Lossy Event Compression based on Image-derived Quad Trees and Poisson Disk Sampling

Srutarshi Banerjee  Zihao W. Wang  Henry H. Chopp, Oliver Cossairt  Aggelos K. Katsaggelos
Northwestern University
srutarshibanerjee2022@u.northwestern.edu

Abstract

With several advantages over conventional RGB cameras, event cameras have provided new opportunities for tackling visual tasks under challenging scenarios with fast motion, high dynamic range, and/or power constraint. Yet unlike image/video compression, the performance of event compression algorithm is far from satisfying and practical. The main challenge for compressing events is the unique event data form, i.e., a stream of asynchronously fired event tuples each encoding the 2D spatial location, timestamp, and polarity (denoting an increase or decrease in brightness). Since events only encode temporal variations, they lack spatial structure which is crucial for compression. To address this problem, we propose a novel event compression algorithm based on a quad tree (QT) segmentation map derived from the adjacent intensity images. The QT informs 2D spatial priority within the 3D space-time volume. In the event encoding step, events are first aggregated over time to form polarity-based event histograms. The histograms are then variably sampled via Poisson Disk Sampling prioritized by the QT based segmentation map. Next, differential encoding and run length encoding are employed for encoding the spatial and polarity information of the sampled events, respectively, followed by Huffman encoding to produce the final encoded events. Our Poisson Disk Sampling based Lossy Event Compression (PDS-LEC) algorithm performs rate-distortion based optimal allocation. On average, our algorithm achieves greater than 6x higher compression compared to the state of the art.

1. Introduction

Inspired by biological visual systems, event cameras are novel sensors designed to capture visual information with a data form drastically different from traditional images and videos [38, 27]. The event pixels do not directly output the intensity signals as traditional cameras do. Instead, each pixel compares the difference between the current log-intensity state and the previous state, and fires an event when the difference exceeds the firing positive or negative thresholds. This sensing mechanism provides several benefits. First, event pixels operate independently which enables very low latency (∼ 10µs) and therefore high speed imaging. Second, event cameras have high dynamic range (HDR, ∼ 120dB) compared to regular frame-based cameras (∼ 60dB). Third, the events reduce redundant captures of static signals. Last, event cameras consume lower power (10mW) than traditional cameras (∼ 1W). As such, event cameras have brought new solutions to many classical as well as novel problems in computer vision and robotics, including high frame-rate video reconstruction [46, 34, 37], with HDR [43, 33] and high resolution [44, 28, 45], and 3D reconstruction of human motion [49] and scenes [31, 23], as well as odometry [11, 41] and tracking [52, 24].

Currently, events are mainly communicated in the raw format using the Address Event Representation (AER) protocol [1]. The current AER protocol, AEDAT 4.0, released in July 2019, [1] uses 96 bits representation for each event tuple (x, y, t, p) which are (x, y) position, timestamp and polarity, while its earlier version, AEDAT 3.1, uses 64 bit representation for each event. The timestamp uses the most bits due to its resolution with 64 bits and 32 bits for AEDAT 4.0 and AEDAT 3.1, respectively. Although AEDAT 4.0 has incorporated lossless encoding standards such as LZ4, LZ4_HIGH, ZSTD and ZSTD_HIGH [1], effective lossy event encoding has not been proposed in the literature or implemented in the event cameras.

In traditional image/video compression standards, lossy compression is achieved by exploiting the spatial and temporal correlations. However, events are discrete points scattered in the space-time volume (see, for example Fig. 1 left). Several prior works have approached event compression [8, 16, 21]. TALVEN [21] has aimed at aggregating events based on event timestamps. While this improves the compression ratio (CR), the benefits of high compression is only evident when aggregating events over a long time du-
Our goal is to design an efficient event compression algorithm that exploits the spatio-temporal redundancy of events without sacrificing temporal information. Our approach considers the hybrid input of RGB images and events, which is a context largely existing in the literature [53, 29, 45, 41]. We reason that the adjacent in time images have rich spatial features that can be leveraged to guide event compression. Therefore, we first compute a quad tree (QT) structure to serve as the priority map. Note that the QT can be generated from events as well. Next, the events are sampled to generate a blue noise distribution locally with the QT defining regions of importance, albeit in 2D space. The blue noise pattern of events is generated by Poisson Disk Sampling (PDS) [10] which randomly samples events. The overall framework is shown in Fig. 1. In particular, this paper makes the following contributions:

(1) We propose a novel algorithm for lossy compression of neuromorphic events based on the QT segmentation map from adjacent images exploiting spatio-temporal redundancy.

(2) We evaluate our approach qualitatively and quantitatively, on existing event datasets and compare our results with state of art and other techniques.

2. Related Work

Conventional image and video based coding techniques have evolved over the last couple of decades which led to a successful telecom revolution globally. Several standards came out as a result of the persistent effort, some of the latest video standards being HEVC, H.264, AVS2, VP9 [39, 48, 19, 30]. With images and videos, the 2D and/or 3D spatio-temporal consistency of the data is exploited to have an efficient coding algorithm. However, events are asynchronous and non-continuous in space and time. Limited work [16, 8] has been done in predicting the distribution of events in space, time and polarity.

2.1. DVS Coding Approaches

The approaches for compressing DVS data can be classified into two categories: (a) specifically encoding events, (b) applying existing methods to events.

A. DVS Specific Coding

(1) Spike Coding: Lossless event compression was first proposed in [8]. The encoding was derived from a spike firing model exploiting spatial and temporal correlation. The events are projected into a sequence of three-dimensional macro-cubes and encoded using Address-Prior (AP) and Time-Prior (TP) modes. The AP mode is designed for scenarios where events are scattered all over the 2D space, while the TP mode is designed for cases where events occur locally. The residuals are encoded using arithmetic coding (CABAC). The authors extended this algorithm to include intercube prediction utilizing the temporal correlation among microcubes [16], although the compression ratio did not improve significantly.

(2) TALVEN: The time aggregation strategy based lossless video encoding strategy was devised in [21]. The events are time-aggregated into an event frame with a histogram count at each pixel location where each event frame is separated based on event polarities. The concatenated frames are coded using conventional lossless video encoding techniques including interframe coding, intraframe coding and entropy coding (CABAC) with the data packets incorporated into Network Abstraction Layer (NAL) units. The CR of this technique is significantly higher than the Spike Coding especially at higher time aggregation.

B. Existing compression methods for Events

The application of existing compression methods tailored to events can be broadly categorized into:

(1) Entropy coding - Entropy coders like Huffman and arithmetic coding can potentially be applied to events [18] considering each event field \((x, y, t, p)\) as separate symbols. The gains of using just this technique are rather limited.

(2) Dictionary based compression - A collection of long strings and shorter codewords is maintained in a dictionary data structure. Advanced dictionary coders Zstd [2], Zlib

(3) IoT specific compression - An Internet of Things (IoT)-specific strategy has been devised for resource constrained devices [9] without violating memory and latency constraints of the IoT devices. Sprintz exploits correlations among successive samples of a multivariate stream. The event stream can be directly converted to a multivariate time series data for compression. This strategy works better when it is required to have higher compression gains with reduced power consumption [18].

(4) Fast integer compression - In order to boost up the encoding and decoding speed (e.g. billions of arrays of integers for search engines), fast integer compression is often the choice. Simple8B [9], Mempy [9], SIMD-BP128 [26], FastPFOR [26] and SNAPPY [4] are some of the fast integer compression algorithms. When DVS data needs a rapid data transfer, fast integer compression can be applied by transforming events into vectors.

A comparison of these approaches has been done in [20, 21] resulting in low compression ratios between 1 and 4. The aforementioned approaches do not consider spatio-temporal encoding of events at the same time.

2.2. Blue Noise Sampling

Blue noise based sampling is widely applied in rendering, texturing, animation and related domains. Blue noise captures the local density of the image through its local point density. It has isotropic properties which lead to high-quality sampling of multidimensional signals without any aliasing [25]. Blue noise distribution has been generated over the years by several techniques—error diffusion [40, 17], dart throwing algorithm [12]—to create Poisson disk distributions and other techniques. Poisson disk algorithms that generate blue noise samples [50, 13, 47, 42, 51, 15] by reducing the computational complexity and improving the blue noise characteristics are widely studied and applied in the field of computer graphics.

We leverage a fast Poisson Disk Sampling (PDS) technique as a method to sample events in 2D space. This generates a blue noise pattern preserving the high frequency spectral components of the events. We compare PDS with random sampling in Section 6 and show that using PDS results in overall higher compression ratio and better event compression metrics.

3. Spatio-Temporal Event Compression

Mathematical analysis in [8, 16] points out the correlation of DVS spikes in both space and time. The adjacent pixels receive the same illumination change (increase or decrease) over time. This leads to spatial as well as temporal correlation in local regions. We exploit this fundamental nature of the DVS spikes for encoding events. First, we generate a quad tree (QT) structure from the adjacent intensity frame which indicates spatial priority regions. Depending on the block sizes, we develop a strategy based on Poisson disk sampling for removing events based on spatio-temporal proximity to each other. The events are encoded based on block sizes using entropy encoding strategies. In the following subsections, the framework is described in more details.

3.1. Generation of Quadtree (QT) structure

The events are generated due to scene complexity (edges/textures of the objects in scene) and (or) relative motion of the camera with respect to the scene. Khan et al. [22] showed that the event rate depends exponentially on the scene complexity metric and linearly on the sensor speed metric. However, the scene complexity provides us with the important events pertaining to objects.

We rely on the intensity frames to generate the priority regions for the events. Although priority can be generated in different ways, in this work we use the QT data structure in order to divide the frame into blocks. The QT can subdivide the frames into different levels resulting in heterogeneous blocks of sizes determined by the leaf level of the QT. Larger block sizes indicate lower priority regions while smaller blocks indicate higher priority. A time aggregated event stream between two successive intensity frames can be approximated by a difference of these intensity frames. Hence, the priority regions for events is proportional to the difference between these intensity frames. We generate the QT based on the system described in [7]. For an event volume $E_{t-1}$, between successive intensity frames $I_{t-1}$ and $I_t$ (both as defined in [7]), we leverage on these intensity frames to derive a relevant QT structure. The QT is generated using Dynamic Programming (Viterbi optimization [36]) which provides a trade-off between the frame distortion ($D$) and frame bit rate ($R_t$). This is done by minimizing $D$ over the leaves of the QT, denoted by $x$, subject to a given maximum bit rate $R_{max}$. In other words, we are solving for the rate-distortion tradeoff for the intensity frames. The constrained discrete optimization is solved using Lagrangian relaxation, leading to solutions in the convex hull of the rate-distortion curve. The Lagrangian cost function is of the form

$$J_\lambda(x) = D(x) + \lambda R(x),$$

where $\lambda \geq 0$ is a Lagrangian multiplier. The intensity bit rate is set to be generated for a fixed rate (within a tolerance) constraint. The QT is generated such that the intensity bit rate satisfies this constraint. $\lambda$ is adjusted at each frame for achieving the desired bit rate. The optimal $x^*$ is computed by a convex search in the Bezier curve [36] over the rate-distortion space which results in convergence in fewer iterations. The optimal QT segmentation map corresponds
to $\lambda^*$ and at a set bit rate, satisfies equation (S1), derived for intensity frame $I_t$ based on the distorted frame $I_{t-1}$. Clearly, at higher bit rates the QT can go to deeper levels (and hence smaller blocks) while for smaller bit rates, the QT branches out to shallow levels. This 2D segmentation map is applied to the event volume $E_{t-1}$ providing regions of priority for event compression, with block size inversely proportional to the priority in the block.

### 3.2. Poisson Disk Sampling

The QT structure provides a segmentation map. Since the events are correlated locally in space and time, there exists local redundancy not only in the temporal dimension but also in the spatial dimensions. The local region in space is quite abstract as it may vary from scene to scene and also within a scene. In this work, the QT block sizes indicate the importance of the regions, which is also an indication of the dimensions of local regions in space-time. There could be local features in bigger blocks, but these are not of high priority based on the set bit rate on intensity frames.

We perform lossy compression on events not only by sampling the events from the QT blocks, but also temporally aggregating the events. Sampling of events inside the QT blocks is challenging due to the presence of both local structure and noise. There may be aggregation of events at object edges, which may be increased due to presence of noise in the sensor. The events are sampled as a blue noise pattern. Blue noise pattern picks up the local event density without adding any inherent artificial structures. This leads to high-quality sampling of multi-dimensional signals while preventing aliasing. In this work, PDS is used to generate blue noise.

PDS is applied on the events inside the QT blocks. PDS sample events, with all events at least $r$ distance apart from each other. Bridson [10] proposed a fast PDS strategy in multidimensional space. However, the algorithm would generate points in space based on the PDS. In our work, we sample events based on this PDS technique. The resulting sampled events are generated with at least $r$ distance apart from each other. Given $M$ original events, we sample $R_e$ events, where $R_e = M$. PDS can be applied on the whole event volume or on regions of the events. Since the QT already provides us with a priority map, PDS is done with different $r$ on blocks of different sizes in the QT.

PDS needs a reference position to start sampling the events. In a QT block, it is difficult to identify a reference starting point for PDS. The reference starting point can be conveniently in any of the corner locations in the QT block (or any other location in the QT block). However, this might lead to a reference point which may not have an event. In order to eliminate any such issues, we consider the reference point as the location of the event at or nearest to the centroid (geometric median) with respect to the event locations in a QT block. Thus, in a neighborhood of $N \times N$ pixels, we find the geometric median $x_m$, of the $M$ events, as shown in Eqn. (2), where each $x_i \in \mathbb{R}^n$ are the event locations in space-time.

$$x_m = \arg \min_{x_m} \sum_{i=1}^{M} \|x_i - x_m\|_2,$$

Except for the chosen point, we do not sample the events lying within a disk of radius $r$, with respect to this reference point. Next, we shift our reference point to the nearest event location outside the radius $r$ and do not sample the events lying within radius $r$ (termed as Poisson Disk Radius (PDR)) with respect to this new reference point. This step is repeated until all the events are at least or greater than a spatial distance $r$ from each other. This sampling strategy has the effect of sampling the event density locally in the region of $N \times N$ pixels, along with introducing a fixed spatial sampling pattern in the event volume. The sampling approach not only reduces the density of the aggregated events near the edges but can also remove the uncorrelated noisy events from local regions in space.

### 3.3. Lossy Event Compression

The QT provides the priority map and PDS provides an efficient strategy to reduce the density of events and noise from the events. However, the QT has different sized blocks which indicate different priorities. We want to sample events based on priority as this will retain local features in high priority regions and remove local features from lower priority regions. This is achieved by using different Poisson disk radius $r$ values for different QT block sizes: higher and smaller $r$ values for bigger and smaller QT blocks, respectively. In our algorithm, we specifically apply PDS to blocks larger than $2 \times 2$. For QT block sizes below $4 \times 4$, we do not sample events as we consider those blocks to be regions of very high priority and hence all events in those regions are critical. As mentioned in subsection 3.1, we obtain the QT for event volume $E_{t-1}$ from reconstructed intensity frame $I_{t-1}$ and actual frame $I_t$.

The lossy event compression is performed in steps depending on the bandwidth available for communication between network-connected IoT devices. The lossy event compression framework has a lossy part as well as a loss-less part. The lossy portion of event encoding consists of 2 parts: (a) quantizing (aggregating) event timestamps as $T_{bin}$ and (b) PDS of the events with spatial overlap of event $(x, y)$ locations within the QT blocks. In step (a), we compute the histogram for the positive and negative events separately for each quantized timestamp. The lossless portion of the event encoding involves differentially encoding the $(x, y)$ location of events followed by Huffman coding. The polarity is encoded by Run Length Encoding (RLE) fol-
Algorithm 1: Event Compression Pseudo Code

Input: Adjacent Intensity frames, Event Volume
set $\lambda_{\text{min}}, \lambda_{\text{max}}, R_{\text{max}}$;
while intensity bit rate $R > R_{\text{max}}$ do
  | adjust $\lambda$ to obtain desired $R$ as in Sec. 3.1
end
Result: Optimal QT
set $T_{\text{bin}}$ and $r_i$;
Aggregate events into event histogram frames
(positive and negative) based on $T_{\text{bin}}$;
while QT blocks $\in \{4 \times 4, 8 \times 8 \text{ and higher}\}$ do
  | compute the centroid $C$ of events in $(x, y)$;
  | Shift $C$ to nearest event location $P$ (if $C \neq P$);
  | if events left to be visited then
    | Remove events within radius $r_i$ from $P$;
    | set $P$ as nearest event outside $r_i$;
  | Encode $(x, y)$ differentially followed by Huffman coding for each block;
  | Encode event counts as Run Length Encoding each block followed by Huffman coding;
end
while QT blocks $\in \{2 \times 2\}$ do
  | Encode $(x, y)$ differentially each block;
  | Encode event counts as Run Length Encoding each block followed by Huffman coding;
end
while QT blocks $\in \{1 \text{ pixel}\}$ do
  | Consider all pixels in the frame;
  | Encode event counts as Run Length Encoding followed by Huffman coding;
end
Output: Compressed Event Volume

4. Experiments

4.1. Dataset

In order to show the benefits of the proposed lossy compression strategy, we use datasets available in the literature which have both intensity frames and events. Sequences

... from RGB-DAVIS dataset [45, 5] are used to show the compression performance in different settings for image reconstruction experiments. The Dynamic and Active-pixel Vision Sensor (DAVIS) dataset [29] is used to show a comparison of PDS-LEC with other state-of-the-art event compression algorithms. Shapes-6dof, Dynamic-6dof, Slider-depth, Outdoors Running sequence has been used in DAVIS dataset. Running1 and Running2 sequences has been derived from Outdoors Running as done in [21].

4.2. Event Compression Metric

(a) Event Distortion Metric: The compressed events essentially represent the original 3D spatio-temporal event volume in a quantized manner. However, the fidelity of the compressed events with respect to undistorted event volume is very important to understand the level of distortion. In this direction, to the best of our knowledge, there is no metric for computing event distortion and we are making a first attempt in quantizing the distortion. The event spatial location $(x, y)$ is not only important, but the timestamp is also a vital parameter for various event based processing algorithms. The distortion of the events in both the spatial and temporal aspects must be computed.

We separate the spatial and temporal fidelity of the distorted event volume with respect to the undistorted events in order to have a complete understanding of the role of different parameters in encoding events. For computing the spatial distortion, we aggregate the events over the time bins into an $(x, y)$ event image. The PSNR and SSIM metrics computed on the aggregated compressed and uncompressed event images are used for computing the spatial distortion. The temporal distortion is rather hard to quantify. We define a temporal error metric to quantify the quantization error in time as described in Eqn. 3:

$$T_{\text{error}} = \frac{1}{N_{\text{fe}}} \sum_{i=1}^{N_{\text{fe}}} \sqrt{\sum_{j} (T_{j,\text{org}} - T_{j,\text{quant}})^2},$$  \hspace{1cm} (3)

where $N_{\text{fe}}$ is the number of event volumes in a sequence, $T_{j,\text{org}}$ is the timestamp of the $j$th event in $i$th frame, and $T_{j,\text{quant}}$ is the quantized timestamp of $j$th event in a compressed $i$th frame. Based on the end application of events, a weighted distortion metric may be computed with weights $w_s$ and $w_t$ for spatial distortion $D_s$ and temporal distortion $D_t$, respectively, thereby prioritizing spatial and / or temporal distortions differently.

(b) End-to-End Event Compression Metric: We also compute the end-to-end compression ratio for the events [21], considering 64 bits for each event. The compression ratio is given by the ratio of uncompressed events and the encoded event bit representation.

$$E_{\text{comp}} = \frac{\text{Bits/event} \times N_{\text{events}}}{\gamma},$$  \hspace{1cm} (4)
where $\gamma$ is the compressed event bits.

5. Performance of proposed framework

Here we show a performance study of the proposed lossy event compression framework with respect to different compression parameters.

5.1. Performance with varying $T_{bin}$ only

In one of the experiments, the QT segmentation is not used. The events are not sampled based on Poisson disk sampling. However, the events are temporally aggregated into $N$ bins, with $N \in \{8, 16, 24\}$. To quantify the benefits of the temporal compression, we compute the compression ratio of the original uncompressed events as well as the compressed events. Table 1 shows the compression ratio and the average $T_{error}$ over the sequence. Clearly, it can be seen that with the increase in the number of $T_{bin}$, the compression ratio reduces along with the reduction of $T_{error}$. Fig. 2(a) shows the original events while Figs. 2(b), (c) and (d) show the variation of events in temporal space with 8, 16 and 24 quantized timestamps respectively. The SSIM in this case is 1, since all events are sampled.

5.2. Performance with varying Intensity Bit Rate

The QT is optimized for particular operational bit rate for intensities only. In these experiments, $r_1 = 1$, $r_8 = 2r_4$, $r_{16} = 3r_4$ and $r_{32} = 4r_4$ with $T_{bin} = 16$ is used. The performance table is shown in Table 2. As bit rate reduces, the PSNR and SSIM reduces, while $T_{error}$ and CR increases. This indicates bigger blocks in the QT for lower bit rates with considerably higher distortion. Sample event frames at varying intensity bit rate is shown in Fig. 2 (row 2).

5.3. Performance with varying Poisson Disk Radius and Temporal Binning

In this experiment, the performance of the event compression system is evaluated at a particular bit rate, but the Poisson disk radius $r_4$ is varied. As in section 5.2, $r_8 = 2r_4$, $r_{16} = 3r_4$ and $r_{32} = 4r_4$. The temporal binning of $T_{bin} = 16$ is used in this experiment. The PSNR and SSIM reduces as $r_4$ increases as shown in Table 3. The $T_{error}$ increases with an increase of $r_4$, indicating the increase of temporal distortion. The compression ratio also increases with an increase of $r_4$. Sample event images for the original and compressed events for the Indoor6 sequence (frame 150) is shown in Fig. 2 (row 3) at 100 kbps. With the increase in $r_4$, the event image becomes less dense. It is noted that by setting different values of $r_4$, we can obtain a target bit rate for the events at a cost of a particular distortion.

6. Comparative Performance measure with other benchmark strategies

The proposed PDS-LEC algorithm is compared with other techniques such as TALVEN [21] and Spike Coding.
Figure 2: Row 1: temporal binning comparison for Indoor9 (human walking); Row 2: bit rate comparison for Outdoor3 (building) with \( r_4 = 1 \); Row 3: \( r_4 \) comparison for Indoor6 (hand gesture).

Figure 3: Compression Ratio of PDS-LEC (our work) at 100 kbps, \( r_4 = 1, 2 \), with TALVEN [21], Spike Coding [16], random (random = 0.5) and CT (thresh = 0.25, 0.30, 0.35) techniques

Figure 4: Compression Ratio of PDS-LEC (our work) at 300 kbps, \( r_4 = 1, 2 \), with TALVEN [21], Spike Coding [16], random (random = 0.5) and CT (thresh = 0.25, 0.30, 0.35) techniques
SSIM and T
bit rates and/or lower PDR since less events are removed. Higher than PDS-LEC, especially at QT derived at higher temporal aggregation, in a few experiments, CR is for 50% random events (using our compression scheme), PDS-LEC has better CR. However, Even for fewer events generated with higher CT (using our algorithms for different bit rates and PDRs in terms of CR. The aggregated timestamps are converted into time frames which result in huge compression. Additionally, differential encoding, HE and RLE further increases CR.

Table 4: Comparison of lossy compression techniques. Rn: Random sampling of 50% events. PL1, PL2: PDS-LEC with $r_4 = 1, 2$ at 0.3 Mbps. TH25, TH30, TH35: Fewer events with CT = 0.25, 0.30, 0.35 respectively.

Figure 5: Image reconstruction with encoded events with time duration 33 ms. (Outdoor 3 sequence, Frame 6)

[16] using sequences from DAVIS dataset [29]. In literature, they are the best performing event compression algorithms. In TALVEN, the events are aggregated into positive and negative event frames with histogram count at each pixel location, which are then coded (refer to subsection 2.1). The compression ratio of other algorithms [6, 3] in literature are even lower than TALVEN and Spike Coding [21] and hence not included in the comparative study. Figs. 3 and 4 show the comparative results with different time aggregations. We also compare our method by replacing the QT with 16 × 16 uniform blocks and PDS by random sampling of 50% of the events followed by temporal aggregation. The rest of the coding strategy in PDS-LEC is followed. Additional comparison is performed by generating fewer events for these sequences by using contrast thresholds (CT) of 0.25, 0.30 and 0.35 in ESIM [32] and apply PDS-LEC by replacing QT with uniform 16 × 16 blocks and removing PDS step. PDS-LEC outperforms current existing algorithms for different bit rates and PDRs in terms of CR. Even for fewer events generated with higher CT (using our compression scheme), PDS-LEC has better CR. However, for 50% random events (using our compression scheme), at lower temporal aggregation, in a few experiments, CR is higher than PDS-LEC, especially at QT derived at higher bit rates and/or lower PDR since less events are removed. However, SSIM and $T_{error}$ metrics as shown in Table 4 indicate performance comparable or worse than PDS-LEC, due to random removal of events resulting in poor SSIM and $T_{error}$ compared to PDS-LEC (please see additional details in supplementary material). Overall improvement in performance with other state-of-art and other alternative lossy techniques is due to temporal aggregation and priority based sampling of events. QT acts as a priority map for the events. The readers are encouraged to read the supplementary material.

7. Applications to Image Reconstruction

We used E2VID [33] for reconstruction of images from events. The events are compressed with PDR of $r_4 = 1$ at 300 kbps. Fig. 5(a) shows the reconstruction with original events (without compression), while Fig. 5(b), (c) and (d) shows the reconstruction for $T_{bin} = 24, 16$ and 8. The PSNR and SSIM are computed with respect to the original reconstructed image, Fig. 5(a). Both PSNR and SSIM metrics increases with the increase of $T_{bin}$, indicating preference of higher $T_{bin}$ for higher image reconstruction quality, with distorted event timestamps closer to actual timestamp.

8. Conclusion and Discussion

The paper proposes a novel lossy event compression algorithm in the spatio-temporal domain based on Poisson disk sampling and time aggregation, which achieves state-of-the-art compression performance. The algorithm uses a QT segmentation of the intensity frames which provides priority regions. We show effectiveness of PDS-LEC in different experiments: temporal, spatial, and spatio-temporal binning with comparisons and application to image reconstruction.

---

1 The readers are encouraged to read the supplementary material.
References


9. Supplementary Material

9.1. Optimized QT based on rate distortion of Images

Quadtree (QT) may be generated in several ways depending on the system and application. In some works in literature [7], the system is developed as a Host-Chip communication problem in bandwidth-constrained environment, where the QT is developed to compress the grayscale intensity frames. In this work, we use similar system setting in order to generate the QT based on the grayscale intensity frames. For a frame \( f_t \), we have a QT segmentation, skip/ acquire modes for the leaves, and values for the leaves of acquire modes, denoted by \( S_t \), \( Q_t \), and \( V_t \), respectively. These are used to reconstruct the frame \( \hat{f}_t \) in a remote host, which is distorted version of \( f_t \). The previously reconstructed frame \( \hat{f}_{t-1} \) is used to copy the values in the skip leaves of \( \hat{f}_t \), while \( V_t \) provides the acquire values of the leaves in \( S_t \).

The full resolution frame \( f_{t+1} \) is acquired at time \( t + 1 \) from the imager. The frames \( f_{t+1} \) and \( \hat{f}_t \) are inputs to the Viterbi Optimization algorithm [35], which provides the optimal QT structure \( S_{t+1} \) and skip-acquire modes \( Q_{t+1} \) subject to the bandwidth constraint \( B \) (expressed as maximum
bit rate $R_{max}$). The Viterbi optimization provides a trade-off between the frame distortion and frame bit rate. This is done by minimizing the frame distortion $D$ over the leaves of the QT $x$ subject to a given maximum frame bit rate $R_{max}$. The reconstructed frame $\hat{x}_t$ along with frame $f_{t+1}$ is used to compute the distortion at frame $t+1$.

The optimization is formulated as follows,

$$\arg \min_{x} D(x), \quad (S1)$$

$$\text{s. t. } R(x) \leq R_{max}$$

The distortion for each node of the QT, $x$, is based on the skip-acquire acquisition mode $Q_t$ of that node. If a particular node $\hat{x}_t$ of a reconstructed frame at time $t$ is skip, the distortion with respect to the new node at time $t+1$, $x_{t+1}$, is given by

$$D_s = |x_{t+1} - \hat{x}_t|, \quad (S2)$$

On the contrary, if the node is an acquire, the distortion is proportional to the standard deviation $\sigma$ as shown in Eqn. S3:

$$D_a = \sigma \times 4^{N-n}, \quad (S3)$$

where $N$ is the maximum depth of the QT and $n$ is the level of the QT where distortion is computed. The root and the most subdivided level is defined to be in level 0 and $N$ respectively. The total distortion is therefore defined as:

$$D = D_s + D_a, \quad (S4)$$

The constrained discrete optimization of Eqn. S1 is solved using Lagrangian relaxation, leading to solutions in the convex hull of the rate-distortion curve [36]. The Lagrangian cost function is of the form:

$$J_\lambda(x) = D(x) + \lambda R(x), \quad (S5)$$

where $\lambda \geq 0$ is a Lagrangian multiplier. It has been shown that if there is a $\lambda^*$ such that

$$x^* = \arg \min_{x} J_{\lambda^*}(x), \quad (S6)$$

which leads to $R(x^*) = R_{max}$ (within a tolerance), then $x^*$ is the optimal solution to Eqn. S1. In order to compress the grayscale intensity frames at desired bit-rate in an optimal manner, Lagrangian multiplier ($\lambda$) in Eqn. S5 is adjusted in each frame to achieve the desired bit rate. The optimal $\lambda^*$ is computed by a convex search in the Bezier curve [36]. A sample curve is shown in Fig. 6. In this work, for compressing the events, we consider the events in both the skip-acquire regions of the QT and compress the events depending on the QT block size. However, depending on the application, events in the acquire regions of the QT can be only compressed, while other events can be discarded.

### 9.2. Comparative Performance

In addition to the summarized results in Section 6 of the main manuscript, we present here additional comparison results using DAVIS dataset [29] sequences: Urban, Shapes, Dynamic, Running1 and Running2. We use our PDS-LEC algorithm with quad tree (QT) generated for 0.1 Mbps and 0.3 Mbps for Poisson Disk Radius (PDR) of $r_4 = 1$ and 2. In addition, the comparison of our algorithm with 50% random event sampling and reduced set of events generated by varying the thresholds to 0.25, 0.30 and 0.35 generated using [32] is shown in Table S1. These are referred to as TH25, TH30 and TH35 in Table S1. We use the same compression strategy as mentioned in Section 6 of the main manuscript. We compare compression ratio (CR), PSNR, SSIM and $T_{error}$ metrics for comparing the performance of the PDS-LEC with other reduced event generation methods. CR, PSNR, SSIM and $T_{error}$ is computed based on the original events provided in the DAVIS dataset [29]. Clearly TH25, TH30 and TH35 have lower PSNR, SSIM, CR and higher $T_{error}$ compared to results generated by PDS-LEC with almost all cases with different PDR and QT generated from different bit rates (PL1r100, PL2r100, PL1r300, PL2r300) as detailed in Table S1. This is seen irrespective of the different temporal binning $T_{bin}$ in the event compression algorithm. However, when considering the results of only TH25, TH30 and TH35, it is seen that the PSNR, SSIM and $T_{error}$ values are highest for TH35 compared to TH25 and TH30. This is an interesting observation, which highlights the fact that at higher contrast threshold (CT), events are fired which identify the significant edges of the scene.

However, for the cases with different CT (TH25, TH30 and TH35), the CR, PSNR, SSIM and $T_{error}$ are worse for all $T_{bins}$ compared to PDS-LEC, thereby suggesting significant difference in structure from the original event
DAVIS [29] sequence. It is also observed for these cases, that for $T_{bin} = 5, 10$ and $20$ ms, the $T_{error}$ is lower than $T_{bin} = 1, 40$ ms. For $T_{bin} = 1$ ms, there are fewer events which are aggregated. This suggest that the events generated by varying CT has less structural similarity in time compared to the original events, while progressively aggregating events temporally generates an event volume which has closer resemblance to original event data. However, for $T_{bin} = 40$ ms, $T_{error}$ is higher as all event timestamps are basically quantized into $40$ ms, which increases difference between original event timestamps and quantized timestamps.

On the other hand, keeping $50\%$ of the events randomly (Rn) using our compression technique as mentioned in section 6 of the main paper, the CR, PSNR, SSIM and $T_{error}$ metrics are comparable to that obtained by PDS-LEC event compression technique. However, it is seen that for $T_{bin} = 1$ ms, the CR for Rn is higher, alongwith lower SSIM, PSNR and higher $T_{error}$ than PDS-LEC. This indicates that events sampled randomly at $T_{bin} = 1$ ms has less resemblance to original events in the DAVIS dataset. For higher $T_{bins} = 5, 10, 20, 40$ ms, the CR of PDS-LEC is higher than Rn, thereby implying events less than $50\%$ are sampled. However PSNR, SSIM and $T_{error}$ of Rn is comparable to PDS-LEC.

It is observed that for PDS-LEC at 0.1 Mbps with $r_4 = 1, 2$, the CR is higher than that at 0.3 Mbps, with slightly lower SSIM, PSNR and higher $T_{error}$. Clearly, PDS-LEC offers the flexibility of varying CR, and hence PSNR, SSIM and $T_{error}$, depending on the bit rate available for transmission and/or storage by varying the PDR ($r_4$) as demonstrated throughout the paper based on priority given by QT. The algorithm performs better than fewer events generated with varying CT, and comparable with events sampled randomly at $50\%$. However, overall, PDS-LEC has better CR and (or) PSNR / SSIM / $T_{error}$ metrics than Rn as seen in Table S1. Moreover, Rn does not offer much flexibility in terms of varying the CR based on available bit rate. Thus PDS-LEC clearly has more potential and flexibility of compressing events in a lossy manner than any other method in literature or other naive techniques.
### Table S1: Comparison of Compression Ratio (CPR), PSNR, SSIM and $T_{error}$ for various lossy compression techniques.

<table>
<thead>
<tr>
<th>Sequence</th>
<th>method</th>
<th>1 ms</th>
<th>5 ms</th>
<th>10 ms</th>
<th>20 ms</th>
<th>40 ms</th>
</tr>
</thead>
<tbody>
<tr>
<td>Urban</td>
<td>Rn</td>
<td>63.57</td>
<td>53.10</td>
<td>0.9952</td>
<td>2.100</td>
<td>64.54</td>
</tr>
<tr>
<td>PL1r100</td>
<td>38.34</td>
<td>65.33</td>
<td>0.9987</td>
<td>1.172</td>
<td>56.41</td>
<td>59.61</td>
</tr>
<tr>
<td>PL2r100</td>
<td>45.22</td>
<td>61.72</td>
<td>0.9970</td>
<td>1.558</td>
<td>83.10</td>
<td>56.90</td>
</tr>
<tr>
<td>PL3r100</td>
<td>55.09</td>
<td>68.22</td>
<td>0.9994</td>
<td>0.929</td>
<td>16.62</td>
<td>6.023</td>
</tr>
<tr>
<td>PL2r300</td>
<td>38.70</td>
<td>64.89</td>
<td>0.9984</td>
<td>0.120</td>
<td>60.22</td>
<td>59.91</td>
</tr>
<tr>
<td>TH25</td>
<td>30.52</td>
<td>45.01</td>
<td>0.9813</td>
<td>2.906</td>
<td>36.21</td>
<td>45.01</td>
</tr>
<tr>
<td>TH30</td>
<td>30.57</td>
<td>45.64</td>
<td>0.9983</td>
<td>2.918</td>
<td>45.96</td>
<td>45.64</td>
</tr>
<tr>
<td>TH35</td>
<td>47.60</td>
<td>46.08</td>
<td>0.9840</td>
<td>0.927</td>
<td>56.61</td>
<td>46.08</td>
</tr>
<tr>
<td>Shaper</td>
<td>Rn</td>
<td>61.11</td>
<td>27.99</td>
<td>0.9470</td>
<td>2.352</td>
<td>46.36</td>
</tr>
<tr>
<td>PL1r100</td>
<td>48.45</td>
<td>30.42</td>
<td>0.9741</td>
<td>2.117</td>
<td>94.88</td>
<td>25.91</td>
</tr>
<tr>
<td>PL2r100</td>
<td>65.99</td>
<td>27.92</td>
<td>0.9545</td>
<td>1.369</td>
<td>161.00</td>
<td>24.50</td>
</tr>
<tr>
<td>PL1r300</td>
<td>40.10</td>
<td>37.20</td>
<td>0.9096</td>
<td>0.724</td>
<td>50.04</td>
<td>31.75</td>
</tr>
<tr>
<td>PL2r300</td>
<td>45.56</td>
<td>34.35</td>
<td>0.9038</td>
<td>0.914</td>
<td>59.35</td>
<td>25.96</td>
</tr>
<tr>
<td>TH25</td>
<td>2.94</td>
<td>1.43</td>
<td>0.8533</td>
<td>1.246</td>
<td>11.22</td>
<td>1.43</td>
</tr>
<tr>
<td>TH30</td>
<td>3.54</td>
<td>1.34</td>
<td>0.8590</td>
<td>1.265</td>
<td>11.33</td>
<td>1.34</td>
</tr>
<tr>
<td>TH35</td>
<td>4.14</td>
<td>4.59</td>
<td>0.8624</td>
<td>1.307</td>
<td>12.36</td>
<td>4.59</td>
</tr>
<tr>
<td>Dynamic</td>
<td>Rn</td>
<td>62.87</td>
<td>28.55</td>
<td>0.9491</td>
<td>2.211</td>
<td>65.58</td>
</tr>
<tr>
<td>PL1r100</td>
<td>38.07</td>
<td>34.63</td>
<td>0.9717</td>
<td>0.810</td>
<td>58.36</td>
<td>28.82</td>
</tr>
<tr>
<td>PL2r100</td>
<td>66.55</td>
<td>31.13</td>
<td>0.9533</td>
<td>0.765</td>
<td>92.63</td>
<td>26.52</td>
</tr>
<tr>
<td>PL1r300</td>
<td>45.55</td>
<td>38.08</td>
<td>0.9060</td>
<td>0.956</td>
<td>44.49</td>
<td>31.58</td>
</tr>
<tr>
<td>PL2r300</td>
<td>37.94</td>
<td>34.70</td>
<td>0.9649</td>
<td>1.299</td>
<td>57.22</td>
<td>28.83</td>
</tr>
<tr>
<td>TH25</td>
<td>4.12</td>
<td>1.66</td>
<td>0.5454</td>
<td>2.409</td>
<td>13.63</td>
<td>1.66</td>
</tr>
<tr>
<td>TH30</td>
<td>4.97</td>
<td>5.34</td>
<td>0.5580</td>
<td>2.449</td>
<td>14.61</td>
<td>5.34</td>
</tr>
<tr>
<td>TH35</td>
<td>5.83</td>
<td>6.790</td>
<td>0.5670</td>
<td>2.551</td>
<td>14.94</td>
<td>6.790</td>
</tr>
<tr>
<td>Running1</td>
<td>Rn</td>
<td>62.87</td>
<td>29.07</td>
<td>0.8861</td>
<td>2.467</td>
<td>65.58</td>
</tr>
<tr>
<td>PL1r100</td>
<td>44.09</td>
<td>32.49</td>
<td>0.9450</td>
<td>0.810</td>
<td>74.45</td>
<td>38.06</td>
</tr>
<tr>
<td>PL2r100</td>
<td>58.51</td>
<td>29.69</td>
<td>0.8932</td>
<td>2.309</td>
<td>125.89</td>
<td>26.88</td>
</tr>
<tr>
<td>PL1r300</td>
<td>38.38</td>
<td>33.94</td>
<td>0.9678</td>
<td>1.444</td>
<td>55.31</td>
<td>29.94</td>
</tr>
<tr>
<td>PL2r300</td>
<td>45.82</td>
<td>32.08</td>
<td>0.9331</td>
<td>1.930</td>
<td>82.57</td>
<td>27.62</td>
</tr>
<tr>
<td>TH25</td>
<td>8.70</td>
<td>7.32</td>
<td>0.5540</td>
<td>2.313</td>
<td>20.34</td>
<td>7.32</td>
</tr>
<tr>
<td>TH30</td>
<td>10.62</td>
<td>8.94</td>
<td>0.5740</td>
<td>3.235</td>
<td>23.43</td>
<td>8.94</td>
</tr>
<tr>
<td>TH35</td>
<td>12.59</td>
<td>10.274</td>
<td>0.5880</td>
<td>3.259</td>
<td>26.71</td>
<td>10.274</td>
</tr>
<tr>
<td>Running2</td>
<td>Rn</td>
<td>63.34</td>
<td>25.64</td>
<td>0.8031</td>
<td>3.372</td>
<td>65.67</td>
</tr>
<tr>
<td>PL1r100</td>
<td>45.84</td>
<td>29.20</td>
<td>0.9919</td>
<td>2.607</td>
<td>80.20</td>
<td>24.92</td>
</tr>
<tr>
<td>PL2r100</td>
<td>59.91</td>
<td>31.91</td>
<td>0.9505</td>
<td>2.130</td>
<td>59.26</td>
<td>28.63</td>
</tr>
<tr>
<td>PL1r300</td>
<td>50.46</td>
<td>28.09</td>
<td>0.9851</td>
<td>2.869</td>
<td>93.00</td>
<td>24.31</td>
</tr>
<tr>
<td>TH25</td>
<td>10.27</td>
<td>7.59</td>
<td>0.5194</td>
<td>4.396</td>
<td>6.23</td>
<td>5.79</td>
</tr>
<tr>
<td>TH30</td>
<td>12.48</td>
<td>9.21</td>
<td>0.5341</td>
<td>4.425</td>
<td>29.49</td>
<td>9.21</td>
</tr>
<tr>
<td>TH35</td>
<td>14.72</td>
<td>10.539</td>
<td>0.5436</td>
<td>4.458</td>
<td>33.07</td>
<td>10.539</td>
</tr>
</tbody>
</table>

(CR / PSNR / SSIM / $T_{error}$)

**Urban:** Rn: Mbps. PL1r130: PDS-LEC with PL2r100: PDS-LEC with PL3r100: PDS-LEC with PL2r300: PDS-LEC with TH25: Fewer events with CT = 0.25. TH30: Fewer events with CT = 0.30. TH35: Fewer events with CT = 0.35.