

# Near-Memory Architecture for Threshold-Ordinal Surface-Based Corner Detection of Event Cameras

Hongyang Shang, An Guo, Shuai Dong, Junyi Yang, Ye Ke, Arindam Basu\*

Department of Electrical Engineering, City University of Hong Kong, Hong Kong, China

**Abstract**—Event-based Cameras (EBCs) are widely utilized in surveillance and autonomous driving applications due to their high speed and low power consumption. Corners are essential low-level features in event-driven computer vision, and novel algorithms utilizing event-based representations, such as Threshold-Ordinal Surface (TOS), have been developed for corner detection. However, the implementation of these algorithms on resource-constrained edge devices is hindered by significant latency, undermining the advantages of EBCs. To address this challenge, a near-memory architecture for efficient TOS updates (NM-TOS) is proposed. This architecture employs a read-write decoupled 8T SRAM cell and optimizes patch update speed through pipelining. Hardware-software co-optimized peripheral circuits and dynamic voltage and frequency scaling (DVFS) enable power and latency reductions. Compared to traditional digital implementations, our architecture reduces latency/energy by  $24.7\times/1.2\times$  at  $V_{dd} = 1.2\text{ V}$  or  $1.93\times/6.6\times$  at  $V_{dd} = 0.6\text{ V}$  based on 65nm CMOS process. Monte Carlo simulations confirm robust circuit operation, demonstrating zero bit error rate at operating voltages above 0.62 V, with only 0.2% at 0.61 V and 2.5% at 0.6 V. Corner detection evaluation using precision-recall area under curve (AUC) metrics reveals minor AUC reductions of 0.027 and 0.015 at 0.6 V for two popular EBC datasets.

**Index Terms**—Near-Memory Computing, Event Camera, Corner Detection, Threshold-Ordinal Surface.

## I. INTRODUCTION

Event-based cameras (EBCs), also known as dynamic vision sensors (DVS), have garnered significant attention due to their exceptional low latency performance and low power consumption. Unlike conventional cameras that capture frames at fixed intervals, event cameras detect changes in contrast over time. When a pixel's contrast change exceeds a threshold, an event is triggered and an asynchronous output—the pixel's position, the timestamp, and the polarity (whether the brightness increased or decreased) is transmitted. As a result, EBCs provide higher dynamic range, lower latency, and reduced bandwidth compared to conventional CMOS image sensors (CIS) [1]. They are capable of achieving effective frame rates exceeding  $> 5000$  fps [2], while conventional video cameras are typically limited to 100 fps.

This unique capability of EBCs makes them an excellent choice for real-time applications such as autonomous driving [3], robotics [4], and unmanned aerial vehicles (UAVs) [5], which rely on fast computer vision. In computer vision, corner detection is a fundamental processing step, playing a vital

role in high-level applications such as object tracking [6], 3D reconstruction [7], and object recognition [8]. It helps identify key features within an image or video stream that are stable and distinguishable from their surroundings.

However, traditional corner detection algorithms are designed for frames. Modified versions have been proposed for EBCs, but they exhibit significant limitations when processing EBC event streams. The eHarris approach [9] retains the event by event (EBE) processing nature, achieves decent accuracy, but imposes prohibitive computational overhead (and concomitantly latency—see [10] and Fig. 1) by requiring Harris score calculations for every asynchronous event. While segment-based methods such as FAST [11] and ARC [12] improve throughput by identifying continuous segments of events, they suffer from elevated false positive rates in cluttered environments due to inherent noise sensitivity. The luvHarris algorithm [10] addresses these issues by decoupling the Harris computation via an as fast as possible, EBE update of Threshold-Ordinal Surface (TOS) and a separate frame by frame (FBF) Harris update that uses the last available TOS, as shown in Fig.1(a). However, the increasing event rate from high-resolution EBCs still gets constrained in the TOS update mechanism, as shown next.

Updating a TOS patch of  $P \times P$  using traditional digital circuit implementation leads to  $O(P^2)$  computational complexity. For a  $7 \times 7$  patch size and a 500 MHz clock frequency, the event update latency is around 392 ns, yielding an event processing rate of only  $\approx 2.6$  Meps—much below the bandwidth of most current event cameras [1] (as shown in Fig. 1(b)), leading to event loss at higher event frequencies. The Harris score calculation is not a problem for the high resolution EBCs since the major operation is a convolution which can be efficiently accelerated by the modern in-memory chips designed for convolutional neural networks (CNN). For example, Harris score calculation for a  $1280 \times 720$  resolution with  $5 \times 5$  Sobel and Harris windows requires  $\approx 236$  Mops. Given the high throughput of many recent CNN chips [13], [14], it is estimated that Harris LUT update rate  $> 1$  KHz. Therefore, although intuitively the Harris score calculation might seem more complex, the per-event update requirement of TOS makes it the bottleneck for hardware implementation, particularly for the high event rate of modern resolution event cameras [15].

To address this issue, we propose a near-memory architecture designed for efficient TOS generation following the recent

\*Corresponding author (e-mail: arinbasu@cityu.edu.hk)

This work was supported by the Research Grants Council of the HK SAR, China (Project No. CityU 11208125 and HKU C7003-24Y).

trend of using Near/In-Memory Computing for event cameras [16], [17]. As shown in Fig. 1(b), this can push the supported event rate to  $\approx 63$  Meps, compatible with high resolution EBCs. The main contributions of this work are:

- A near-memory computing architecture (NMC-TOS) for the TOS update with simplified minus one logic (MOL) and comparison modules (CMP) for reduced delay and power.
- A read-write decoupled 8T SRAM with pipelining approach to further reduce the delay required for processing a patch.
- Dynamic voltage and frequency scaling (DVFS) to adjust the chip’s voltage dynamically based on the event rate for reduced average power consumption.
- Using the above techniques, we demonstrate a speedup of  $1.9\times$  to  $24.7\times$  and reduced power consumption by a factor of  $1.2\times$  to  $6.6\times$  compared to the traditional implementation.
- System level simulations at the worst-case operating voltage of 0.6 V show a decrease of 0.027 and 0.015 in the area under the precision-recall curve (AUC) for the shapes\_dof [18] and dynamic\_dof [18] datasets, respectively.

## II. BACKGROUND

### A. Event Based Camera and Data Representation

Event cameras generate sparse event outputs that encode changes in temporal contrast, making them highly effective for high-speed, real-time applications. Each event,  $v = (v_x, v_y, v_p, v_t)$  is typically represented using the Address Event Representation (AER) protocol where the coordinates ( $v_x$  and  $v_y$ ), the polarity ( $v_p$ ), and a timestamp ( $v_t$ ) are used. The polarity indicates whether the event is triggered by an increase (ON event) or decrease (OFF event) in temporal contrast while the timestamp records the triggering time of the event.

### B. Corner Detection Methods for EBC

Since event camera data are in AER format rather than the traditional 2D image format, some algorithms have been developed specifically for corner detection based on event data.

The eHarris algorithm [9] creates a binary surface of active events indicating the occurrence of an event in the recent past. The corner is computed by evaluating the Harris score of the binary surface for each event. Although the accuracy of corner classification using the eHarris method achieves higher accuracy than ARC [12] & FAST [11], their computational complexity results in high latency since the Harris calculation has to be performed for every event.

Segment-based algorithms like FAST [11] and ARC [12] can be directly applied to the Surface of Active Events (SAE), performing calculations at a higher throughput because they do not require complex operations on the whole image. These algorithms detects corners by comparing the event timestamps in the local neighborhood around the current event. They use a circular neighborhood mask to identify continuous segments

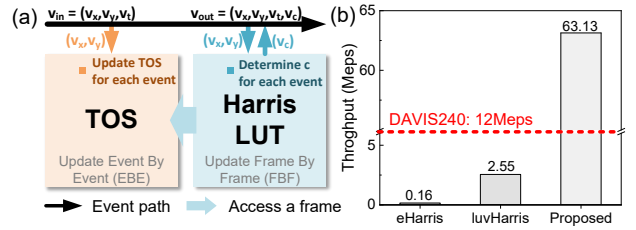


Fig. 1. (a) Architecture of luvHarris where the TOS is updated for each incoming event,  $v_{in}$ . The Harris lookup table (LUT) is updated for the full frame by accessing the TOS and  $v_{in}$  is tagged as a corner  $c$  or not by referencing the last available Harris LUT.(b) The maximum throughput of eHarris, the conventional implementation of luvHarris, and the proposed NMC-TOS compared with the maximum bandwidth of the DAVIS240 [1].

of events with higher timestamps than the surrounding pixels. If these segments form a continuous arc, the current event is classified as a corner in [11] while if the angle of this arc exceeds a certain threshold (usually 90 degrees), the event is classified as a corner in [12]. However, both FAST and ARC only utilize information from the local circular region, making them more sensitive to noise. This sensitivity can lead to high false positive rates [10].

The luvHarris algorithm [10] aims to improve corner detection throughput while maintaining high accuracy. To achieve this, luvHarris introduces two key innovations: first, it proposes a variant of the SAE called the Threshold-Ordinal Surface (TOS), which uses 8-bit unsigned integers to represent the novelty of the recorded events. Second, luvHarris decouples the event-by-event TOS update from the frame-based Harris score calculation, and uses the last computed Harris score as a look up table (LUT). However, the method is demonstrated on high-end, multithreaded processors and not suited for edge applications. Simpler processors can achieve limited throughput due to the TOS update per event as shown in Section I.

## III. OVERALL ARCHITECTURE

Fig. 2(a) illustrates the overview of the corner detection system, which includes the blocks of Spatio-temporal Correlation Filtering (STCF), DVFS, TOS and corner detector.

### A. Spatio-Temporal Correlation Filtering Denoising

The Spatio-Temporal Correlation Filtering (STCF) method is designed to reduce background activity (BA) noise in EBCs [19]. BA noise events are typically isolated, while signal events occur in groups of temporally and spatially correlated events. The STCF filter checks the event count within a spatio-temporal neighborhood defined by a time window ( $TW_{STCF}$ ). If enough supporting events (e.g., 2) are present, it is classified as a signal, else, it is classified as noise.

### B. Dynamic Voltage and Frequency Scaling

Since event cameras capture only temporal contrast changes per pixel which is dependent on the scene, the event rate fluctuates rather than being fixed like the frame rate of traditional cameras. This has been historically exploited by designing

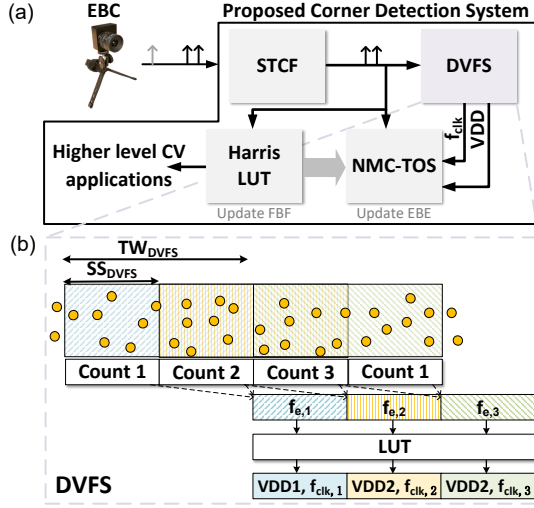


Fig. 2. The workflow for corner detection on the event stream of an EBC. The events received by the EBC first pass through an STCF filter to remove noise. Then, the TOS is constructed by NMC-TOS EBE, while the event frequency is detected using the DVFS module. Finally, corner detection is performed FBF.

asynchronous hardware [20], [21], but the lack of publicly available asynchronous design tools makes its adoption difficult. Instead, we propose to dynamically adjust the clock frequency based on the event rate, a method compatible with traditional synchronous design. By using lower voltage during low event rates and adjusting the clock to a lower frequency, the energy consumption for each event when updating the TOS is effectively reduced. As shown in Fig. 2(b), a moving window averaging is employed to count the number of events with window size  $TW_{DVFS}$  and stride  $SS_{DVFS}$  (we fix stride to be 50% in this work). The value of  $TW_{DVFS}$  and the bitwidth of the counters depend on application; for driving datasets [22],  $TW_{DVFS} = 10$  ms and bitwidth of 20 were sufficient.

A round-robin counter mechanism is employed to perform the counting operation (Fig. 2(b)). The method involves three counters working in sequence, such that when one counter is counting, the other two are used to provide an estimate of the event rate  $f_e$ . As shown in Fig. 2(b), each counter's duration is fixed to  $\frac{TW_{DVFS}}{2}$ . A pointer  $ptr$  determines the current counter accepting events and its position is shifted among the three counters in a circular sequence given by:  $ptr \leftarrow (ptr + 1) \bmod(3)$ . The event frequency  $f_e$  is mapped to the required dynamic voltage ( $VDD$ ) and clock frequency ( $f_{clk}$ ) via a LUT.

### C. NMC-TOS and Harris Based Corner Detector

TOS uses 8-bit unsigned integers to represent the novelty of events. Details of the algorithm are shown in Algorithm 1. Whenever a new event  $v$  occurs, the TOS updates the corresponding patch data by subtracting 1 (where the patch size is  $P$ ) and applies a threshold with a fixed value, TH. Then, the pixel location is updated to 255. Hardware implementation of TOS using near memory computing (NMC-TOS) is illustrated

### Algorithm 1 Event-by-event TOS update

```

Require:  $v = \langle v_x, v_y, v_t \rangle, TOS$ 
for  $x = v_x - \frac{P-1}{2} : v_x + \frac{P-1}{2}$ 
  for  $y = v_y - \frac{P-1}{2} : v_y + \frac{P-1}{2}$ 
     $TOS_{xy} \leftarrow TOS_{xy} - 1$ 
    if  $TOS_{xy} < TH$  then  $TOS_{xy} \leftarrow 0$ 
   $TOS_{v_x v_y} \leftarrow 255$ 

```

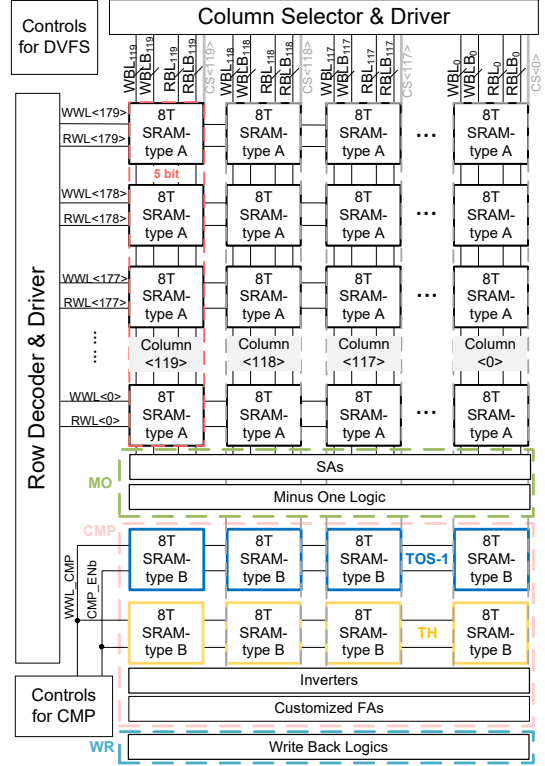


Fig. 3. The overall block architecture of NMC-TOS is divided; an EBC like DAVIS240 with resolution  $240 \times 180$  requires two such blocks. Each block of the TOS array consists of 180 rows and 120 columns of 5-bit words. The peripheral circuits include MO module, CMP module, WR module, buffer, and control circuits.

in section IV. The standard Harris operator in run on this TOS representation to detect corners [10].

## IV. NEAR-MEMORY COMPUTING FOR TOS

### A. Hardware Overview

To optimize the TOS memory size, we note that although the maximum TOS value is 255 requiring 8 bits for storage, the threshold typically does not go below  $\approx 225$  in practice. As a result, the highest 3 bits can be omitted from on-chip storage to save space, and each TOS is stored using only 5 bit words. The architecture uses a block size of  $180 \times 600$  8T SRAM (type A) array, as shown in Fig. 3, which corresponds to storing 5-bit words for a sensor resolution of  $180 \times 120$ . This block size can be repeated as many times as needed to accommodate different resolution cameras. For example, two such blocks are required to cover the entire sensor for a DAVIS240 sensor

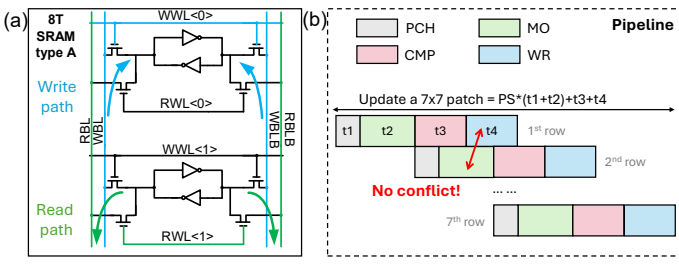


Fig. 4. 8T SRAM-type A and the pipeline method. (a) Decoupling the WBL and RBL makes it possible for the write-back and read operations to occur simultaneously. (b) A pipeline example for updating a  $7 \times 7$  patch.

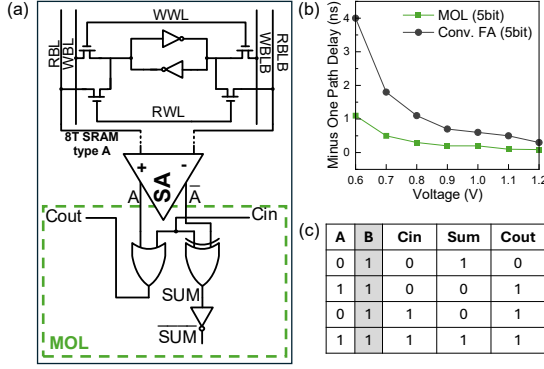


Fig. 5. Minus one (MO) module. (a) MO module SA for readout of SRAM and simplified minus one logic (MOL). (b) MOL reduces path delay compared to 28T full adders (FA). (c) Truth table of MOL.

whose resolution is  $240 \times 180$ . The overall architecture of a single block, as shown in Fig. 3, includes the DVFS control module, row/column selectors, drivers, a  $180 \times 600$  8T SRAM (type A) array, the minus one (MO) module, the compare (CMP) module, and the write-back (WR) module. The MO module consists of the sense amplifier (SA) and the minus one logic (MO), while the CMP module includes the  $2 \times 600$  8T SRAM (type B), inverters, and the customized full adder (FA).

### B. 8T SRAM for Pipeline

For the SRAM array that stores the TOS value, a read-write decoupled 8T SRAM architecture is employed, as shown in Fig. 4(a). This SRAM structure separates the read bitline (RBL) and write bitline (WBL) by replicating two access NMOS transistors, allowing reading and writing to occur simultaneously. Assuming the patch size is  $P$ , without this architecture, the reading of the next row must occur after the write-back of the previous row, which introduces a delay of  $(t_1 + t_2 + t_3 + t_4) \times P$  where the  $t_i$  correspond to the delays of four phases of operation as shown in Fig. 4(b). However, with the 8T SRAM architecture, pipelining can be used for the multiple computation steps, compressing the required computational delay to  $P \times (t_1 + t_2) + t_3 + t_4$ , which decreases the delay by about  $2 \times$ .

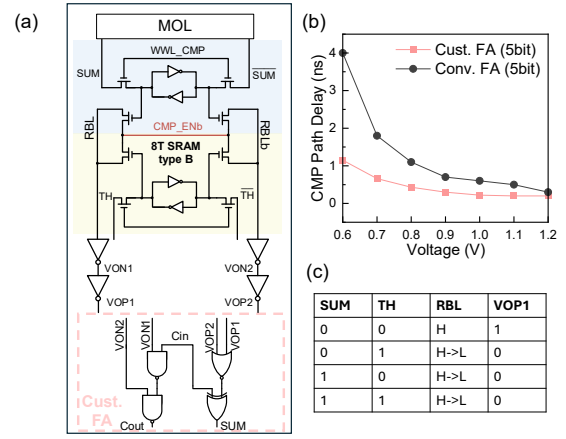


Fig. 6. Comparison (CMP) module. (a) CMP consists of two rows of 8T SRAM type B (one stores the MOL result and the other stores threshold (TH)), inverters for readout and customized FAs. (b) Customized FA reduces path delay compared to 28T FA. (c) Operation principle of CMP.

### C. Minus One (MO) Module and Comparison (CMP) Module

For the decrement by 1 operation in neighbouring pixels, the MO module is shown in Fig. 5(a). The TOS stored in 8T SRAM type A is read out by a latched SA [23], and directly sent to the simplified minus one logic (MOL). Compared to the conventional 28T full adder, the computational delay is smaller, as shown in 5(b). Since one addend B is always -1 (all bits are 1 in 2's complement), the full adder is simplified to avoid the logical redundancy typically introduced by using a 28T full adder in traditional designs [24], as shown in the truth table of Fig. 5(c).

For the CMP module, another type of 8T SRAM is adopted as shown in Fig. 6(a), with one row used to store the SUM output of the MOL (which is TOS-1) and another row to store the threshold (TH). The computation method of this architecture is an NOR operation (as shown in Fig. 6(c)), meaning that the RBL only remains high if both of SUM and TH at the same bit significance are stored as 0; otherwise, RBL drops to 0 V. Notice that, RBL in the CMP module is decoupled from the big capacitance of SRAM array's RBL. As a benefit, this SRAM allows RBL to achieve full swing with only two rows for computation. Hence, inverters can be used to determine the status of RBL and RBLB, significantly reducing power consumption compared to SA. Based on the characteristics of this NOR computation, the subsequent custom full adder is implemented instead of a traditional full adder [25], which further reduces the delay as demonstrated in Fig. 6(b).

### D. Detailed operation of the NMC Architecture

The waveform of the NMC-TOS is shown in Fig. 7. For different operating voltage given by the DFVS module, the required clock cycles of the four phases are same, but the clock frequency is different. As described in Sec. IV-B, the NMC-TOS primarily involves four working stages: PCH, MO, CMP, and WR. These stages are performed sequentially for each row

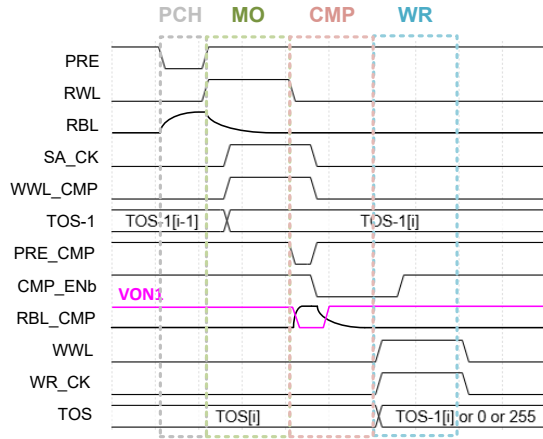


Fig. 7. The overall timing diagram of one row calculation in NMC-TOS comprising precharge (PCH), minus-one (MO), comparison (CMP) and write back (WR).

in the patch. Initially, the PRE signal triggers a low pulse to precharge the RBL of the SRAM-type A array, preparing for the subsequent reading of the stored TOS. Then, during the MO stage, the RWL signal rises, and if the TOS stores a 0, the RBL will decrease. At this point, the SA clock (SA\_CK) arrives slightly later to ensure setup time. Meanwhile, the WWL\_CMP is enabled, allowing the result of the MO to be written to the TOS-1 SRAM in the CMP module. During this phase, the precharge of RBL\_CMP (PRE\_CMP) is also occurring for the next calculation. In the next CMP stage, the arrival of CMP\_ENb causes the voltage on RBL\_CMP to change, and VON/VOP is output accordingly (as shown by the pink line in the figure). Finally, in the WR stage, WR\_CK controls the DFF to latch the write back value (TOS-1 or 0 or 255) and WWL is enabled to write the value back into the main TOS SRAM. The value latched by the DFF is determined by the Cout signal from the most significant bit of the CMP module.

## V. RESULTS

The NMC macro was implemented in a 65 nm CMOS process and SPICE simulation results are reported next.

### A. Dynamic Voltage Frequency Scaling

The effect of DVFS is shown in Fig. 8 for 111.4M number of events from the driving dataset [22]. It can be observed that the event rate sampled by the 20-bit counter, sensitively captures changes in the event rate. Additionally, the DVFS module dynamically adjusts the operating voltage and maximum event frequency of the NMC-TOS based on the sampled rate. Throughout the entire dataset, the event rate never reached the maximum operating frequency of 63.1 Meps at 1.2V, indicating that the NMC-TOS can continuously update the TOS without any event loss during operation in this dataset. Table I summarizes the event rates and corresponding power savings for different datasets [18], [22] with 1.4× to 5.3× savings achievable.

TABLE I  
POWER IMPROVEMENT USING DVFS UNDER SEVERAL DATASETS

| Dataset          | Max Event Rate (Meps) | # Events (M) | Power w DVFS (mW) | Power w/o DVFS (mW) |
|------------------|-----------------------|--------------|-------------------|---------------------|
| driving [22]     | 25.9                  | 111.4        | 0.44              | 1.24                |
| laser [22]       | 39.5                  | 57.6         | 3.90              | 5.37                |
| spinner [22]     | 11.4                  | 54.1         | 0.38              | 1.50                |
| dynamic_dof [18] | 4.5                   | 57.1         | 0.02              | 0.13                |
| shapes_dof [18]  | 1.9                   | 18.0         | 0.01              | 0.04                |

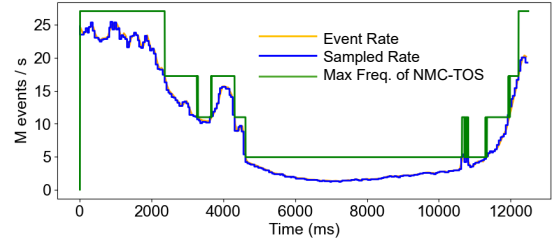


Fig. 8. The effect of DVFS in the driving dataset [22] shows the proposed method can modify the maximum event handling capacity according to fluctuations in event rate.

### B. Latency and Energy Comparisons

As shown in Fig. 9(a), the blue region represents the results of NMC-TOS for updating a patch, considering the effects of pipeline and the varied operating voltage from 1.2 V to 0.6 V. It can be observed that at an operating voltage of 1.2 V, the latency is the lowest (16 ns), while the energy consumption is relatively high (139 pJ). As the operating voltage decreases to 0.6 V, latency increases to 203 ns, but energy consumption decreases to 26 pJ. From Fig. 9(b), it can be seen that using the NMC architecture reduces the latency by 13.0× compared to the conventional method, and employing a pipeline further reduces the latency by 24.7×. Regarding the normalized energy (as shown in Fig.9(c)), the NMC architecture reduces energy consumption by 1.2× compared to the conventional method. Moreover, using DVFS at the most aggressive setting (operating at 0.6 V) allows for a further reduction in energy consumption by 6.6×.

Detailed power breakdown shown in Fig. 10(a) indicates that the peripheral circuits (PP) is the main source of power consumption, accounting for 45.9%, which is due to the placement of both the computation and write-back modules in the peripheral circuits. Additionally, the array account for 31.9%. The driver and the SA used for SRAM array reads contribute 11.6% and 10.6%, respectively. For different event rates, it can be observed from Fig. 10(b) that the NMC method consistently outperforms the conventional method, and the use of DVFS further reduces power consumption. For example, at the event rate of 45 Meps, NMC reduces power dissipation by 1.2× compared to the conventional one while adding DVFS to it provides a further reduction of 1.37×.

In terms of delay, it is evident in Fig.10(c) that the minus one (MO) phase exhibits the largest delay, accounting for 30.6%. The delays for the other phases are as follows: pre-charge (PCH) accounts for 13.9%, computation (CMP)

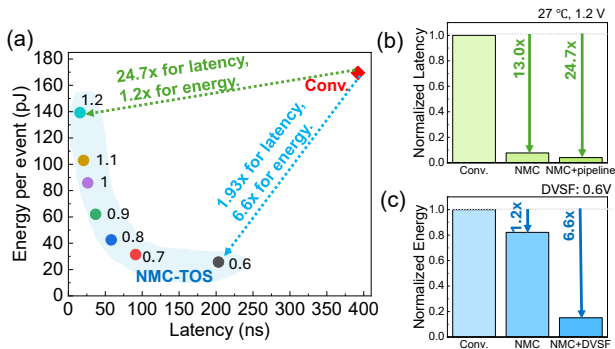


Fig. 9. (a) Energy consumption and latency of the conventional digital circuit method and the NMC-TOS architecture at different operating voltages. (b) Impact of using NMC method and NMC+pipeline on latency. (c) Impact of using NMC method and NMC+DVFS on energy consumption.

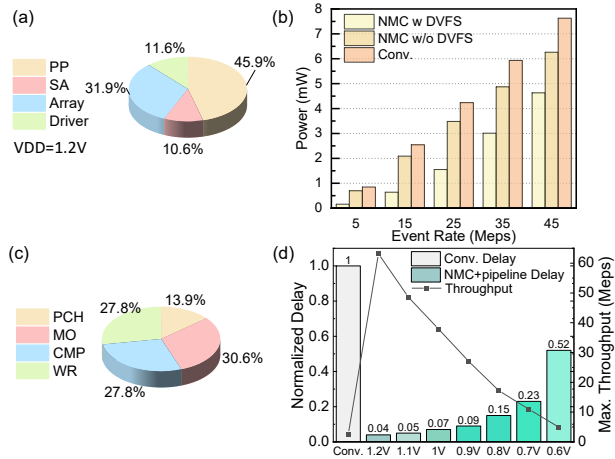


Fig. 10. Analysis of power, delay and throughput in NMC-TOS. (a) Breakdown of energy consumption for each module at an operating voltage of 1.2V. (b) Comparison of power consumption at different event frequencies. (c) Delay of different computation phases at an operating voltage of 0.6V. (d) Comparison of computational latency per event and maximum throughput that can be handled by conventional methods and NMC-TOS.

accounts for 27.8%, and write-back (WR) accounts for 27.8%. As shown in Fig. 10(d), it is observed that at higher operating voltages, the computation delay decreases and throughput increases. Compared to the conventional method, updating a  $7 \times 7$  TOS patch benefits from the parallelism of the NMC architecture, and the addition of a pipeline further reduces the delay. Compared to the traditional method's throughput of 2.6 Meps, NMC-TOS achieves a throughput of 63.1-4.9 Meps. Even at the lowest operating voltage of 0.6V, the throughput is improved by  $1.9 \times$  over the conventional one.

### C. Impact of Hardware Non-Ideality on the Algorithm

To estimate errors due to hardware non-idealities, Monte Carlo simulations were run to estimate BER which showed errors only occur below 0.62 V, with a BER of 0.2% at 0.61V and 2.5% at 0.6 V. In the write-back circuit, when the value stored in the original TOS memory is 0, the write-back is disabled, meaning that the error occurs only in pixels

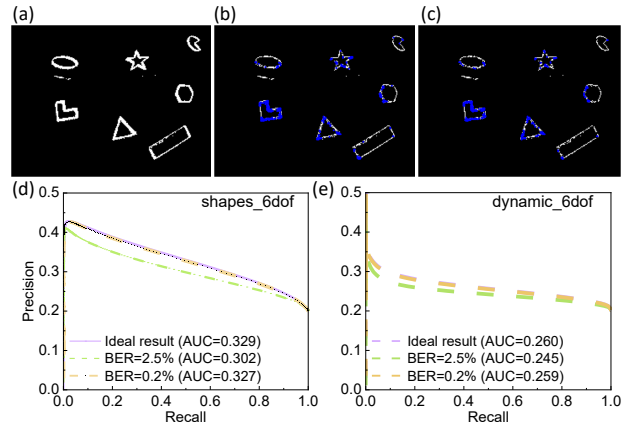


Fig. 11. Visualizing 5000 events from the shapes\_dof dataset. (a) SAE representation of timestamps. (b) TOS without precision loss and the detected corner. (c) TOS considering BER at 0.6V and the detected corner. (d) Precision-Recall results on the (d) shapes\_dof and (e) dynamic\_dof.

with valid values. Additionally, since the highest 3 bits are omitted as mentioned in IV-A, the erroneous values range between 224 and 255, significantly reducing the impact on the computation. We analyzed the effect of the worst-case BER on the computation by injecting errors in a software simulation of the pipeline on shapes\_dof dataset [18]. Fig. 11 compares SAE [11] using timestamps as grayscale values in (a), error-free TOS with corner results in (b), TOS considering hardware errors with corner results in (c). It can be observed that the hardware errors do not significantly affect the results visually.

To quantitatively assess the impact of errors, we compared precision-recall curves for the corner detection results as done in [10]. As shown in Fig.11(c) and (d), it can be observed that, on the shapes\_dof dataset, when the BER is 2.5%, the AUC decreases by only about 0.027 and remains unchanged when the BER is 0.2%. For the dynamic\_dof dataset, the AUC decreased by only 0.015 at a BER of 2.5% and is again almost unchanged at a BER of 0.2%. This shows possibility of using DVFS with power supplies down to 0.6 V in our pipeline.

## VI. CONCLUSION

This work addresses the issues of low throughput and high power consumption in state-of-the-art corner detection algorithms for event cameras using TOS. Implementation of earlier hardware architectures in edge devices result cannot keep up with the throughput of current high-resolution EBCs. Therefore, a near-memory computing architecture is employed to accelerate the TOS update algorithm on the peripheral circuit. By utilizing an 8T SRAM with read-write decoupling, pipelining can be used resulting in a delay reduction of up to  $24.7 \times$  compared to traditional cases. Additionally, DVFS is applied to dynamically adjust the operating voltage based on event rates, achieving  $6.6 \times$  reduction in energy compared to traditional digital implementations. Using system level simulations of corner detection, we show that at the lowest power supply of 0.6 V, the decrease in AUC is only 0.027 and 0.015 for shapes\_dof and dynamic\_dof, respectively.

## REFERENCES

- [1] G. Gallego, T. Delbrück, G. Orchard, C. Bartolozzi, B. Taba, A. Censi, S. Leutenegger, A. J. Davison, J. Conradt, K. Daniilidis *et al.*, “Event-based vision: A survey,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 44, no. 1, pp. 154–180, 2020.
- [2] H. Rebecq, R. Ranfil, V. Koltun, and D. Scaramuzza, “High speed and high dynamic range video with an event camera,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 43, no. 6, pp. 1964–1980, 2019.
- [3] G. Chen, H. Cao, J. Conradt, H. Tang, F. Rohrbein, and A. Knoll, “Event-based neuromorphic vision for autonomous driving: A paradigm shift for bio-inspired visual sensing and perception,” *IEEE Signal Processing Magazine*, vol. 37, no. 4, pp. 34–49, 2020.
- [4] F. Mahlknecht, D. Gehrig, J. Nash, F. M. Rockenbauer, B. Morrell, J. Delaune, and D. Scaramuzza, “Exploring event camera-based odometry for planetary robots,” *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 8651–8658, 2022.
- [5] W. Shariff, M. S. Dilmaghani, P. Kilty, M. Moustafa, J. Lemley, and P. Corcoran, “Event cameras in automotive sensing: A review,” *IEEE Access*, 2024.
- [6] F. Du, P. Liu, W. Zhao, and X. Tang, “Correlation-guided attention for corner detection based visual tracking,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 6836–6845.
- [7] N. D. Reddy, M. Vo, and S. G. Narasimhan, “Carfusion: Combining point tracking and part detection for dynamic 3d reconstruction of vehicles,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 1906–1915.
- [8] M. Bansal, M. Kumar, M. Kumar, and K. Kumar, “An efficient technique for object recognition using Shi-Tomasi corner detection algorithm,” *Soft Computing*, vol. 25, no. 6, pp. 4423–4432, 2021.
- [9] V. Vasco, A. Glover, and C. Bartolozzi, “Fast event-based Harris corner detection exploiting the advantages of event-driven cameras,” in *2016 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2016, pp. 4144–4149.
- [10] A. Glover, A. Dinale, L. D. S. Rosa, S. Bamford, and C. Bartolozzi, “Iuvharris: A practical corner detector for event-cameras,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 12, pp. 10087–10098, 2021.
- [11] E. Mueggler, C. Bartolozzi, and D. Scaramuzza, “Fast event-based corner detection,” 2017.
- [12] I. Alzugaray and M. Chli, “Asynchronous corner detection and tracking for event cameras in real time,” *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3177–3184, 2018.
- [13] B. Zimmer, R. Venkatesan, Y. S. Shao, J. Clemons, M. Fojtik, N. Jiang, B. Keller, A. Klinefelter, N. Pinckney, P. Raina *et al.*, “A 0.32–128 TOPS, scalable multi-chip-module-based deep neural network inference accelerator with ground-referenced signaling in 16 nm,” *IEEE Journal of Solid-State Circuits*, vol. 55, no. 4, pp. 920–932, 2020.
- [14] K. Matsubara, H. Lieske, M. Kimura, A. Nakamura, M. Koike, S. Morikawa, Y. Hotta, T. Irita, S. Mochizuki, H. Hamasaki *et al.*, “A 12-nm autonomous driving processor with 60.4 TOPS, 13.8 TOPS/W CNN executed by task-separated ASIL d control,” *IEEE Journal of Solid-State Circuits*, vol. 57, no. 1, pp. 115–126, 2021.
- [15] Sony and Prophesee, “Sony-Prophesee IMX636 Event-Based Vision Sensor,” <https://www.prophesee.ai/event-based-sensor-imx636-sony-prophesee/>, 2024, accessed: 2024-07-18. [Online]. Available: <https://www.prophesee.ai/event-based-sensor-imx636-sony-prophesee/>
- [16] S. K. Bose and A. Basu, “A 389 TOPS/W, always ON region proposal integrated circuit using in-memory computing in 65 nm CMOS,” *IEEE Journal of Solid-State Circuits*, vol. 58, no. 2, pp. 554–568, 2022.
- [17] X. Zhang and A. Basu, “A 915–1220 TOPS/W, 976–1301 GOPS hybrid in-memory computing based always-on image processing for neuromorphic vision sensors,” *IEEE Journal of Solid-State Circuits*, vol. 58, no. 3, pp. 589–599, 2022.
- [18] E. Mueggler, H. Rebecq, G. Gallego, T. Delbruck, and D. Scaramuzza, “The event-camera dataset and simulator: Event-based data for pose estimation, visual odometry, and SLAM,” *The International journal of robotics research*, vol. 36, no. 2, pp. 142–149, 2017.
- [19] 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.
- [20] F. Akopyan, J. Sawada, A. Cassidy, R. Alvarez-Icaza, J. Arthur, P. Merolla, N. Imam, Y. Nakamura, P. Datta, G.-J. Nam *et al.*, “Truenorth: Design and tool flow of a 65 mw 1 million neuron programmable neurosynaptic chip,” *IEEE transactions on computer-aided design of integrated circuits and systems*, vol. 34, no. 10, pp. 1537–1557, 2015.
- [21] M. Davies, N. Srinivasa, T.-H. Lin, G. Chinya, Y. Cao, S. H. Choday, G. Dimou, P. Joshi, N. Imam, S. Jain *et al.*, “Loihi: A neuromorphic manycore processor with on-chip learning,” *Ieee Micro*, vol. 38, no. 1, pp. 82–99, 2018.
- [22] Prophesee, “Recordings and Datasets,” 2025, accessed: 2025-07-02. [Online]. Available: <https://docs.prophesee.ai/stable/datasets.html#datasets>
- [23] C. Yu, T. Yoo, H. Kim, T. T.-H. Kim, K. C. T. Chuan, and B. Kim, “A logic-compatible eDRAM compute-in-memory with embedded ADCs for processing neural networks,” *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 68, no. 2, pp. 667–679, 2020.
- [24] A. Guo, C. Xi, F. Dong, X. Pu, D. Li, J. Zhang, X. Dong, H. Gao, Y. Zhang, B. Wang *et al.*, “A 28-nm 64-kb 31.6-TFLOPS/W digital-domain floating-point-computing-unit and double-bit 6T-SRAM computing-in-memory macro for floating-point CNNs,” *IEEE Journal of Solid-State Circuits*, vol. 59, no. 9, pp. 3032–3044, 2024.
- [25] Y. Chen, J. Mu, H. Kim, L. Lu, and T. T.-H. Kim, “BP-SCIM: A reconfigurable 8T SRAM macro for bit-parallel searching and computing in-memory,” *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 70, no. 5, pp. 2016–2027, 2023.