Entrophy 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 modeling 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 tasks 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.

### 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, $S_T = \{e_i\}, i=1, N_e$, stores all $N_e$ events triggered during $T$ timestamps using the EE representation as a sequence $e_{N_e}^T = e_1 e_2 \ldots e_{N_e}$. The ST representation rearranges $S_T$ as a set of ST subsequences, $S_T = \{S_k\}, k=0, T-1$, 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., $y_i^k$, and then of the remaining dimension ($x_i^k$) if $e_{y_i^k} = y_i^k - y_i^k-1 = 0$. ELC-ARES encodes $S_T$ as a set of $T$ subsequence $(S_k)$, each represented using four DSs: $N_e^k, \{y_i^k\}, \{x_i^k\}, \{p_i^k\}$, see Fig. 1.
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 $\Delta = (\delta_1, \delta_2, \delta_3)$. To encode each $S_k$, three TTP versions were proposed based on the type of encoded ST DS: TTP$_x$ to encode $N_e$, TTP$_y$ to encode $\{y_k\}_{i=1}^{N_k}$, and TTP$_z$ to encode $\{x_k\}_{i=1}^{N_k}$.

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 ($b_0b_1b_2b_3$, or $b_0$); and (b) the binary representation of the residual error, $\varepsilon = x - \hat{x}$, using $n_{\delta_1}$ 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$_z$ 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 $x^n = x_0x_1 \ldots x_{n-1}$, a sequence of $n$ elements $x_j$ selected from an alphabet $\mathcal{A}_K = \{s_0, s_1, \ldots, s_{\kappa - 1}\}$ of $\kappa$ symbols, and $C_K = \{c_0c_1 \ldots c_{\kappa - 1}\}$, where $c_i$ is the frequency of symbol $s_i$. The sequence $x^n$ is encoded using the probability distribution $P_n = [p_0 p_1 \ldots p_{n-1}]$, where $p_i = \frac{c_i}{C_K}$. 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 prob-

![Figure 2](image-url)
ability that the next element \(x_n\) is symbol \(s_i\) as follows:

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

An order \(N\) AMM uses \(\kappa^N\) contexts of previous \(N\) symbols, where \(x_n\) is encoded given an alphabet \(A_n\). 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 \(\{x_{k+1}^{N_k}\}_{i=1}^{N_k}\), and several models for the data elements generated by \(TTP_e, TTP_y,\) and \(TTP_x\) for \(N_k\), \(\{y^{k}|N_t\}_{i=1}^{N_t}\), and \(\{x^{k+1}|N_r\}_{i=1}^{N_r}\), respectively. Six AMM types are distinguished.

(1) Decision Tree (DT), for encoding the decision tree of: \(TTP_e (DTx)\), see Fig. 2a; \(TTP_y (D’T)\), see Fig. 2b; and \(TTP_x (D’Tx)\), see Fig. 2c. E.g., the modified \(TTP_e\) algorithm encodes the following data elements: \(b_i, \forall i = 0, 1, 2, 3\) to signal R1-R3; \(b_i\) to signal R4-R5; and \(b_i\) to signal \(\text{sgn}(\epsilon_i)\). All \(DT\) models use alphabet \(A_2 = \{0, 1\}.

(2) Encode \(x (Ex)\), for encoding \(|\epsilon_x| = |x - \hat{x}|\) using: range R1, where model \(Ex\ell_1 \ell\) (where \(\ell\) depends on \(\delta_i\) value) encodes \(x_n = |\epsilon_x|\) using \(A_{\delta_i} = \{0, 1, 1, \delta_i - 1\}\); range R2, where model \(Ex\ell_2\ell\) encodes \(x_n = |\epsilon_x| - \delta_i\) using \(A_{\delta_i}\); and range R3, where model \(Ex\ell_3\ell\) encodes \(x_n = |\epsilon_x| - \delta_i - \delta_k\) using \(A_{\delta_i}\), 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 \(Ey\ell_1\ell, Ey\ell_2\ell,\) and \(Ey\ell_3\ell\), respectively, see Fig. 2b.

(4) Encode \(e (Ee)\), for encoding \(|\epsilon_e| = |N_k - N_k|\) similarly as \(Ex\), i.e., by using one of R1 and R2 ranges of \(TTP_e\) with model \(E\ell_1\) and \(E\ell_2\), respectively, see Fig. 2c.

(5) Polarity (P), for encoding \(p_i, \forall i = 1, 2, \ldots, N_k\) using the model \(P1\), having \(A_2 = \{-1, 1\}.

(6) Binary (B), for the binary representation of: \(\text{Fig. 6}\)

\[
(x - 1|10) = \overline{b_{n-1}} \ldots \overline{b_0} (2) \text{ in } R4 \text{ of } TTP_x \text{ by encoding } x_n = b_{n-1} \text{ using model } BR4 \text{ and } x_{n+i} = b_i, \forall i = 0, 1, \ldots, n - 2 \text{ using model } BXL; \) \(\text{Fig. 6}\)

\[
(W - y|10) = \overline{b_{n-1}} \ldots \overline{b_0} (2) \text{ in } R5 \text{ of } TTP_y \text{ by encoding } x_n = b_{n-1} \text{ using model } BR5 \text{ and } x_{n+i} = b_i, \forall i = 0, 1, \ldots, n - 2 \text{ using model } BYL.
\]

Table 1 lists all the order \(-0\) AMMs introduced by ELC-ARES, i.e., 33 models, together with their ID and alphabet size. Sets of 12, 1, 2, and 4 models are introduced for the \(DT, Ee, P,\) and \(B\) types, respectively, for encoding the DSS generated by \(TTP_e, TTP_y,\) and \(TTP_x\). 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 \(\Delta\), see Sec. 2.5.

In R6 of \(TTP_e\), large \(|\epsilon_e|\) are codified by employing Elias Gamma Coding (EGC) [8] for \(x_e = |\epsilon_e| - \Delta_e - 2\), 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 \(x_e\) number of bits; \(BR5\) for the remaining \(x_{\gamma}\) binary digits, see Sec. 2.6.

2.4. Exploring New Prediction Strategies

In LLC-ARES, prediction \(y^k_i\) is fixed to \(y^k_{i-1}\) due to the ST representation, while \(N_k\) 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, \(x^k_i\), is computed using different strategies using \(\{x^k_i\}_{j=1}^{15}\). 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 \(i > 2, x^k_i = \text{med}(|x^k_{i-j}|_{j=1:50,i>2})\). Secondly, let us denote \(Pxx1\) the prediction strategy where \(TTP\) is in charge of handling large prediction errors (using \(R4-R6\)) and computation complexity is minimized as \(x^k_i = x^k_x\), 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 \(\Delta_{H+1}\), see Eq. (2). Since \(Pxx1\) 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).

\[
\Delta_{H,T345}^{k+1} = \left\{ \begin{array}{ll} \Delta^{c1} = (2^3, 2^4, 2^5) & \text{if } k = 0; \\ (2^5, 2^6, 2^6) & \text{if } k > 0 \& k^6 < 8; \\ (2^4, 2^4, 2^5) & \text{otherwise}. \end{array} \right.
\]

\[
\Delta_{H,T455}^{k+1} = \left\{ \begin{array}{ll} \Delta^{c1} = (2^3, 2^4, 2^5) & \text{if } k = 0; \\ (2^6, 2^6, 2^6) & \text{if } k > 0 \& k^6 < 8; \\ (2^5, 2^5, 2^5) & \text{otherwise}. \end{array} \right.
\]

\(T345\) may set \(\Delta_{H}^{k+1}\) as follows: \(\delta_1 = 2^3, 2^4,\) or \(2^5\) for models \(Ex\delta_1, Ex\delta_2,\) or \(Ex\delta_3,\) respectively; \(\delta_2^6 = 2^4,\) or \(2^5\) for models \(Ex\delta_1, Ex\delta_2,\) or \(Ex\delta_3,\) respectively; \(\delta_3 = 2^5,\) or \(2^6\) for models \(Ex\delta_1, Ex\delta_2,\) respectively.

\(T455\) may set \(\Delta_{H}^{k+1}\) as follows: \(\delta_1 = 2^4, 2^5,\) or \(2^6\) for models \(Ex\delta_1, Ex\delta_2,\) or \(Ex\delta_3,\) respectively; \(\delta_2 = 2^5,\) or \(2^6\) for models \(Ex\delta_1, Ex\delta_2,\) respectively; and \(\delta_3 = 2^5,\) or \(2^6\) for models \(Ex\delta_1, Ex\delta_2,\) respectively.

\(\Delta-W\) may be set as follows: \(\delta_1 = 2^1, 2^2,\) or \(2^3\) for models \(Ey\delta_1, Ey\delta_2,\) respectively; \(\delta_2 = 2^1, 2^2,\) or \(2^3\) for models \(Ey\delta_1, Ey\delta_2,\) or \(Ey\delta_3,\) respectively; and \(\delta_3 = 2^2,\) or \(2^3\) for models \(Ey\delta_1, Ey\delta_2,\) or \(Ey\delta_3,\) respectively.
Algorithm 1: ELC-ARES Encoder of $S^k$

Input: $N_{e}^k$, $\{y_{i}^{k}\}_{i=1}^{N_{e}^k}$, $\{x_{i}^{k}\}_{i=1}^{N_{e}^k}$, $\{p_{i}^{k}\}_{i=1}^{N_{e}^k}$, $\{N_{i}^{k}\}_{i=1}^{k-3}$; $H, W, P_x$ (prediction strategy), and $T_{p}$ (initialization);
1 Initialize all models listed in Tab. 1;
2 $N_{e}^k \leftarrow$ Predict $N_{e}^k$ using $\{N_{i}^{k}\}_{i=k-3}^{3}$;
3 Encode $N_{e}^k$ using Alg. 1 ($N_{e}^k$, $N_{e}^k$, $\Delta$);
4 if $N_{e}^k > 0$ then
5 $y_{i}^k \leftarrow$ Encode Alg. 4 ($y_{i}^{k}$, $y_{i}^{k-1}$, $\epsilon_{i}^{k}$, $[1, W]$), $\Delta^{(1)}$;
6 $x_{i}^k \leftarrow$ Encode Alg. 4 ($x_{i}^{k}$, $x_{i}^{k-1}$, $\epsilon_{i}^{k}$, $[1, H]$), $\Delta^{(1)}$;
7 $p_{i}^k \leftarrow$ Encode using model $P_1$; $\Delta^{(1)}$;
8 for $i = 2, 3, \ldots, N_{e}^k$ do
9 $y_{i}^{k} \leftarrow$ Alg. E3 ($y_{i}^{k}$, $y_{i-1}^{k}$, $[y_{i-1}^{k}, W]$), $\Delta_{W}^{k}$;
10 $x_{i}^{k} \leftarrow$ Alg. E4 ($x_{i}^{k}$, $x_{i-1}^{k}$, $[y_{i-1}^{k}, H]$), $\Delta_{H}^{k}$;
11 $p_{i}^{k} \leftarrow$ Alg. E5 ($p_{i}^{k}$, $p_{i-1}^{k}$), $\Delta_{H}^{k}$;
12 $\Delta_{H}^{k+1} \leftarrow$ Update $\Delta_{H}^{k}$ using $e^{k} = y_{k}^{N_{e}^k} - y_{k}^{1}$ and $T_{x}$;
13 $\Delta_{W}^{k+1} \leftarrow$ Update $\Delta_{W}^{k}$ using $e^{k} = y_{k}^{N_{e}^k} - y_{k}^{1}$;
14 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 $\Delta^{k+1}$ initialization, see Sec. 2.5. For pseudocode of Alg. E1-E4 is presented in the supplementary material file. Alg. E1, E2, and E4 present the modified $TTP_{e}$ (see Fig. 2c). $TTP_{y}$ (Fig. 2b), and $TTP_{x}$ (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 Alg. D1-D4 is presented in the supplementary material file. Alg. D1, D2, and D4 present the modified $TTP_{e}$ (see Fig. 2a). $TTP_{y}$ (see Fig. 2c), and $TTP_{x}$ (Fig. 2e) decoder, while Alg. 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 (Mspev), number of events encoded per second; (e) Runtime (uspev), 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

ELC-ARES is designed to employ the $w5$ prediction strategy and the T455 initialization. Here, ELC-ARES is designed to employ the $P21$ 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 $w50$. 

3927
Fig. 3. Compression results of ELC-ARES versions. Sequences are sorted in ascending order of event acquisition density.

![Compression Ratio Results](image)

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

<table>
<thead>
<tr>
<th>Codec</th>
<th>$P_x$</th>
<th>$T_x$</th>
<th>CR</th>
<th>RC$_{LLC}$</th>
<th>RC$_{ELC}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>LLC-ARES</td>
<td>w5</td>
<td>T345</td>
<td>4.371</td>
<td>1</td>
<td>0.824</td>
</tr>
<tr>
<td>ELC-ARES-v1</td>
<td>w5</td>
<td>T345</td>
<td>5.256</td>
<td>1.202</td>
<td>0.991</td>
</tr>
<tr>
<td>ELC-ARES-v2</td>
<td>w50</td>
<td>T455</td>
<td>5.261</td>
<td>1.204</td>
<td>0.992</td>
</tr>
<tr>
<td>ELC-ARES-v3</td>
<td>w50</td>
<td>T345</td>
<td>5.282</td>
<td>1.208</td>
<td>0.996</td>
</tr>
<tr>
<td>ELC-ARES-v4</td>
<td>Px1</td>
<td>T345</td>
<td>5.286</td>
<td>1.209</td>
<td>0.996</td>
</tr>
<tr>
<td>ELC-ARES</td>
<td>Px1</td>
<td>T455</td>
<td>5.306</td>
<td>1.214</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 3. Average coding performance over DSEC

<table>
<thead>
<tr>
<th>Codec</th>
<th>ZLIB</th>
<th>LZMA</th>
<th>Bzip2</th>
<th>LLC-ARES</th>
<th>ELC-ARES</th>
</tr>
</thead>
<tbody>
<tr>
<td>BR</td>
<td>20.32</td>
<td>16.80</td>
<td>15.91</td>
<td>14.82</td>
<td>12.58</td>
</tr>
</tbody>
</table>

Fig. 4 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 provide 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.

**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-
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.

### 4. Conclusions

The paper proposed a novel entropy coding-based method for encoding raw event data. ELC-ARES employs the event coding framework to rearrange the sequence and the TTP algorithm to generate a set of data elements. The LLC-ARES performance was improved by modifying the TTP algorithm to employing entropy encoding-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


