2025 Design, Automation & Test in Europe Conference (DATE 2025)

Simultaneous Denoising and Compression for DVS with Partitioned Cache-Like Spatiotemporal Filter

Qinghang Zhao, Yixi Ji, Jiaqi Wang, Jinjian Wu, and Guangming Shi
School of Artificial Intelligence, Xidian University, Xi'an, China
{qhzhao, jinjian.wu, gmshi}@xidian.edu.cn; {23171214523, 20009101376}@stu.xidian.edu.cn

***Abstract*—Dynamic vision sensor (DVS) is a novel neuromorphic imaging device that asynchronously generates event data corresponding to changes in light intensity at each pixel. However, the differential imaging paradigm of DVS renders it highly sensitive to background noise. Additionally, the substantial volume of event data produced in a very short time presents significant challenges for data transmission and processing. In this work, we present a novel spatiotemporal filter design, named PCLF, to achieve simultaneous denoising and compression for the first time. The PCLF employs a hierarchical memory structure that utilizes symmetric multi-bank cache-like row and column memories to store event data from a partitioned pixel array, which exhibits low memory complexity of $\mathrm{O(m+n)}$ for an $\mathrm{m \times n}$ DVS. Furthermore, we propose a probability-based criterion to effectively control the compression ratio. We have implemented our design on an FPGA, demonstrating capabilities for real-time operation (≤ 60 ns) and low power consumption (< 200 mW). Extensive experiments conducted on real-world DVS data across various tasks indicate that our design enables a reduction of event data by 30% to 68%, while maintaining or even enhancing the performance of the tasks.**

***Index Terms*—Dynamic vision sensor, denoising, data compression, spatiotemporal filter, FPGA**

I. Introduction

The Dynamic Vision Sensor (DVS) is a kind of neuromorphic imaging device that mimics biological vision. When the relative variation in perceived light intensity exceeds a specific threshold, the corresponding pixel asynchronously generates an event. This unique imaging paradigm enables DVS to achieve high temporal resolution, high dynamic range, and low power consumption [1, 2], rendering it valuable for a wide range of computer vision (CV) tasks, such as object detection, object tracking, video deblurring, and video frame interpolation.

However, the imaging paradigm of DVS makes it highly sensitive to the background activity (BA) caused by thermal noise and junction leakage current [3], which adversely affects the quality of the output signal, leads to increased communication bandwidth and power consumption, and ultimately impacts the performance of subsequent tasks. Therefore, denoising plays a crucial role in DVS data processing. Denoising techniques can be broadly categorized into offline and online methods. Offline methods leverage probabilistic undirected graphs [4], event density [5], deep neural networks [6], and graph neural networks [7]. Although these approaches demonstrate adequate performance, they often involve significant computational demands, rendering them difficult to implement in hardware and unsuitable for real-time applications. In contrast, online methods primarily employ spatiotemporal filters (STFs), which exploit the fact that noise is random while valid events exhibit spatiotemporal correlations. Despite the development of various STFs [3, 8, 9], these designs typically encounter issues related to high memory complexity or diminished denoising performance.

In addition, the high sensitivity and high temporal resolution of DVS result in a massive influx of events in a short time, especially in scenarios involving rapid motion, which presents significant transmission, computation, and storage burdens. To address this challenge, lossless compression methods have been proposed [10, 11], which primarily utilize coding techniques. However, the hardware overhead associated with these coding and encoding processes is considerable, and the potential for latency should not be underestimated. In reality, there exists substantial information redundancy within DVS event data. Consequently, it is not always essential to retain all event data in its entirety; thus, lossy compression methods may be applicable for a majority of use cases.

In this work, we introduce a Partitioned Cache-Like Spatiotemporal Filter (PCLF) design, characterized by linear memory complexity of $\mathrm{O(m+n)}$ for a DVS with a resolution of $\mathrm{m \times n}$. This design facilitates the simultaneous denoising and compression of event data. The specifics of our design will be elaborated upon in Section III, and the principal contributions of this work are summarized as follows:

- We present a novel STF design, termed PCLF, which enables the simultaneous denoising and compression of events for the first time by employing a probability-based criterion to determine whether an event should be retained or discarded.
- We propose a hierarchical memory organization within the PCLF design that utilizes symmetric cache-like row and column memories for a partitioned pixel array, which achieves a low memory complexity of $\mathrm{O(m+n)}$ for $\mathrm{m \times n}$ DVS while maintaining high performance.
- We implement our design on an FPGA, which demonstrates real-time capabilities (≤ 60 ns) and low power consumption (< 200 mW). Furthermore, extensive experiments on real-world DVS data indicate that our approach can reduce event data by 30% to 68% while preserving or even enhancing the performance of downstream tasks.

Corresponding author: Jinjian Wu. This work was supported by the National Natural Science Foundation of China under Grant 62104182.

II. Background

A. Principle of Spatiotemporal Filter

An event generated by the pixels of a DVS can be represented as $e_k = (x_k, y_k, t_k, p_k)$, where x_k and y_k denote the row and column coordinates of the pixel, respectively. t_k is the timestamp indicating when the event occurs, and $p_k \in \{+1, -1\}$ represents the polarity, indicating whether the variation of perceived light intensity is positive or negative.

The STF serves as a method for DVS denoising, predicated on the principle that the imaging of objects exhibits high spatial and temporal correlation, whereas noise is characterized by randomness. Considering an event e_k and a previous one $e_i = (x_i, y_i, t_i, p_i)$ in the stream, if the Chebyshev distance between the two pixels is within a certain threshold d_{th}, they are considered spatially correlated: $|x_i - x_k| \leqslant d_{th}$ and $|y_i - y_k| \leqslant d_{th}$. Similarly, if the difference in their timestamps is within a designated threshold: $|t_i - t_k| \leqslant \tau_{th}$, the events are regarded as temporally correlated. The threshold parameters d_{th} and τ_{th} thus delineate the sizes of the spatial and temporal windows, respectively. e_k and e_i are considered correlated if they exhibit both spatial and temporal correlation, denoted as $\langle e_k,\, e_i \rangle$. The event e_k is regarded as a valid signal if the number of correlated events exceeds a predetermined threshold N_{cr}, which can be formalized as follows:

$$\mathbb{E}_k = \{e_i \mid \langle e_k,\, e_i \rangle\} \tag{1}$$

$$|\mathbb{E}_k| \geq N_{cr} \tag{2}$$

where $\mathbb{E}_k$ represents the set of events correlated with e_k. If the condition in (2) is not satisfied, then e_k is classified as noise.

B. Related Works

It can be seen that the STF is straightforward in principle and easy to implement in hardware, in which the key is to preserve the information of previous events. In this regard, various STF designs have been proposed. Fig. 1(a) depicts the memory utilization of the elementary version, in which each sensor pixel corresponds to one memory unit to store the timestamp of an event. Based on this principle, Delbruck et al. introduced the background activity filter (BAF) [12]. In BAF, the timestamp of the input event is compared with that of the corresponding memory unit to determine whether it is noise. Subsequently, the memory units in the spatial window are updated with the input event. Similarly, Guo et al. [3] proposed the Spatiotemporal Correlation Filter (STCF), which allows the parameter N_{cr} to vary from 1 to 8 for a 3×3 spatial window, thereby enhancing the adaptability of the design. However, both BAF and STCF exhibit $\mathrm{O(m \cdot n)}$ memory complexity. As the resolution of DVS increases, the memory overhead can become substantial. For instance, utilizing a 4-byte timestamp requires 4MB of memory for a 1M-pixel DVS [13], which is prohibitively expensive for on-chip implementation.

Liu et al. [9] introduced an enhanced design referred to as Subsampling Shared Memory (SSM), in which $r \times r$ pixels share a single memory unit, as illustrated in Fig. 1(b). While the memory capacity of SSM is reduced by a factor of r^2, it retains an $\mathrm{O(m \cdot n)}$ complexity scheme. Khodamoradi et al. [8] presented a STF with $\mathrm{O(n)}$ memory complexity for the first time. As demonstrated in Fig. 1(c), each row of pixels shares one memory unit in Memory1, which stores the row coordinate (x) and timestamp (t) of input events originating from that row. Memory2 operates in a similar fashion. This design has been implemented on an FPGA and has demonstrated efficiency for simple applications. However, its performance deteriorates significantly in complex scenarios involving multiple moving objects, as a single memory unit for an entire row or column is insufficient for recording past events.

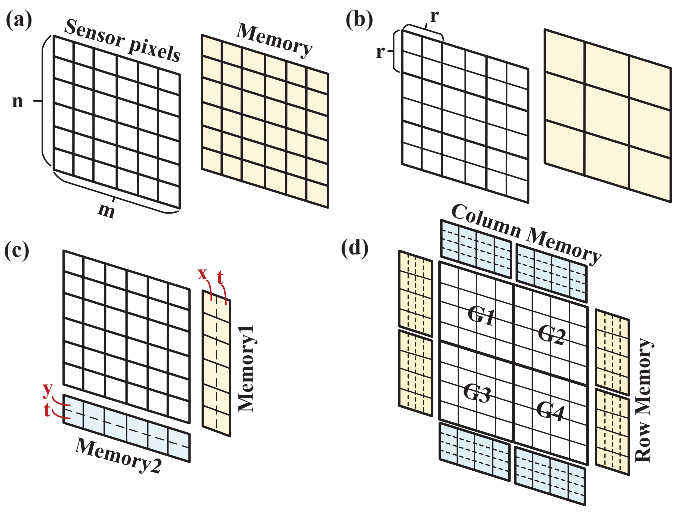


Fig. 1. Memory organization of different spatiotemporal filters.

III. Partitioned Cache-Like Spatiotemporal Filter

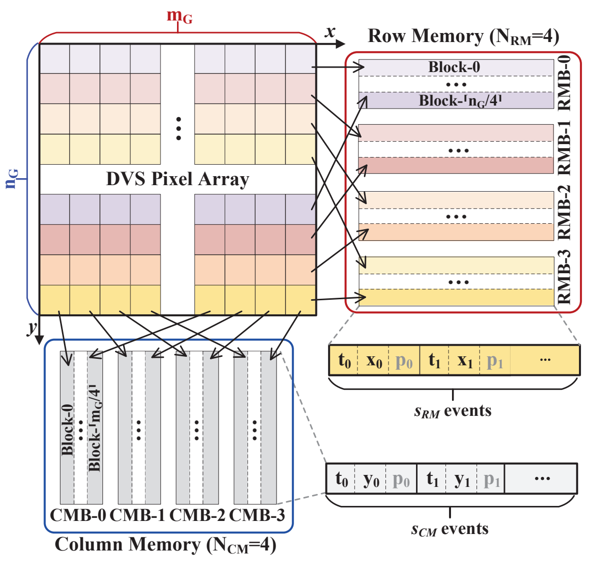
In this section, we present the PCLF design aimed at achieving concurrent denoising and compression while maintaining a low memory complexity of $\mathrm{O(m + n)}$ for an $\mathrm{m \times n}$ DVS. The symbols used throughout this paper are compiled in Table I.

TABLE I
List of Symbols and Notations.

Symbol	Description
τ_{th}	Threshold for the assessment of temporal correlation
d_{th}	Threshold for the assessment of spatial correlation
n_k	Number of correlated events in the spatiotemporal window of e_k
γ	Scale factor that adjusts the shape of the probability function
N_{RM}	Number of Row Memory Banks
N_{CM}	Number of Column Memory Banks
s_{RM}	Number of events that can be stored in each block of row memory bank
s_{CM}	Number of events that can be stored in each block of column memory bank

A. Memory Organization

We design a hierarchical memory organization in PCLF. First, as illustrated in Fig. 1(d), the DVS pixels are logically segmented into multiple regions, with each region associated with two corresponding memories: Row Memory and Column Memory. The memory organization for an individual region with a resolution of $\mathrm{m_G \times n_G}$ is illustrated in Fig. 2. The Row Memory comprises N_{RM} memory banks (denoted as RMB-0, RMB-1, ...), with each bank containing $\lceil n_G / N_{RM} \rceil$ memory



m_G
x
Row Memory (N_{RM}=4)
Block-0
Block-$\lceil n_G/4 \rceil$
RMB-0
RMB-1
RMB-2
RMB-3
n_G
DVS Pixel Array
y
t_0 x_0 p_0 t_1 x_1 p_1 ···
s_{RM} events
Block-0
Block-$\lceil m_G/4 \rceil$
t_0 y_0 p_0 t_1 y_1 p_1 ···
s_{CM} events
CMB-0 CMB-1 CMB-2 CMB-3
Column Memory (N_{CM}=4)

Fig. 2. The cache-like memory organization of each partitioned pixel array.

blocks, wherein each memory block is capable of storing s_{RM} events. When a new event e_k is generated from this region, its information is stored in the $\lfloor y_k/N_{RM} \rfloor$-th block of RMB-$[y_k \bmod N_{RM}]$. For instance, if $N_{RM} = 4$, events arising from rows 0, 4, 8, etc., will be allocated to Block-0, Block-1, Block-2, and so forth in RMB-0, respectively. In essence, events generated from the same row of pixels are stored in a common memory block and at arbitrary positions, akin to the operation of a set-associative cache. In alignment with the principles of STF, upon the arrival of a new event, it is essential to retrieve information regarding preserved events of neighboring pixels within the spatial window for correlation assessment. Consequently, multiple memory blocks corresponding to several contiguous rows must be accessed. The employed multi-bank memory design is specifically intended to facilitate simultaneous access to these memory blocks. In practice, the size of the spatial window is specified as an odd number to ensure symmetry. For a designated $D \times D$ spatial window, we define N_{RM} as $2^{\lceil \log_2 D \rceil}$ in our design rather than directly as D. This approach allows for the distinct identification of memory banks via the last $\lceil \log_2 D \rceil$ bits of the reference address, thereby obviating the necessity for a modular operation with respect to D. For example, by setting N_{RM} to 4 to accommodate a 3×3 spatial window, the last two bits of y_k will indicate which memory bank to access.

The memory block functions as the fundamental unit for read and write operations. Typically, s_{RM} is significantly smaller than the number of pixels within a single row. Therefore, each memory block fills rapidly as events stream in. When this occurs, newly arrived events replace previous events residing in the same block. We employ a First-In-First-Out (FIFO) replacement strategy in our design, which is advantageous from a hardware perspective. Furthermore, due to the principles of temporal and spatial locality, the earliest event in storage is less likely to correlate with incoming events. Given that the timestamps of event streams are monotonically non-decreasing, the FIFO policy behaves similarly to a Least Recently Used (LRU) strategy for DVS.

When the imaging scene becomes complex with multiple moving objects, the need arises for larger storage capacity to accommodate the events generated in rapid succession. While one straightforward solution is to increase the capacity of each memory block, this approach introduces additional energy consumption overhead not only during read and write operations but also when concurrently comparing the input event against a multitude of stored events for correlation assessment. Instead, we propose partitioning the pixel array of the DVS into several regions, each with its own dedicated row and column memory. This configuration maintains small memory block sizes and minimizes energy consumption overhead, as only the memories corresponding to specific regions are accessed during each operation.

The Column Memory operates in a manner analogous to Row Memory, except that the column coordinate y_k of the input event is stored instead of the row coordinate x_k. We will omit the details here to avoid redundancy.

B. Hardware Architecture

Fig. 3 illustrates the architecture of PCLF for $\mathrm{m_G} \times \mathrm{n_G}$ regions. PCLF comprises the Row Correlation Module (RCM), the Column Correlation Module (CCM), and the Event Drop Module (EDM). In both RCM and CCM, the memory is organized as previously described. In this exemplary design, RCM has four memory banks ($N_{RM} = 4$), and each memory block is capable of storing four events ($s_{RM} = 4$). The threshold distance d_{th} is set to 1, indicating that previous events within a 3×3 spatial window should be examined, specifically involving the two adjacent rows and two columns. Assuming the current input event is e_k and the last two bits of y_k are $(00)_2$, first $y_k - 1$ and $y_k + 1$ are calculated with two adders, resulting in last two bits being $(11)_2$ and $(01)_2$, respectively. Consequently, RMB-0, RMB-3, and RMB-1 of the Row Memory will be accessed, with access addresses equal to the most significant $BW_y - 2$ bits of y_k, $y_k - 1$, and $y_k + 1$, denoted as *raddr*, *raddrn*, and *raddrp*, respectively. The read data from RMB-0, referred to as *rdata*, contains information of the least s_{RM} events generated by the same row of e_k. *rdata* is then sent to the Event Correlation Unit (ECU), which first assesses whether e_k correlates with the stored events by calculating the difference in row coordinates and timestamps, comparing them against the preset thresholds τ_{th} and d_{th}. The ECU subsequently counts the number of correlated events. The data read from RMB-1 and RMB-3 (*rdatap* and *rdatan*) undergo similar processing with two other ECUs, resulting in a count of correlated events in the adjacent rows. Finally, the outputs from all three ECUs are aggregated, denoted as s_r.

Subsequently, the input event replaces a previous event in the corresponding memory block of RMB-0, adhering to the FIFO policy. To indicate the position of replacement, a small scratchpad memory, referred to as Write Position Tracker (WPT), is employed for each memory bank. The WPT consists of as many memory units as memory blocks within the memory bank, with a bitwidth of $\lceil \log_2 s_{RM} \rceil$. WPT tracks the most

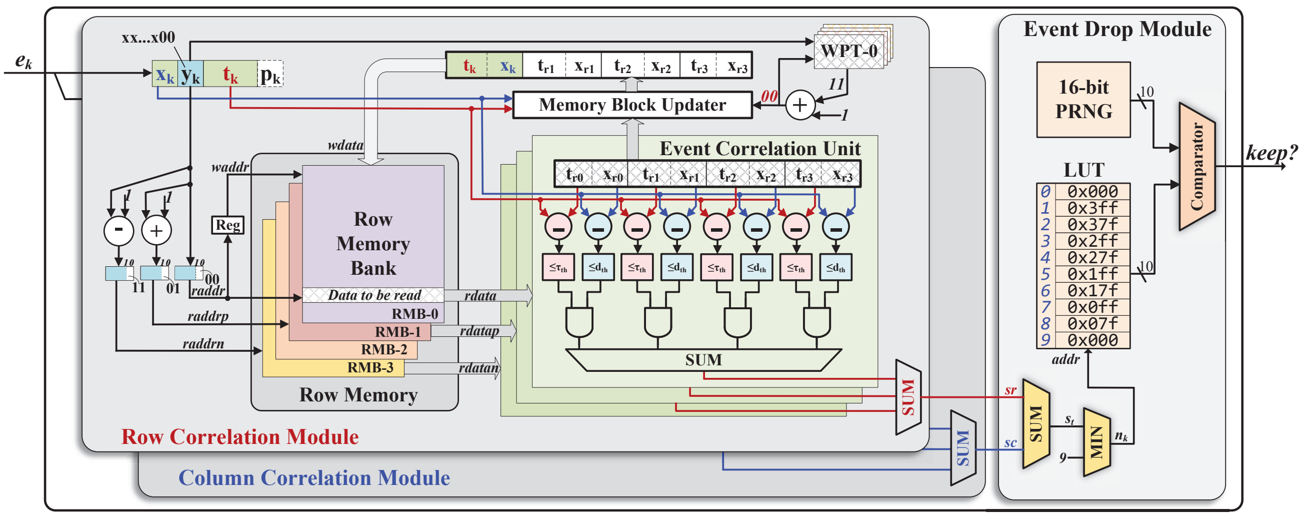


Fig. 3. The architecture of the proposed PCLF.

recent replacement position, which increments by 1 with each update to the memory bank. For instance, if the value of WPT-0's corresponding element is $(11)_2$, this indicates that the last update occurred at position 3; thus, the input event will be written to position 0.

The CCM operates analogously to RCM, and further details are omitted for the sake of brevity.

EDM sums the outputs of both RCM and CCM to calculate the total number of correlated events across all three rows and three columns, denoted as s_t. Since there may be duplicate events present in the Row and Column Memory, s_t can exceed 9 within the 3×3 spatial window. A minimum selection unit is adopted to constrain the output to a maximum of 9, represented as n_k.

Previous STFs are designed solely for denoising, determining the validity of input based on whether the number of correlated events exceeds a predetermined threshold. This work thinks more broadly from a novel probabilistic perspective. Specifically, when $n_k = 0$, indicating that no correlated events have been detected, the input is classified as noise. Conversely, if $n_k > 0$, there is a likelihood that the DVS has recently generated valid events from neighboring pixels. Given the motion continuity of the imaging object, the generated events tend to exhibit redundancy. As the value of n_k increases, the amount of redundant information also increases. Consequently, we will retain the input event with probability, where a larger n_k correlates with a lower retention probability. In other words, the probability of retaining an event is monotonically non-increasing as n_k increases. To account for this relationship, we propose the following probability functions:

$$f_{\text{convex}}(n_k) = (1 - \log_s n_k)^{\gamma} \tag{3}$$

$$f_{\text{linear}}(n_k) = \frac{s - n_k}{s - 1} \tag{4}$$

$$f_{\text{concave}}(n_k) = (\log_s (s - n_k - 1))^{\frac{1}{\gamma}} \tag{5}$$

where $s = D^2$ is the total number of pixels within the $D \times D$ spatial window, and γ is a scale factor that modulates the shape of the probability function. These probability functions can be selected, along with the parameter γ, to effectively control the ratio of discarded events.

To implement various probability functions, we employ a pseudo-random number generator (PRNG) and a lookup table (LUT) in EDM, as illustrated in Fig. 3. The PRNG utilizes a 16-bit linear feedback shift register (LFSR), with the feedback polynomial defined as $x^{16} + x^{15} + x^{13} + x^4 + 1$, achieving a maximum period of $2^{16} - 1$. However, each number generated by the LFSR appears only once per cycle period. To enhance the quality of the sequence, we utilize the lower 10 bits of the LFSR output to generate random numbers. This approach ensures that each 10-bit number can be observed 64 times (with the exception of zero, which appears 63 times) before the LFSR completes one full cycle. The LUT contains $s+1$ entries, with the preserved value calculated as follows:

$$\text{LUT}[n_k] = \begin{cases} 0 & \text{if } n_k = 0 \\ \max\{0, \lceil 2^{10} \times f_p(n_k) - 1 \rceil\} & \text{if } n_k > 0 \end{cases} \tag{6}$$

where $f_p(\cdot)$ represents the adopted probability function as defined in (3) to (5). For instance, the LUT shown in Fig. 3 illustrates the stored values for a linear probability function. The retrieved value is subsequently compared with the pseudo-random number generated by the PRNG to determine whether the input event should be retained or dropped.

As depicted in Fig. 3, the proposed PCLF can process an input event and produce a result within four clock cycles. Considering the pixel array partitioning, which may require an additional two cycles to compute the row and column coordinates for each region using a multi-stage subtractor, the total delay for the complete implementation is limited to six clock cycles. This indicates that the PCLF is suitable for real-time applications.

C. Bitwidth Reduction of Timestamp

The timestamp of the event generated by DVS is typically 32 bits or more [13]. In previous implementations of STF [8, 14], the timestamp is stored in memory unaltered, which dominates memory usage. The stored timestamp is utilized for the temporal correlation assessment of events. Therefore, the bitwidth of the timestamp can be reduced provided that it meets the following requirements:

1) The time threshold τ_{th} can be accurately represented;
2) Misjudgment of the temporal correlation between events is minimized.

For the first requirement, it is essential to retain the lower bits of the timestamp with high priority; this is because the lower the weight of the least significant bits (LSBs), the finer the representation of a number. Let T_s denote the range that the timestamp field can represent. If the first subsequent event of the current input within the spatial window falls within the time range of $[n \cdot T_s, n \cdot T_s + \tau_{th}]$, there is a likelihood of misjudgment regarding temporal correlation. Consequently, addressing the second requirement necessitates the retention of higher bits of the timestamp to maximize the ratio $\frac{T_s}{\tau_{th}}$. Intuitively, these two requirements may appear to be in conflict. However, our experimental findings suggest that the optimal value for τ_{th} lies within a moderate range. Accordingly, in our design, the timestamp of the input event is truncated to 8 bits, prioritizing the second requirement while still ensuring effective compliance with the overall constraints.

IV. Evaluation

A. Hardware Implementation

TABLE II
Resource Utilization and Power Consumption of FPGA Implementation of PCLF.

Configuration		Resource Utilization			Power[1]
Resolution	Partition	LUT	LUTRAM	FF	(mW)
240×180	2×2	8212	1600	5456	118
346×260	1×1	3647	896	2173	110
346×260	2×2	9228	1664	6172	124
346×260	3×3	17091	3744	11242	164
1280×800	1×1	6768	4640	2648	138
1280×800	2×2	17424	7680	7228	158
Available		20800	9600	41600	\

[1] Estimated by Vivado with switching activity information provided.

In this section, we validate the effectiveness of our proposed PCLF. We implemented our design on a low-cost AMD XC7A35T FPGA from the Artix-7 family. Several versions are developed with different configurations of DVS resolution and partitioned regions to facilitate the experiments described in the subsequent sections. For these implementations, the parameters N_{RM} and N_{CM} are set to 8, and s_{RM} and s_{CM} are set to 4. All implementations operate based on a 100 MHz clock, ensuring that the delay time does not exceed 60 ns. The resource utilization and power consumption for each configuration are summarized in Table II. Specifically, the total power consumption across different configurations ranges from 110 mW to 164 mW. Overall, our design demonstrates low power consumption and efficient resource utilization, making it suitable for low-end FPGA devices or for implementation with ASIC.

B. Validation with Tracking Task

We employ various downstream CV tasks to validate the effectiveness of our PCLF. We begin with object tracking experiments, which utilize the ViPT [15] tracker on the large-scale RGB-event dataset VisEvent [16], all with a resolution of 346×260. In our PCLF design, we implement three partitioning strategies for the pixel array: 1×1 (no partitioning), 2×2, and 3×3, as detailed in Table II. Since previous STFs have primarily been utilized for denoising, we compare our approach against BAF [12] followed by a simple k-drop-1 method [17], which is denoted as "BAF-Nk". For instance, "BAF-N3" indicates that one event is dropped for every three events processed. We maintain uniform parameters across all methods: τ_{th} is set at $2^{10}\mu s$, with the lower 9 bits of the timestamp truncated; d_{th} is 2, representing a 5×5 spatial window; and f_{concave} is utilized with γ set to 4. The compression ratios (CR) for all methods are as follows: 46.08%, 43.16%, and 42.16% for PCLF with 1×1, 2×2, and 3×3 partitioned regions, respectively, and 60.26% for BAF-N3. Fig. 4 shows representative visualized samples with fixed cumulative time as raw data. We can see that our approach exhibits a better visual effect, while raw data shows rough edges and BAF-N3 loses some details.

With the processed event data and RGB data combined, the tracker is trained and evaluated under consistent conditions. For the tracking evaluation metrics, we select the Precision Rate (PR) and Success Rate (SR) [16]. The results are listed in Table III, which shows that our method achieves comparable tracking performance to the baseline, with SR even slightly improved. In contrast, the performance of the BAF-N3 method declines significantly.

TABLE III
Precision Rate and Success Rate on the VisEvent test set.

Metric	Raw data	BAF-N3	PCLF (1×1)	PCLF (2×2)	PCLF (3×3)
PR	**0.724**	0.716	0.723	0.720	0.723
SR	0.589	0.582	**0.590**	0.587	**0.590**

C. Validation with Recognition Task

In this section, we further validate our design through object recognition tasks.

1) Convolutional Neural Network: Our experiments are based on the N-Caltech 101 dataset [18], which consists of 101 classes and a total of 8,246 samples, all with a resolution of 240×180. We investigate the effects of varying the compression ratio (CR) at 30% and 50%. A CR of 30% is attained through specific parameter settings: τ_{th} is 48 μs, the lower 4 bits of the timestamp are truncated, d_{th} is 2, and f_{concave} is adopted with γ set to 4. For a CR of 50%, d_{th} is adjusted to 1 and f_{convex} is utilized, also with γ set to 4. The implemented PCLF is subsequently utilized to process the event data.

For classification, we utilize the ResNet-34 model [19] and fine-tune both the first and last layers of the pre-trained model. Our training strategy incorporates a cross-entropy loss function along with the ADAM optimizer, beginning with an initial learning rate of 10^{-4}, which is decreased by a factor of 10

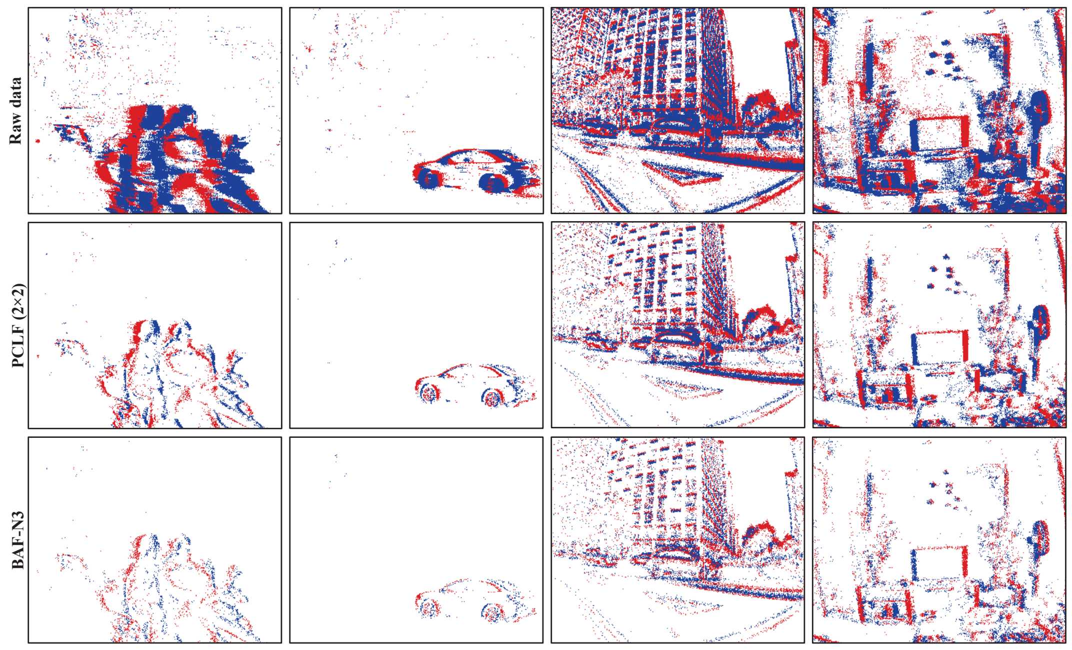


Fig. 4. Visualization of the VisEvent dataset processed with PCLF and BAF-N3.

every 30 iterations. A batch size of 64 is employed, with the total number of training iterations fixed at 100.

TABLE IV
CLASSIFICATION RESULTS OF RESNET-34 ON N-CALTECH 101 DATASET.

CR	Accuracy (Relative)	Memory	Training Speed
0 (Raw data)	79.30% (-)	1.7G	7.75 min/epoch
30%	79.45% (+0.15%)	1.0G	5.62 min/epoch
50%	79.49% (+0.19%)	881.9M	4.21 min/epoch

The training efficiency in terms of memory requirements, training speed, and classification results is presented in Table IV. Notably, the classification accuracy is slightly improved with both CR values. Concurrently, a significant enhancement in training efficiency is observed, with memory requirements reduced by up to 48% and training speed improved by $1.8\times$. These results suggest that our PCLF design facilitates more efficient processing in downstream tasks with CNN models while maintaining or even enhancing accuracy.

2) Spiking Neural Network: Spiking Neural Networks (SNNs) represent an emerging class of networks that operate on an event-driven principle. In this context, SNNs are inherently suited for processing DVS data. In this experiment, we utilize the DVS-CIFAR10 dataset [20]. Additionally, we have constructed a dataset using the CeleX-V event camera [13], referred to as Transportation8. This dataset comprises 8 classes of vehicles, with each class containing 1,800 samples with a resolution of 1280×800. Both datasets are processed using our PCLF under various configurations of partitioned regions. Subsequently, the SNN model TET [21] is trained on these datasets for classification. The results are presented in Table V.

TABLE V
CLASSIFICATION RESULTS ON DVS-CIFAR10 AND TRANSPORTATION8.

Dataset	Method	CR	Accuracy
DVS-CIFAR10	Raw data	-	77.36%
	PCLF (2×2)	42%	78.40%
Transportation8	Raw data	-	84.75%
	PCLF (1×1)	59%	84.49%
	PCLF (2×2)	68%	85.23%

It is evident that the classification accuracy for both processed datasets surpasses the original results across all configurations, despite discarding 42% to 68% of the total events, which are not fed into the TET model. The asynchronous operation of SNNs suggests that neurons remain inactive when no input is received, indicating that event compression has the potential to significantly reduce power consumption associated with SNN data processing.

V. CONCLUSION

This work presents a novel spatiotemporal filter design, termed PCLF, to realize simultaneous denoising and compression for DVS. The design utilizes a partitioned cache-like memory architecture, resulting in a low memory complexity of $O(m+n)$. We implemented the proposed design on an FPGA and conducted experiments across various downstream tasks utilizing real-world DVS data. The experimental results indicate that our PCLF design achieves a high compression rate while maintaining or even enhancing the performance of event-based vision tasks. Our design provides a cost-effective solution for real-time elimination of noise and redundant event data from DVS, thereby expanding its potential applications.

References

[1] P. Lichtensteiner, C. Posch, and T. Delbruck, "A 128x128 120db 15μs latency asynchronous temporal contrast vision sensor," *IEEE Journal of Solid-State Circuits*, no. 2, pp. 566–576, 2008.

[2] A. Amir, B. Taba, D. Berg, T. Melano, J. McKinstry, C. Di Nolfo, T. Nayak, A. Andreopoulos, G. Garreau, M. Mendoza *et al.*, "A low power, fully event-based gesture recognition system," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 7243–7252.

[3] S. Guo and T. Delbruck, "Low cost and latency event camera background activity denoising," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 1, pp. 785–795, 2022.

[4] J. Wu, C. Ma, L. Li, W. Dong, and G. Shi, "Probabilistic undirected graph based denoising method for dynamic vision sensor," *IEEE Transactions on Multimedia*, vol. 23, pp. 1148–1159, 2020.

[5] P. Zhang, Z. Ge, L. Song, and E. Y. Lam, "Neuromorphic imaging with density-based spatiotemporal denoising," *IEEE Transactions on Computational Imaging*, 2023.

[6] R. Baldwin, M. Almatrafi, V. Asari, and K. Hirakawa, "Event probability mask (epm) and event denoising convolutional neural network (edncnn) for neuromorphic cameras," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 1701–1710.

[7] Y. Alkendi, R. Azzam, A. Ayyad, S. Javed, L. Seneviratne, and Y. Zweiri, "Neuromorphic camera denoising using graph neural network-driven transformers," *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[8] A. Khodamoradi and R. Kastner, "O(n)-space spatiotemporal filter for reducing noise in neuromorphic vision sensors," *IEEE Transactions on Emerging Topics in Computing*, vol. 9, no. 1, pp. 15–23, 2021.

[9] H. Liu, C. Brandli, C. Li, S.-C. Liu, and T. Delbruck, "Design of a spatiotemporal correlation filter for event-based sensors," in *2015 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2015, pp. 722–725.

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

[11] C. Wang, X. Wang, C. Yan, and K. Ma, "Feature representation and compression methods for event-based data," *IEEE Sensors Journal*, vol. 23, no. 5, pp. 5109–5123, 2023.

[12] T. Delbruck *et al.*, "Frame-free dynamic digital vision," in *Proceedings of Intl. Symp. on Secure-Life Electronics, Advanced Electronics for Quality Life and Society*, vol. 1. Citeseer, 2008, pp. 21–26.

[13] S. Chen and M. Guo, "Live demonstration: Celex-v: A 1m pixel multi-mode event-based sensor," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. IEEE, 2019, pp. 1682–1683.

[14] A. Linares-Barranco, F. Gomez-Rodriguez, V. Villanueva, L. Longinotti, and T. Delbruck, "A USB3.0 FPGA event-based filtering and tracking framework for dynamic vision sensors," *Proceedings - IEEE International Symposium on Circuits and Systems*, vol. 2015-July, pp. 2417–2420, 2015.

[15] J. Zhu, S. Lai, X. Chen, D. Wang, and H. Lu, "Visual prompt multi-modal tracking," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2023, pp. 9516–9526.

[16] X. Wang, J. Li, L. Zhu *et al.*, "Visevent: Reliable object tracking via collaboration of frame and event flows," *IEEE Transactions on Cybernetics*, 2023.

[17] K. Kodama, Y. Sato, Y. Yorikado *et al.*, "1.22μm 35.6Mpixel RGB Hybrid Event-Based Vision Sensor with 4.88μm-Pitch Event Pixels and up to 10K Event Frame Rate by Adaptive Control on Event Sparsity," in *2023 IEEE International Solid-State Circuits Conference (ISSCC)*, 2023, pp. 92–94.

[18] G. Orchard, A. Jayawant, G. K. Cohen, and N. Thakor, "Converting static image datasets to spiking neuromorphic datasets using saccades," *Frontiers in neuroscience*, vol. 9, p. 159859, 2015.

[19] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[20] H. Li, H. Liu, X. Ji, G. Li, and L. Shi, "Cifar10-dvs: an event-stream dataset for object classification," *Frontiers in neuroscience*, vol. 11, p. 309, 2017.

[21] S. Deng, Y. Li, S. Zhang, and S. Gu, "Temporal efficient training of spiking neural network via gradient reweighting," *arXiv preprint arXiv:2202.11946*, 2022.