. Sec. 4 draws the conclusions.

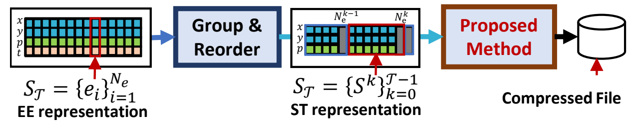


Figure 1. The event coding framework [21] was modified to employ the proposed method and encode the input event sequence, $\mathcal{S}_\mathcal{T}$, under the Same-Timestamp (ST) representation.

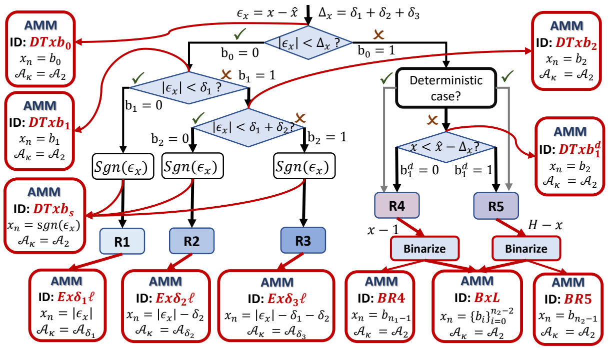
2. Proposed Method

A novel entropy coding-based lossless compression method is proposed to encode the raw data captured by a DVS sensor and stored as asynchronous event sequences. Fig. 1 show the event coding framework [21], where the representation of the input asynchronous event sequence is changed from Event-by-Event (EE) to ST, i.e., as a set of same-timestamp subsequences, where each subsequence is represented of a set of data structures (DSs), see Sec. 2.1. Each ST DS is then encoded by employing a specific version of the TTP algorithm, where a decision tree is used to divide the input range into several coding ranges arranged at concentric distances from a prediction, see Sec. 2.2. The TTP-based algorithms generate a set of data elements, which are represented using a reduced number of bits and then simply collected in a bitstream.

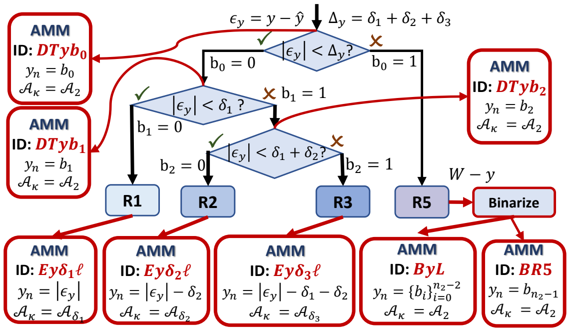
In this work, the event coding framework is modified to employ the proposed method, called Entropy coding-based Lossless Compression of ARES (ELC-ARES), described in Sec. 2.3. ELC-ARES encoded each DS in the ST representation by employing a modified TTP variation designed using more efficient entropy coding techniques. New prediction strategies are also explored, see Sec. 2.4. A new threshold parameter initialization is proposed, see Sec. 2.5. The detailed algorithmic implementation is presented in Sec. 2.6.

2.1. Same-Timestamp Representation

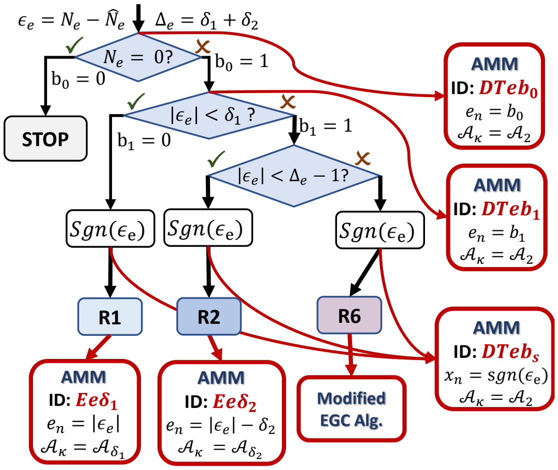
For a $W \times H$ resolution camera, an asynchronous event, $e_i = (x_i, y_i, p_i, t_i)$, collects the spatial, (x_i, y_i), $\forall x_i \in [1, H], y_i \in [1, W]$, polarity, $p_i \in \{-1, 1\}$, and timestamp, t_i, information. An event sequence, $\mathcal{S}_\mathcal{T} = \{e_i\}_{i=1}^{N_e}$, stores all N_e events triggered during $\mathcal{T}$ timestamps using the EE representation as a sequence $\mathbf{e}^{N_e} = e_1 e_2 \ldots e_{N_e}$. The ST representation rearranges $\mathcal{S}_\mathcal{T}$ as a set of ST subsequences, $\mathcal{S}_\mathcal{T} = \{S^k\}_{k=0}^{\mathcal{T}-1}$, where each ST subsequence, $S^k = \{(x_i^k, y_i^k, p_i^k)\}_{i=1}^{N_e^k}$, collects all the N_e^k events that were triggered at the same timestamp t^k. Each S^k is firstly ordered in the ascending order of the largest spatial information dimension, e.g., y_k^i, and then of the remaining dimension (x_k^i) if $\epsilon_{y_i^k} = y_i^k - y_{i-1}^k = 0$. ELC-ARES encodes $\mathcal{S}_\mathcal{T}$ as a set of $\mathcal{T}$ subsequence (S^k), each represented using four DSs: N_e^k, $\{y_i^k\}_{i=1}^{N_e^k}$, $\{x_i^k\}_{i=1}^{N_e^k}$, and $\{p_i^k\}_{i=1}^{N_e^k}$, see Fig. 1.



(a) The modified TTP_x version.



(b) The modified TTP_y version.



(c) The modified TTP_e version.

Figure 2. Each TTP algorithm variation is modified to employing entropy coding-based techniques. The red rectangles mark the additional blocks introduced by the proposed method.

2.2. Triple Threshold Partition Algorithm

The TTP algorithm divides the input data range into several coding ranges arranged at concentric distances from an initial prediction. The binary representation of the residual error is partitioned into smaller intervals based on a decision tree designed using the triple threshold $\mathbf{\Delta} = (\delta_1, \delta_2, \delta_3)$.

Type	DS	Model ID	$\mathcal{A}$
Decision Tree	$\{x_i^k\}_{i=1}^{N_e^k}$	$DTxb_0, DTxb_1, DTxb_2$	$\mathcal{A}_2$
		$DTxb_s, DTxb_1^x$	
	$\{y_i^k\}_{i=1}^{N_e^k}$	$DTyb_0, DTyb_1, DTyb_2$	
	N_e^k	$DTeb_0, DTeb_1, DTeb_s$	
Encode x	$\{x_i^k\}_{i=1}^{N_e^k}$	$Ex\delta_1 1, Ex\delta_1 2, Ex\delta_1 3$	$\mathcal{A}_{\delta_1}$
		$Ex\delta_2 1, Ex\delta_2 2$	$\mathcal{A}_{\delta_2}$
		$Ex\delta_3 1, Ex\delta_3 2$	$\mathcal{A}_{\delta_3}$
Encode y	$\{y_i^k\}_{i=1}^{N_e^k}$	$Ey\delta_1 1, Ey\delta_1 2$	$\mathcal{A}_{\delta_1}$
		$Ey\delta_2 1, Ey\delta_2 2, Ey\delta_2 3$	$\mathcal{A}_{\delta_2}$
		$Ey\delta_3 1, Ey\delta_3 2, Ey\delta_3 3$	$\mathcal{A}_{\delta_3}$
Encode e	N_e^k	$Ee\delta_1$	$\mathcal{A}_{\delta_1}$
		$Ee\delta_2$	$\mathcal{A}_{\delta_2}$
Polarity	$\{p_i\}_{i=1}^{N_e^k}$	$P1$	$\mathcal{A}_2$
Binary	$\{x_i^k\}_{i=1}^{N_e^k}$	$BxL, BR4, BR5$	$\mathcal{A}_2$
	$\{y_i^k\}_{i=1}^{N_e^k}$	$ByL, BR5$	

Table 1. The list of AMMs employed by ELC-ARES.

To encode each S^k, three TTP versions were proposed based on the type of encoded ST DS: TTP_e to encode N_e^k, TTP_y to encode $\{y_i^k\}_{i=1}^{N_e^k}$, and TTP_x to encode $\{x_i^k\}_{i=1}^{N_e^k}$.

Fig. 2a depicts the TTP_x version, where five ranges (R1-R5) are created based on an input prediction $\hat{x}$, see Sec. 2.4, and a search range $[1, H]$. LLC-ARES directly writes into the compressed file the following DSs: *(a)* the decision tree using up to four bits ($b_0 b_1 b_s$, $b_0 b_1 b_2 b_s$, $b_0 b_1^d$, or b_0); and *(b1)* the binary representation of the residual error, $\epsilon_x = x - \hat{x}$, using n_{δ_k} bits for R1-R3; *(b2)* the true value $x - 1$ using n_1 bits for R4; *(b3)* $H - x$ using n_2 bits for R5. Fig. 2b depicts the TTP_y version where, similarly, four ranges (R1-R3, R5) are created using the input prediction $\hat{y}$ and the search range $[1, W]$, while Fig. 2c depicts the TTP_e version where three ranges (R1-R2, R5) are created using the input prediction $\hat{N}_e$. In contrast, ELC-ARES modifies each TTP version to encode all the corresponding DSs using more efficient entropy coding techniques, mark in Fig. 1 using red rectangles and described in-detail in Sec. 2.3 below.

2.3. Proposed Entropy Coding-based Solution

Given $\mathbf{x}^n = x_0 x_1 \ldots x_{n-1}$, a sequence of n elements x_j selected from the an alphabet $\mathcal{A}_\kappa = \{s_0, s_1, \ldots, s_{\kappa-1}\}$ of κ symbols, and $C_\kappa = [c_0\, c_1\, \ldots\, c_{\kappa-1}]$, where c_i is the frequency of symbol s_i. The sequence $\mathbf{x}^n$ is encoded using the probability distribution $P_n = [p_0\, p_1\, \ldots\, p_{\kappa-1}]$, where $p_i = \frac{c_i}{n}$. The first element, x_0, is first encoded using an uniform distribution $P_0 = [\frac{1}{\kappa}\, \frac{1}{\kappa}\, \ldots\, \frac{1}{\kappa}]$ and then P_0 is updated to P_1 using a probability estimator. Similarly, a next element x_n is first encoded using P_n and then P_n is updated to P_{n+1}. Here, the Laplace estimator [14] computes the probability that the next element x_n is symbol s_i as follows:

$$p_i(x_n = s_i) = \frac{n_{s_i} + 1}{n + \kappa}, \forall i = 0, 1, \ldots, \kappa - 1. \quad (1)$$

An order$-N$ AMM uses κ^N contexts of previous N symbols, where x_n is encoded given an alphabet $\mathcal{A}_\kappa$. In this work, only the order-0 AMM is used as our test show inconsistent coding gains when searching for the optimal model order while using an increased complexity.

ELC-ARES encodes S^k by employing model $P1$ for $\{p_i^k\}_{i=1}^{N_e^k}$, and several models for the data elements generated by TTP_e, TTP_y, and TTP_x for N_e^k, $\{y_i^k\}_{i=1}^{N_e^k}$, and $\{x_i^k\}_{i=1}^{N_e^k}$, respectively. Six AMM types are distinguished.

(1) Decision Tree (DT), for encoding the decision tree of: TTP_x (DTx), see Fig. 2a; TTP_y (DTy), see Fig. 2b; and TTP_e (DTe), see Fig. 2c. E.g., the modified TTP_x algorithm encodes the following data elements: $b_i, \forall i = 0, 1, 2, 3$ to signal R1-R3; b_1^d to signal R4-R5; and b_s to signal $sgn(\epsilon_x)$. All DT models use alphabet $\mathcal{A}_2 = \{0, 1\}$.

(2) Encode x (Ex), for encoding $|\epsilon_x| = |x - \hat{x}|$ using: range R1, where model $Ex\delta_1\ell$ (where ℓ depends on δ_i value) encodes $x_n = |\epsilon_x|$ using $\mathcal{A}_{\delta_1} = \{0, 1, \ldots, \delta_1 - 1\}$; range R2, where model $Ex\delta_2\ell$ encodes $x_n = |\epsilon_x| - \delta_1$ using $\mathcal{A}_{\delta_2}$; and range R3, where model $Ex\delta_3\ell$ encodes $x_n = |\epsilon_x| - \delta_1 - \delta_2$ using $\mathcal{A}_{\delta_3}$, see Fig. 2a.

(3) Encode y (Ey), for encoding $|\epsilon_y| = |y - \hat{y}|$ similarly as Ex, i.e., by using one of R1, R3, and R3 ranges of TTP_y with $Ey\delta_1\ell$, $Ey\delta_2\ell$, and $Ey\delta_3\ell$, respectively, see Fig. 2b.

(4) Encode e (Ee), for encoding $|\epsilon_e| = |N_e^k - \hat{N}_e^k|$ similarly as Ex, i.e., by using one of R1 and R2 ranges of TTP_e with model $Ee\delta_1$ and $Ee\delta_2$, respectively, see Fig. 2c.

(5) Polarity (P), for encoding $p_i^k, \forall i = 1, 2, \ldots, N_e^k$ using the model $P1$, having $\mathcal{A}_2 = \{-1, 1\}$.

(6) Binary (B), for the binary representation of: **(6.i)** $(x - 1)_{(10)} = \overline{b_{n_1-1} \ldots b_1 b_0}_{(2)}$ in R4 (of TTP_x) by encoding $x_n = b_{n_1-1}$ using model $BR4$ and $x_{n+i} = b_i, \forall i = 0, 1, \ldots, n_1 - 2$ using model BxL; **(6.ii)** $(H - x)_{(10)} = \overline{b_{n_2-1} \ldots b_1 b_0}_{(2)}$ in R5 (of TTP_x) by encoding $x_n = b_{n_2-1}$ using model $BR5$ and $x_{n+i} = b_i, \forall i = 0, 1, \ldots, n_2 - 2$ using model BxL; **(6.iii)** $(W - y)_{(10)} = \overline{b_{n_2-1} \ldots b_1 b_0}_{(2)}$ in R5 (of TTP_y) by encoding $x_n = b_{n_2-1}$ using model $BR5$ and $x_{n+i} = b_i, \forall i = 0, 1, \ldots, n_2 - 2$ using model ByL.

Tab. 1 lists all the order-0 AMMs introduced by ELC-ARES, i.e., 33 models, together with their ID and alphabet size. Sets of $11, 2, 1,$ and 4 models are introduced for the DT, Ee, P, and B types, respectively, for encoding the DSs generated by TTP_x, TTP_y, and TTP_e. While sets of 7 and 8 models are introduced for the Ex and Ey types, respectively, to cover all different values of that may be possible when updating $\mathbf{\Delta}$, see Sec. 2.5.

In R6 of TTP_e, large $|\epsilon_e|$ are codified by employing Elias Gamma Coding (EGC) [8] for $x_\gamma = |\epsilon_e| - \Delta_e - 2$, see Fig. 2c. The modified TTP_e employs the modified EGC where the generated bitstream is further encoded using two AMMs: $BR4$ for the unary representation of x_γ number of bits; $BR5$ for the remaining x_γ binary digits, see Sec. 2.6.

2.4. Exploring New Prediction Strategies

In LLC-ARES, prediction $\hat{y}_i^k$ is fixed to y_{i-1}^k due to the ST representation, while $\hat{N}_e^k$ depends on the scene motion. Let us denote $w5$ the LLC-ARES prediction strategy, where the median function is applied either on a small prediction window of size 5 or a larger prediction window of size 15.

In this paper, an improved prediction, $\hat{x}_i^k$, is computed using different strategies using $\{x_j^k\}_{j=1}^{i-1}$. Firstly, let us denote $w50$ as the improved and more complex prediction strategy where the median function is applied on a very large prediction window of size 50, i.e., for $i > 2, \hat{x}_i^k = med(\{x_{i-j}^k\}_{j=1:50, i>j})$. Secondly, let us denote $Px1$ the prediction strategy where TTP is in charge of handling large prediction errors (using R4-R6) and computation complexity is minimized as $\hat{x}_i^k = x_1^k$, i.e., the position of the first event trigger at t^k.

2.5. New Threshold Initialization

Let us denote $T345$ the LLC-ARES threshold initialization of $\mathbf{\Delta}_H^{k+1}$, see Eq. (2). Since $Px1$ computes a worst prediction than $w5$ or $w50$, a new threshold initialization, denote $T455$, is introduced so that much large residual errors are allocated to ranges R1-R3, see Eq. (3).

$$\mathbf{\Delta}_{H,T345}^{k+1} = \begin{cases} \mathbf{\Delta}^{e1} = (2^3, 2^4, 2^5) & \text{if } k = 0; \\ (2^5, 2^5, 2^6) & \text{if } k > 0 \ \&\ \epsilon^k < 8; \\ (2^4, 2^4, 2^5) & \text{otherwise.} \end{cases} \quad (2)$$

$$\mathbf{\Delta}_{H,T455}^{k+1} = \begin{cases} \mathbf{\Delta}^{e1} = (2^4, 2^5, 2^5) & \text{if } k = 0; \\ (2^6, 2^6, 2^6) & \text{if } k > 0 \ \&\ \epsilon^k < 8; \\ (2^5, 2^5, 2^5) & \text{otherwise.} \end{cases} \quad (3)$$

$T345$ may set $\mathbf{\Delta}_H^{k+1}$ as follows: δ_1 as 2^3, 2^4, or 2^5 for models $Ex\delta_1 1$, $Ex\delta_1 2$, or $Ex\delta_1 3$, respectively; δ_2 as 2^4, or 2^5 for models $Ex\delta_2 1$ or $Ex\delta_2 2$, respectively; and δ_3 as 2^5, or 2^6 for models $Ex\delta_3 1$ or $Ex\delta_3 2$, respectively.

$T455$ may set $\mathbf{\Delta}_H^{k+1}$ as follows: δ_1 as 2^4, 2^5, or 2^6 for models $Ex\delta_1 1$, $Ex\delta_1 2$, or $Ex\delta_1 3$, respectively; δ_2 as 2^5 or 2^6 for models $Ex\delta_2 1$ or $Ex\delta_2 2$, respectively; and δ_3 as 2^5 or 2^6 for models $Ex\delta_3 1$ or $Ex\delta_3 2$, respectively.

$\mathbf{\Delta}_W$ may be set as follows: δ_1 as 2^1 or 2^2 for models $Ey\delta_1 1$ or $Ey\delta_1 2$, respectively; δ_2 as 2^1, 2^2, or 2^3 for models $Ey\delta_2 1$, $Ey\delta_2 2$, or $Ey\delta_2 3$, respectively; and δ_3 as 2^2, 2^3, or 2^4 for models $Ey\delta_3 1$, $Ey\delta_3 2$, or $Ey\delta_3 2$, respectively.

Algorithm 1: ELC-ARES Encoder of S^k

Input: $N_e^k, \{y_i^k\}_{i=1}^{N_e^k}, \{x_i^k\}_{i=1}^{N_e^k}, \{p_i^k\}_{i=1}^{N_e^k}), \{N_e^j\}_{j=k-3}^{k-1}, H, W, P_x$ (prediction strategy), and T_x (initialization);

- **Initialize** all models listed in Tab. 1;
- $\hat{N}_e^k \leftarrow$ **Predict** N_e^k using $\{N_e^j\}_{j=k-3}^{k-1}$;
- **Encode** N_e^k using Alg. E1 ($N_e^k, \hat{N}_e^k, \boldsymbol{\Delta}_e$);
- **if** $N_e^k > 0$ **then**
  - **Encode** y_1^k using Alg. E4 ($y_1^k, \hat{y}_r^k, \epsilon_{y_1^k}, [1,\ W], \boldsymbol{\Delta}^{e1}$);
  - **Encode** x_1^k using Alg. E4 ($x_1^k, \hat{x}_r^k, \epsilon_{y_1^k}, [1,\ H], \boldsymbol{\Delta}^{e1}$);
  - **Encode** $p_n = p_1^k$ using model $P1$; **Update** $P1$;
  - **for** $i = 2, 3, \ldots, N_e^k$ **do**
    - **Encode** y_i^k by Alg. E3 ($y_i^k, y_{i-1}^k, [y_{i-1}^k,\ W], \boldsymbol{\Delta}_W^k$);
    - $\hat{x}_i^k \leftarrow$ **Predict** x_i^k using $P_x(\{x_j^k\}_{j=1,2,\ldots,i-1})$;
    - **Encode** x_i^k using Alg. E4 ($x_i^k, \hat{x}_i^k, \epsilon_{y_i^k}, [1,\ H], \boldsymbol{\Delta}_H^k$);
    - **Encode** $p_n = p_i^k$ using model $P1$; **Update** $P1$;
  - $\boldsymbol{\Delta}_H^{k+1} \leftarrow$ **Update** $\boldsymbol{\Delta}_H^k$ using $\epsilon^k = y_{N_e^k}^k - y_1^k$ and T_x;
  - $\boldsymbol{\Delta}_W^{k+1} \leftarrow$ **Update** $\boldsymbol{\Delta}_W^k$ using $\epsilon^k = y_{N_e^k}^k - y_1^k$;

Algorithm 2: ELC-ARES Decode of S^k

Input: $\{N_e^j\}_{j=k-3}^{k-1}, H, W, P_x, T_x$;

Output: $N_e^k, \{y_i^k\}_{i=1}^{N_e^k}, \{x_i^k\}_{i=1}^{N_e^k}, \{p_i^k\}_{i=1}^{N_e^k}$;

- **Initialize** all models listed in Tab. 1;
- $\hat{N}_e^k \leftarrow$ **Predict** N_e^k using $\{N_e^j\}_{j=k-3}^{k-1}$;
- $N_e^k \leftarrow$ **Decode** using Alg. D2 ($\hat{N}_e^k, \boldsymbol{\Delta}_e$);
- **if** $N_e^k > 0$ **then**
  - $y_1^k \leftarrow$ **Decode** using Alg. D1 ($\hat{y}_r^k, \epsilon_{y_1^k}, [1,\ W], \boldsymbol{\Delta}^{e1}$);
  - $x_1^k \leftarrow$ **Decode** using Alg. D1 ($\hat{x}_r^k, \epsilon_{y_1^k}, [1,\ H], \boldsymbol{\Delta}^{e1}$);
  - $p_1^k \leftarrow$ **Decode** using model $P1$; **Update** $P1$;
  - **for** $i = 2, 3, \ldots, N_e^k$ **do**
    - $y_i^k \leftarrow$ **Decode** by Alg. D4 ($y_{i-1}^k, [y_{i-1}^k,\ W], \boldsymbol{\Delta}_W^k$);
    - **Predict** x_i^k using $P_x(\{x_j^k\}_{j=1,2,\ldots,i-1})$;
    - $x_i^k \leftarrow$ **Decode** using Alg. D1 ($\hat{x}_i^k, \epsilon_{y_i^k}, [1,\ H], \boldsymbol{\Delta}_H^k$);
    - $p_i^k \leftarrow$ **Decode** using model $P1$; **Update** $P1$;
  - $\boldsymbol{\Delta}_H^{k+1} \leftarrow$ **Update** $\boldsymbol{\Delta}_H^k$ using $\epsilon^k = y_{N_e^k}^k - y_1^k$ and T_x;
  - $\boldsymbol{\Delta}_W^{k+1} \leftarrow$ **Update** $\boldsymbol{\Delta}_W^k$ using $\epsilon^k = y_{N_e^k}^k - y_1^k$;
- **Return** $S^k(N_e^k, \{y_i^k\}_{i=1}^{N_e^k}, \{x_i^k\}_{i=1}^{N_e^k}, \{p_i^k\}_{i=1}^{N_e^k})$;

2.6. Algorithmic Implementation

2.6.1 ELC-ARES Encoder

Alg. 1 presents the pseudocode of ELC-ARES encoder for subsequence S_k of timestamp t^k, see Fig. 1. S_k is stored under ST representation as the set of following DSs: N_e^k, $\{y_i^k\}_{i=1}^{N_e^k}$, $\{x_i^k\}_{i=1}^{N_e^k}$, and $\{p_i^k\}_{i=1}^{N_e^k}$.

Almost every line of the LLC-ARES encoder was modified in the ELC-ARES encoder by: *(i)* modifying the TTP algorithm to employ entropy coding-based techniques, see Sec. 2.3; *(ii)* employing a new prediction strategy, see Sec. 2.4; *(iii)* employing a new $\boldsymbol{\Delta}_H^{k+1}$ initialization, see Sec. 2.5. For pseudocode of Algs. E1-E4 is presented in the supplementary material file. Algs. E1, E2, and E4 present the modified TTP_e (see Fig. 2c), TTP_y (Fig. 2b), and TTP_x (see Fig. 2a) encoder, respectively, while Alg. E2 presents the modified EGC encoder.

2.6.2 ELC-ARES Decoder

Alg. 2 presents the pseudocode of ELC-ARES Decoder for subsequence S_k as the set of following DSs: N_e^k, $\{y_i^k\}_{i=1}^{N_e^k}$, $\{x_i^k\}_{i=1}^{N_e^k}$, and $\{p_i^k\}_{i=1}^{N_e^k}$. Similarly, almost every line of the LLC-ARES decoder was modified in the ELC-ARES decoder. For pseudocode of Algs. D1-D4 is presented in the supplementary material file. Algs. D1, D2, and D4 present the modified TTP_x (see Fig. 2a), TTP_e (see Fig. 2c), and TTP_y (Fig. 2c) decoder, while Algs. D4 presents the pseudocode of the modified EGC decoder, employed in Alg. D2.

3. Experimental Evaluation

3.1. Experimental Setup

The experimental evaluation is performed on DSEC [1, 11] training data, containing 82 sequences of $W \times H = 640 \times 480$ pixel resolution, where only the first 100 s or each sequence are retained. Here, the sequences are sorted in ascending order of event acquisition density, see Fig. 3.

The raw data size is computed as 8 bytes per event (64 bpev). The results are reported using: **(a)** Relative compression (RC), ratio between compressed size and compressed anchor size; **(b)** Compression ratio (CR), ratio between raw data size and compressed size; **(c)** Bit rate (BR) (bpev), number of bits needed to encode one event; **(d)** Event density (Mevps), number of events encoded per second; **(e)** Runtime (μspev), average time needed to encode one event.

ELC-ARES is written in C, and its performance is compared with the following state-of-the-art codecs: **(a)** ZLIB [6] (version 1.2.3 available online [28]); **(b)** LZMA [18]; **(c)** Bzip2 (version 1.0.5 available online [22]); and **(d)** LLC-ARES [21] (designed for ESP integration). The ST representation provides an improved performance of up to 96% compared with the EE order, see [21]. Hence, in this paper, data is always stored using the ST representation.

3.2. Ablation study

LLC-ARES is designed to employ the $w5$ prediction strategy and the $T345$ initialization. Here, ELC-ARES is designed to employ the $Px1$ prediction strategy and the $T455$ initialization. An ablation study is introduced to analyse the performance of the proposed prediction strategies and threshold initializations using the following versions: *(1)* ELC-ARES-v1, employs the same configuration as LLC-ARES; *(2)* ELC-ARES-v2, employs $w50$ and

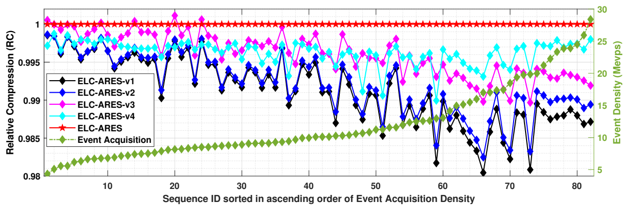


Figure 3. Compression results of ELC-ARES versions. Sequences are sorted in ascending order of event acquisition density.

Codec	P_x	T_x	CR	RC$_{LLC}$	RC$_{ELC}$
LLC-ARES	*w5*	$T345$	4.371	1	0.824
ELC-ARES-v1	*w5*	$T345$	5.256	1.202	0.991
ELC-ARES-v2	*w50*	$T455$	5.261	1.204	0.992
ELC-ARES-v3	*w50*	$T345$	5.282	1.208	0.996
ELC-ARES-v4	*Px1*	$T345$	5.286	1.209	0.996
ELC-ARES	*Px1*	$T455$	**5.306**	**1.214**	**1**

Table 2. Average performance of ELC-ARES versions over DSEC

Codec	ZLIB	LZMA	Bzip2	LLC-ARES	ELC-ARES
CR	3.225	3.922	4.144	4.371	**5.306**
BR	20.32	16.80	15.91	14.82	**12.58**

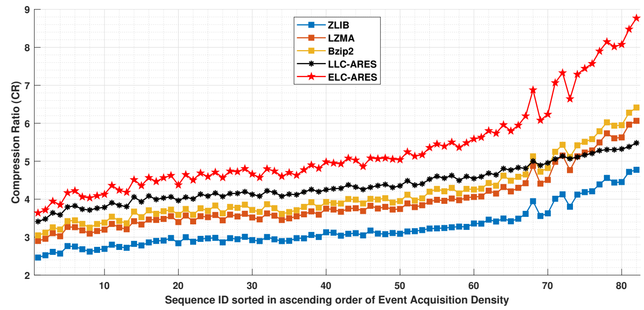
Table 3. Average coding performance over DSEC

$T345$; *(3)* ELC-ARES-v3, employs $w50$ and $T455$; and *(4)* ELC-ARES-v4, employs $Px1$ and $T455$.

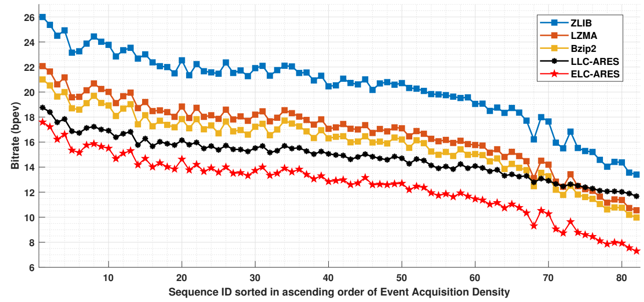
Fig. 3 presents the RC$_{ELC}$ results over DSEC to compare the performance of each ELC-ARES version relative to the proposed ELC-ARES algorithm, which is set as anchor codec. Tab. 2 shows the average results in terms of CR, RC$_{ELC}$ (relative to ELC-ARES), and RC$_{LLC}$ (relative to LLC-ARES). One can note that the entropy coding techniques provide an average improvement of 20.2% compared with LLC-ARES. ELC-ARES provides 21.4% improvement compared with LLC-ARES (or equivalent the LLC-ARES performance is only 82.4% of the LLC-ARES performance). The event coding framework-based codecs provides an improved performance regardless of the prediction quality as the extreme prediction cases are efficiently detected and treated separability.

3.3. Lossless compression results

Fig. 4 shows the compression results over DSEC, where Fig. 4a shows CR results and Fig. 4b the BR results. Note that ELC-ARES provides the best performance for any event sequence acquisition density. Tab. 3 shows the average coding performance over DSEC. Note that, compared with the state-of-the-art codecs, LLC-ARES, Bzip2, LZMA, and ZLIB, the proposed codec, ELC-ARES, provides an average CR improvement of $21.4\%, 28.03\%, 35.27\%$, and 64.54%, respectively, and an average BR improvement of $17.8\%, 26.5\%, 33.6\%$, and 61.56%, respectively.

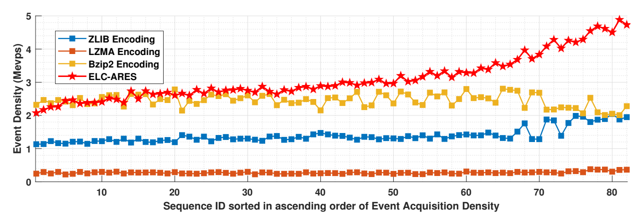


(a) Compression Ratio results.

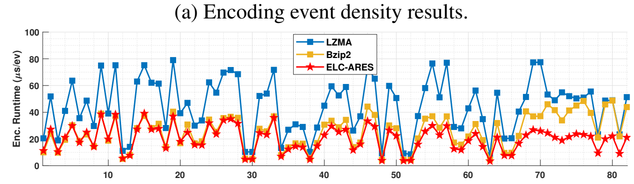


(b) Bitratre results.

Figure 4. Lossless compression results over DSEC.



(a) Encoding event density results.



(b) Encoding runtime results.

Figure 5. Runtime results over DSEC.

3.4. Runtime results

Fig. 5 shows the average runtime performance over DSEC, where Fig. 5a shows the encoding event density results, and Fig. 5b shows the encoding runtime results. The proposed method provides the best performance for the medium and high event density sequences, while only Bzip2 provides a similar performance for low event den-

Codec	ZLIB	LZMA	Bzip2	ELC-ARES
Event density	1.392	0.275	2.453	**3.109**
Runtime	44.70	227.27	25.74	**20.23**

Table 4. Average runtime performance over DSEC.

sity sequences. Tab. 4 shows the average coding performance over DSEC. Compared with the state-of-the-art lossless compression codecs, suitable for SoC integration, Bzip2, LZMA, and ZLIB, the proposed method, ELC-ARES, provides an average encoding runtime improvement of 123.42%, 1031.73%, and 26.77%, respectively.

LLC-ARES provides an encoding runtime smaller with 46% compared with ELC-ARES. Its runtime results are not included in the comparison as it employs only low-complexity techniques to be suitable for ESP integration, while all other codecs are designed for SoC integration.

4. Conclusions

The paper proposed a novel entropy coding-based method for encoding raw event data. ELC-ARES employs the event coding framework to rearranged the sequence and the TTP algorithm to generate a set of data element. The LLC-ARES performance was improved by modifying the TTP algorithm to employing entropy coding-based techniques to encode the set of data elements using a large set of order-0 AMMs. Novel prediction strategies were explored. A new threshold initialization was introduced. The experimental evaluation demonstrates that ELC-ARES provides a 21.4% improvement compared with LLC-ARES, and more than 28.03% compared with state-of-the-art data compression codecs. The paper explored the prospect of designing novel lossless compression codecs for asynchronous event sequences using traditional entropy coding techniques.

References

[1] DSEC dataset. https://dsec.ifi.uzh.ch/dsec-datasets/download/. Accessed: 2021-10-01. 5

[2] Raymond Baldwin, Ruixu Liu, Mohammed Mutlaq Almatrafi, Vijayan K Asari, and Keigo Hirakawa. Time-ordered recent event (tore) volumes for event cameras. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1, 2022. 2

[3] Srutarshi Banerjee, Zihao W. Wang, Henry H. Chopp, Oliver Cossairt, and Aggelos K. Katsaggelos. Lossy event compression based on image-derived quad trees and Poisson disk sampling. In *2021 IEEE International Conference on Image Processing (ICIP)*, pages 2154–2158, 2021. 2

[4] Zhichao Bi, Siwei Dong, Yonghong Tian, and Tiejun Huang. Spike coding for dynamic vision sensors. In *2018 Data Compression Conference*, pages 117–126, 2018. 1

[5] Michael Burrows and David J. Wheeler. A block-sorting lossless data compression algorithm. 1994. 2

[6] P. Deutsch and J.-L. Gailly. Zlib compressed data format specification version 3.3. *IETF, RFC 1950*, May 1996. 2, 5

[7] Siwei Dong, Zhichao Bi, Yonghong Tian, and Tiejun Huang. Spike coding for dynamic vision sensor in intelligent driving. *IEEE Internet of Things Journal*, 6(1):60–71, 2019. 1

[8] P. Elias. Universal codeword sets and representations of the integers. *IEEE Transactions on Information Theory*, 21(2):194–203, 1975. 2, 4

[9] Thomas Finateu, Atsumi Niwa, Daniel Matolin, Koya Tsuchimoto, Andrea Mascheroni, Etienne Reynaud, Pooria Mostafalu, Frederick Brady, Ludovic Chotard, Florian LeGoff, Hirotsugu Takahashi, Hayato Wakabayashi, Yusuke Oike, and Christoph Posch. 5.10 a 1280×720 back-illuminated stacked temporal contrast event-based vision sensor with $4.86\mu m$ pixels, 1.066geps readout, programmable event-rate controller and compressive data-formatting pipeline. In *2020 IEEE International Solid- State Circuits Conference - (ISSCC)*, pages 112–114, 2020. 1

[10] Guillermo Gallego, Tobi Delbrück, Garrick Orchard, Chiara Bartolozzi, Brian Taba, Andrea Censi, Stefan Leutenegger, Andrew J. Davison, Jörg Conradt, Kostas Daniilidis, and Davide Scaramuzza. Event-based vision: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(1):154–180, 2022. 1

[11] Mathias Gehrig, Willem Aarents, Daniel Gehrig, and Davide Scaramuzza. Dsec: A stereo event camera dataset for driving scenarios. *IEEE Robotics and Automation Letters*, 6(3):4947–4954, 2021. 5

[12] Nabeel Khan, Khurram Iqbal, and Maria G. Martini. Lossless compression of data from static and mobile dynamic vision sensors-performance and trade-offs. *IEEE Access*, 8:103149–103163, 2020. 2

[13] Nabeel Khan, Khurram Iqbal, and Maria G. Martini. Time-aggregation-based lossless video encoding for neuromorphic vision sensor data. *IEEE Internet of Things Journal*, 8(1):596–609, 2021. 2

[14] Pierre Simon Laplace. *Essai philosophique sur les probabilités*. Courcier, 1814. 3

[15] Patrick Lichtsteiner, Christoph Posch, and Tobi Delbruck. A $128\times$ 128 120 db 15 μs latency asynchronous temporal contrast vision sensor. *IEEE Journal of Solid-State Circuits*, 43(2):566–576, 2008. 1

[16] Ana I. Maqueda, Antonio Loquercio, Guillermo Gallego, Narciso Garcia, and Davide Scaramuzza. Event-based vision meets deep learning on steering prediction for self-driving cars. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. IEEE, Jun. 2018. 2

[17] Maria Martini, Jayasingam Adhuran, and Nabeel Khan. Lossless compression of neuromorphic vision sensor data based on point cloud representation. *IEEE Access*, 10:121352–121364, 2022. 2

[18] I. Pavlov. LZMA SDK (Software Development Kit). https://www.7-zip.org/. Accessed: 2021-07-19. 2, 5

[19] Ionut Schiopu and Radu Ciprian Bilcu. Lossless compression of event camera frames. *IEEE Signal Processing Letters*, 29:1779–1783, 2022. 2

[20] Ionut Schiopu and Radu Ciprian Bilcu. Low-complexity lossless coding for memory-efficient representation of event camera frames. *IEEE Sensors Letters*, 6(11):1–4, 2022. 2

[21] Ionut Schiopu and Radu Ciprian Bilcu. Low-complexity lossless coding of asynchronous event sequences for low-power chip integration. *Sensors*, 22(24), 2022. 1, 2, 5

[22] J. Seward. bzip2 pre-build binaries. http://gnuwin32.sourceforge.net/packages/bzip2.htm. Accessed: 2021-07-19. 2, 5

[23] Shintaro Shiba, Yoshimitsu Aoki, and Guillermo Gallego. Event collapse in contrast maximization frameworks. *Sensors*, 22(14), 2022. 1

[24] Shintaro Shiba, Yoshimitsu Aoki, and Guillermo Gallego. Secrets of event-based optical flow. In *European Conference on Computer Vision (ECCV)*, pages 628–645, 2022. 1

[25] Yunjae Suh, Seungnam Choi, Masamichi Ito, Jeongseok Kim, Youngho Lee, Jongseok Seo, Heejae Jung, Dong-Hee Yeo, Seol Namgung, Jongwoo Bong, Sehoon Yoo, Seung-Hun Shin, Doowon Kwon, Pilkyu Kang, Seokho Kim, Hoonjoo Na, Kihyun Hwang, Changwoo Shin, Jun-Seok Kim, Paul K. J. Park, Joonseok Kim, Hyunsurk Ryu, and Yongin Park. A 1280×960 dynamic vision sensor with a $4.95 - \mu m$ pixel pitch and motion artifact minimization. In *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–5, 2020. 1

[26] Gary J. Sullivan, Jens-Rainer Ohm, Woo-Jin Han, and Thomas Wiegand. Overview of the high efficiency video coding (hevc) standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 22(12):1649–1668, 2012. 2

[27] S. M. Nadim Uddin, Soikat Hasan Ahmed, and Yong Ju Jung. Unsupervised deep event stereo for depth estimation. *IEEE Transactions on Circuits and Systems for Video Technology*, 32(11):7489–7504, 2022. 1

[28] G. Vollan. ZLIB pre-build DLL. http://www.winimage.com/zLibDll/. Accessed: 2021-07-19. 5

[29] Lin Wang, Yujeong Chae, Sung-Hoon Yoon, Tae-Kyun Kim, and Kuk-Jin Yoon. Evdistill: Asynchronous events to end-task learning via bidirectional reconstruction-guided cross-modal knowledge distillation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 608–619, June 2021. 1

[30] Ziwei Wang, Liyuan Pan, Yonhon Ng, Zheyu Zhuang, and Robert Mahony. Stereo hybrid event-frame (SHEF) cameras for 3d perception. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 9758–9764, 2021. 1

[31] Yajing Zheng, Zhaofei Yu, Song Wang, and Tiejun Huang. Spike-based motion estimation for object tracking through bio-inspired unsupervised learning. *IEEE Transactions on Image Processing*, 32:335–349, 2023. 1

[32] Yi Zhou, Guillermo Gallego, Xiuyuan Lu, Siqi Liu, and Shaojie Shen. Event-based motion segmentation with spatio-temporal graph cuts. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–13, 2021. 1

[33] A. Zhu, L. Yuan, K. Chaney, and K. Daniilidis. Unsupervised event-based learning of optical flow, depth, and egomotion. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 989–997, Los Alamitos, CA, USA, Jun. 2019. IEEE Computer Society. 2

[34] Lin Zhu, Siwei Dong, Tiejun Huang, and Yonghong Tian. Hybrid coding of spatiotemporal spike data for a bio-inspired camera. *IEEE Transactions on Circuits and Systems for Video Technology*, 31(7):2837–2851, 2021. 2

[35] Zhiyu Zhu, Junhui Hou, and Xianqiang Lyu. Learning graph-embedded key-event back-tracing for object tracking in event clouds. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022. 1