REMOT: A Hardware-Software Architecture for Attention-Guided Multi-Object Tracking with Dynamic Vision Sensors on FPGAs

Yizhao Gao*
yzgao@eee.hku.hk
University of Hong Kong
Hong Kong

Song Wang*
wangsong@eee.hku.hk
University of Hong Kong
Hong Kong

Hayden Kwok-Hay So
hso@eee.hku.hk
University of Hong Kong
Hong Kong

ABSTRACT

In contrast to conventional vision sensors that produce images of the entire field-of-view at a fixed frame rate, dynamic vision sensors (DVS) are neuromorphic devices that only produce sparse events in response to changes in light intensity local to each pixel, making them promising technologies for use in demanding edge scenarios where energy-efficient intelligent computations are needed. While several early research have demonstrated promising results in performing high-level machine vision tasks using vision events only, these algorithms are often too complex for real-time deployments in edge systems with limited processing and storage capabilities. In this work, a novel hardware-software architecture, called REMOT, is proposed to leverage the unique properties of DVS to perform real-time multi-object tracking (MOT) on FPGAs. REMOT incorporates a parallel set of reconfigurable hardware attention units (AUs) that work in tandem with a modular attention-guided software framework running in the attached processor. Each hardware AU autonomously adjusts its region of attention by processing each vision event as they are produced by the DVS. Using information aggregated by the AUs, high-level analyses are performed in software. To demonstrate the flexibility and modularity of REMOT, a family of MOT algorithms with different hardware-software configurations and tradeoffs have been implemented on 2 different edge reconfigurable systems. Experimental results show that REMOT is capable of processing 0.43–2.22 million events per second at 1.75–5.68 watts, making them suitable for real-time operations while maintaining good MOT accuracy in our target datasets. When compared with a software-only implementation using the same edge platforms, our HW-SW implementation results in up to 33.6 times higher event processing throughput and 25.9 times higher power efficiency.

CCS CONCEPTS

• **Computer systems organization** → **Real-time system architecture**; **Reconfigurable computing**; • **Computing methodologies** → **Tracking**.

*Both authors contributed equally to this research.

FPGA '22, February 27-March 1, 2022, Virtual Event, CA, USA

ACM ISBN 978-1-4503-9149-8/22/02.
https://doi.org/10.1145/3490422.3502365

KEYWORDS

dynamic vision sensors; multi-object tracking; event sensors; event camera; hardware/software co-design; attention unit; FPGA; HOTA

ACM Reference Format:
Yizhao Gao, Song Wang, and Hayden Kwok-Hay So. 2022. REMOT: A Hardware-Software Architecture for Attention-Guided Multi-Object Tracking with Dynamic Vision Sensors on FPGAs. In *Proceedings of the 2022 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA '22), February 27-March 1, 2022, Virtual Event, CA, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3490422.3502365

1 INTRODUCTION

Dynamic vision sensors (DVS), also called event cameras, or silicon retinas, are neuromorphic vision sensing devices that have been receiving renewed interests in recent years due to their promised energy-efficiency advantages for use in edge applications such as intelligent transportation systems, internet-of-thing sensing and autonomous vehicles guidance [15]. Sometimes referred as neuromorphic vision sensors, a DVS detects *changes* in exposed light intensity and reports them asynchronously as spiking events localized to the pixels involved. This is in contrast to a conventional image sensor where light intensity values of every pixel of a frame are reported synchronously at a regular interval. See Fig. 1 and Section 2 for further details.

On one hand, since events are sparse and are produced only for times when and at locations where there are activities, DVS provide unique opportunities for energy-efficient hardware processing. For instance, a system may choose to conserve energy by running in low power mode until sufficient events have occurred. On the other hand, the asynchronous sensing of DVS produces bursts of events that require high throughput processing to avoid excessive buffering and to maintain low processing latency (See Fig. 2). Depending on the activity level and the resolution of the camera, up to millions of events per second may be produced by the sensors, making it particularly challenging for embedded processors with limited computing power and memory storage to process in real time.

Further adding to this processing challenge is that the events produced by DVS are fragmented with limited visual information when compared to the images produced by a conventional frame-based sensor. As a result, it remains a great challenge to perform even common computer vision tasks such as object classification and object tracking using only event output. Many existing event-based vision algorithms have been relying on techniques such as time-surface constructions to serve as an intermediate representation on which complex learning-based algorithms may be built upon [20, 36]. Although they have achieved promising preliminary

results, performing such complex algorithms in edge devices in real time remains an open challenge.

In this work, a novel attention-guided hardware-software architecture is proposed to address the computational challenges of DVS while facilitating codesign of high-level event-based computer vision tasks. The proposed architecture, called REMOT, is designed around the concept of an attention unit (AU). In a REMOT system, a layer of parallel AUs is implemented on the FPGA hardware, which collectively process the stream of asynchronous events from a DVS in situ as they are produced. Leveraging the inherently localized nature of DVS events, each AU autonomously tracks a subset of the events that fall under its region of attention (ROA). Using the aggregated information about ROA, high-level computer vision algorithms are developed by querying and manipulating the operations of AUs through a structured software framework.

To demonstrate the flexibility of the REMOT architecture, a family of attention-guided multi-object tracking (MOT) algorithms for DVS has been implemented with a wide range of hardware and software configurations. Furthermore, the same REMOT architecture has been implemented on two different FPGA platforms to demonstrate its portability and scalability, as well as to perform design space exploration studies. Our results show that the proposed architecture is scalable and is capable of processing 0.58 to 2.2 million events per second (Meps) while consuming 1.75 to 5.68 watts of system power. With regard to MOT accuracy, we show that our proposed attention-guided MOT algorithms can achieve 43.1 % to 73.2 % in terms of HOTA metric across a range of datasets. To the best of our knowledge, this is the first work that utilizes hardware-software codesign strategies to support high-level event-based computer vision tasks implemented efficiently on FPGAs. To this end, we consider the main contributions of this work as:

- We proposed a first-of-its-kind attention-guided hardware-software architecture that can effectively support real-time FPGA implementation of event-based computer vision tasks in edge applications;
- We demonstrated the flexibility, scalability, and real-time performance of the proposed architecture by performing design space exploration for implementations on two FPGA-based edge platforms;
- We proposed a family of real-time event-based attention-guided multi-object tracking algorithms that run on our proposed architecture and demonstrated their efficacy using a set of real-world traffic monitoring data.

In the next section, background and related work on DVS and multi-object tracking will first be discussed. The hardware-software architecture and our attention-guided MOT algorithms will be discussed in Section 3. An extensive evaluation of our proposed hardware-software system will be shown in Section 4. Limitations of our current system will be discussed in Section 5 and we will conclude in Section 6.

2 BACKGROUND & RELATED WORK

2.1 Dynamic Vision Sensors

Dynamic Vision Sensors only report local brightness change asynchronously for each pixel. Whenever the change in log intensity of a pixel is higher than a predefined threshold, it emits an event,

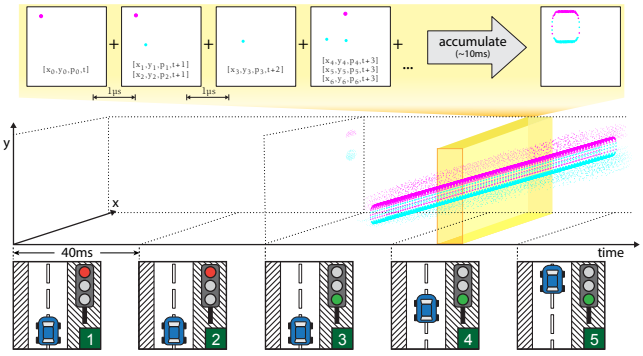


Figure 1: Comparing working principles of DVS cameras (middle) and conventional frame-based cameras (bottom). Event is encoded with its pixel location (x, y), polarity p (shown as magenta and cyan dots), and is typically timestamped at 1 µs intervals, t. (top right) Accumulating events forms a 2-D representation in the original x-y coordinate.

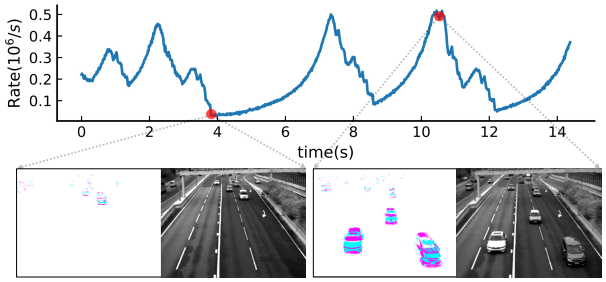


Figure 2: The number of events produced by a DVS camera depends on the amount of dynamic activities in the scene.

or spike, which is usually encapsulated in an address event representation (AER) format for downstream processing [15]. A typical event in AER can be written as $[x, y, p, t]$, where x, y is the location of the event, p is the polarity of brightness change in ± 1, and t is the timestamp generated by the sensor.

Fig. 1 illustrates the working principle of a dynamic event sensor (DVS) by showing outputs of a DVS camera alongside a conventional camera that is monitoring a hypothetical traffic intersection. The figure shows the period before and after the stopped vehicle resumes motion upon traffic light turning into green. At the lower half of the figure, 5 frames from the conventional camera are shown, which capture the scene at a regular interval, e.g. 25 ms. The first 2 frames capture *identical* and *redundant* information when the vehicle is idle. After frame 3, when the traffic light turns green, the frame-based camera continues to capture the motion of the car at disjoint locations. On the other hand, a DVS reports changes in light intensity *asynchronously* as spiking events shown as magenta and cyan dots in the middle of Fig. 1. Conversely, no event is generated when there is no change in light intensity, such as when the vehicle is idle, and when the traffic lights stay unchanged. Once the car starts moving, a very dense cloud of events can be observed that closely tracks the movement of the vehicle.

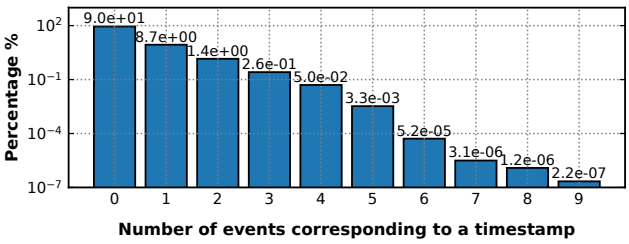


Figure 3: Event statistics of inbound traffic dataset.

Due to its asynchronous behavior, a variable data rate can be expected from DVS depending on the activity level of the scene. Fig. 2 shows the rate of events produced over time in a traffic scene. As the cars move towards the camera, the relative speed in the view increases, which results in higher events rates. Common industrial DVS timestamps events in 1 µs resolution. However, in some active dynamic scenes, more than one events might share a same timestamp value. Fig. 3 shows the statistics in our traffic dataset. The special column labeled as "0" corresponds to the percentage of time when no event is generated, which captures its sparsity in the time dimension. In this particular example, the DVS was idle 90 % of the time. Furthermore, among all the timestamps with events, 83 % contains only 1 event. On average, the data rate of our current dataset ranges from 0.22 to 0.3 million events per second (Meps). This rate ultimately determines the minimum average processing throughput our proposed hardware-software system must achieve to avoid dropping of events.

In this work, we employed an advanced Dynamic and Active Pixel Vision Sensor (DAVIS) camera [7] that implements both a dynamic vision sensor and a conventional frame-based active pixel sensor (APS) on the same pixel array. Since the two sensors are integrated at pixel level, no image registration is needed between the events and frame output. We took advantage of this feature to produce ground truth bounding boxes in our dataset.

2.2 Hardware Processing of DVS Output

The asynchronous and sparse natures of DVS output brings both opportunities and challenges to processing them efficiently in hardware. Table 1 shows a list of previous works that demonstrated efficient hardware processing of DVS output. Depending on the operating principle of a work's main algorithm, two different performance measurements have typically been used in the literature regarding DVS processing in hardware. Designs with their main algorithms operating on raw event input from the DVS usually emphasize their event processing throughput as measured in million events per second (Meps). These works are marked with a check mark in the table under the "Per-event processing" column. On the other hand, a group of event-based hardware algorithms performed computer vision tasks using frame-like intermediate representations such as by aggregating events over a time window. In these cases, the literature typically reports performance in terms of frames per second (fps).

Owing to the neuromorphic nature of DVS, a number of works have explored the use of spiking neural networks (SNN) [18] to perform dynamic vision tasks including object classification [10, 25], and object tracking [34, 35]. Subsequently, from a hardware implementation perspective, both dedicated SNN chip [35], or general purpose SNN accelerators including Intel Loihi [11] and IBM TrueNorth [27] have been used to accelerate the corresponding SNN inference task, resulting in highly energy-efficient processing in general.

At the same time, another school of work approached the challenge of performing event-based vision tasks by developing custom architecture and algorithms that operate on the DVS events natively. For instance, [23] developed a real-time object tracking system based on center-of-mass computation using FPGA for object tracking. In [37], a hand-gesture recognition system with real-time FPGA implementation by using a hierarchy of time-surface was proposed. In [33], object classification and detection was performed by mapping and categorizing the input events using PCA-RECT transform on FPGA.

Recently, leveraging their extraordinary success in processing conventional frame-based images, deep learning methods that utilize convolutional neural networks (CNNs) have also been exploited to process DVS output for various dynamic vision tasks [24, 30, 31]. Unfortunately, typical CNN accelerators are designed to operate with dense tensors and thus cannot fully take advantage of the sparseness of DVS output to improve power efficiency. To address that, some progress has been made in accelerating CNN with sparse feature maps, making them suitable for inference on DVS histogram output [2].

REMOT follows the line of work of operating directly on the event output from a DVS by introducing a customized HW/SW architecture to perform multi-object tracking. However, we further expand its flexibility by allowing it to be reconfigurable and programmable through dedicated codesign strategies.

2.3 Multi-object Tracking Using DVS

Multi-object tracking [32] is a challenging computer vision task that tracks multiple objects in a dynamic scene. Apart from detecting an object in a scene, it also requires an algorithm to assign a unique index to each independent object and track its trajectory. There has been a wide variety of MOT algorithms proposed for frame-based camera, with "tracking by detection" being the mainstream [3, 4, 6]. It is mainly composed of two steps: (i) apply a detector to detect objects in each frame; (ii) perform association on the detected objects across frames.

Following a similar approach, a number of event-based object tracking algorithms have been proposed. For instance, in [21], the authors demonstrated effective tracking using a correlation filter on top of a CNN structure. In the work of EBBIOT [1], a vehicle tracking system based on event sensors was demonstrated. Using an adaptive time surface formulation of events, Chen et al. [8] have demonstrated multiple object tracking in a controlled environment. Leveraging recent advent in machine learning, an offline-online learning approach was proposed in [19] to perform event-based object tracking with comparable performance to frame-based algorithms. In a recent work of EKLT [16], the use of simultaneous event and frame-based input to perform feature tracking was proposed.

While the above works have achieved good object tracking performance, their complex designs were not optimized for real-time

Table 1: Previous Hardware Deployments of Event-based Vision Tasks using Dynamic Vision Sensors.

	Hardware Platform	Task	Main Algorithm	Per-Event Processing	Performance	System Power	Dynamic/Chip Power
[34]	Loihi[11]	Object Tracking	SNN	✓	N/A	N/A	N/A
[38]	FPGA+TrueNorth[27]	Object Tracking, Classification	Event-based Tracker + SNN	✗	15fps	N/A	0.55W
[35]	Neuromorphic Chips	Object Recognition, Tracking	SNN	✓	3Meps	N/A	0.4W
[22, 23]	FPGA	Object Tracking	Center of Mass Calculation	✓	120-140ns	10W	N/A
[37]	FPGA	Gesture Recognition	Hierarchiy Of Time Surface	✓	0.16-2Meps	1.6W	0.077W
[30]	FPGA	Pedestrian Detection	BNN	✗	130fps	N/A	N/A
[33]	FPGA	Object Detection	PCA, kd-tree, SVM	✓	550ns	3W	0.37W
[24]	FPGA	Object Classification	CNN	✗	160fps	1W	0.27W
[31]	GPU	Object Detection	CNN	✗	25fps	N/A	N/A
REMOT	FPGA	Object Tracking	Attention-Guided MOT	✓	0.43-2.2Meps	1.75-5.68W	N/A

implementations. For that, the authors of E-MS [5] demonstrated an effective real-time event-based multi-object tracking by performing mean-shift clustering on the incoming events as they were produced. Subsequently, in [23], the authors demonstrated a low-latency event-based object tracker by performing inline center-of-mass computation using FPGA.

Our proposed REMOT framework similarly performs multi-object tracking directly on the dynamic vision events as they are produced. Taking advantage of the high temporal resolution and spatial sparsity of DVS, REMOT identifies and tracks multiple objects simultaneously using a layer of AUs. Each AU is an independent tracker that only pays attention to a small region that is updated in a per-event manner (Section 3). As an object moves, the attention region follows its motion based on the corresponding events that are produced, thereby tracking the moving object. During the lifetime of an AU, it will be assigned a unique global index as the tracking ID. Therefore, no object association in the traditional MOT sense is needed. Instead, supervisory functions are needed to ensure that each AU is indeed tracking useful objects. These high-level decisions are made based on the status of the AUs and may operate in millisecond scale comparable to a frame speed. In this way, a hierarchy of vision system is established with different processing rates on different levels.

3 REMOT ARCHITECTURE & ALGORITHMS

REMOT defines a hardware-software architecture and its associated operations that allow real-time event-based multi-object tracking algorithms to be developed (Fig. 4). The design of REMOT is based on the notion of an attention unit (AU). An AU is an autonomous entity that observes events from a DVS as they are produced. An AU maintains a region of attention (ROA) that defines the area in the imaging field where this AU is currently paying attention to. An implementation of REMOT will typically include a large set of independent AUs, which collectively attend to different parts of the imaging field where there are interesting events. Furthermore, a high-level controller oversees the operations of the entire set of AUs, and makes group decisions based on the state of each individual AU. It is by carefully manipulating the actions of the AUs that a family of attention-guided multi-object algorithms can be defined.

As shown in Fig. 4, the set of AUs can be implemented naturally as independent units running in hardware so they can process every event at line rate as they are produced. On the other hand, the high-level controller can naturally be implemented as software running on a microprocessor. While we have employed this natural mapping in this work, future implementations may choose to implement part of the controller logic in hardware to improve system performance.

In its most basic form, each AU supports only a small set of predefined actions:

- *Expand* — An AU may choose to expand its ROA when it observes an event that it is *interested* in. In that case, the AU captures the event and adjusts its ROA accordingly. Part of the algorithm designer's job is to define the criteria under which an AU is interested in an event (top left, Fig. 4).
- *Shrink* — An AU may choose to shrink its ROA as events age and no longer require attention. In that case, the AU may *forget* the event according to criteria set up by the algorithm (bottom left, Fig. 4).
- *Merge* — An algorithm may choose to merge two AUs during run time as more information about each AU is aggregated. When 2 AU merges, a new AU is formed with a combined ROA that is the union of the two original ROAs. Since merge requires information about more than one AU, the decision to merge is initiated by the controller that oversees all AUs (top right, Fig. 4).
- *Split* — An algorithm may decide that an AU should split into multiple AUs depending on algorithm-specific criteria (Fig. 4). When an AU splits, two new AUs are formed with each of them inheriting a subset of the original region of attention (bottom right, Fig. 4).

In addition, REMOT also defines the *spawn* and *delete* operations for the AU controller. An AU is spawned when no existing AU is interested in a new coming event. The ROA of the new AU will center around the new event. As for *delete*, if the ROA of an AU has shrunk to a size less than a threshold or an AU being idle for a period of time, the controller may decide to delete the AU. In software implementations, an unlimited number of AU can be spawned. However, in hardware implementations, the number of physical AU is fixed and limits the maximum number of AU that can be spawned. Consequently, it is possible that events might be dropped due to lack of AU and affect the overall MOT accuracy.

3.1 Multi-object Tracking with REMOT

As mentioned in Section 2.3, the task of multi-object tracking (MOT) in REMOT can now be recast into the problem of deriving algorithms to guide the AUs into correctly paying attention to and thereby tracking objects of interest. This can be accomplished by designing appropriate algorithms for each of the AU actions above.

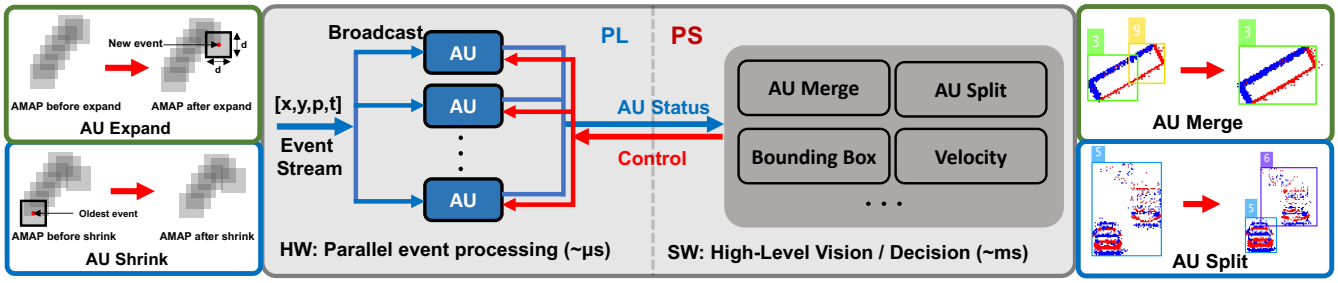


Figure 4: HW/SW Architecture.

3.1.1 Expand/Shrink Algorithms. Our current implementation of expand and shrink actions are centered around the use of an attention map (AMAP) in each AU that records its current ROA. Each AU also contains an Active Event FIFO to record its recently interested events. When a new event arrives within the ROA of an AU (i.e., $\text{AMAP}[x, y] > 0$), the $d \times d$ area of the AMAP centered at $[x, y]$ is incremented by 1 and the event is pushed into the FIFO. When an event ages and is popped out from the FIFO, its corresponding $d \times d$ area will be decremented by 1, shrinking the AMAP accordingly.

3.1.2 Merging Algorithms. We propose two merging algorithms, the distance-based algorithm and the ratio-based algorithm. When two AUs are merged, the tracking events from both AUs are joined and the attention regions are superposed. The bounding box of the merged AU is calculated based on the blended events set and the tracking ID of the merged AU is the ID of the earliest spawned AU.

Distance-based merging algorithm uses the Hausdorff distance to determine whether two neighboring AUs should be merged into a single AU. If the Hausdorff distance between two sets of active events from two AUs is less than a threshold value, a *merge* operation will be carried out. The Hausdorff distance is effective to evaluate the distance between two sets of points, defined as:

$$\begin{aligned} H(A, B) &= \max(h(A, B), h(B, A)) \\ h(A, B) &= \max_{a \in A} \min_{b \in B} \|a - b\|_2 \end{aligned} \tag{1}$$

where $\|\cdot\|$ denotes the L2 norm and A and B are two set of points.

Ratio-based merging algorithm decides whether to merge two neighboring AUs based on the ratio of Interaction over Minimum (IoM). If the IoM ratio is larger than a threshold value, the neighboring AUs will be combined. The IoM ratio is defined as the overlapping area over the minimum area of two neighboring AUs.

3.1.3 Splitting Algorithms. The splitting operation leverages different cluster algorithms to determine whether the internal events develop into different separated groups. In REMOT, we employ two splitting algorithms to partition AUs: density-based splitting algorithm and hierarchy-based splitting algorithm. Each attention region of the separated AU is reconstructed using the new clustered events. A new tracking ID is assigned to the new spawned AUs.

Density-based splitting algorithm uses the density-based spatial clustering of applications with noise (DBSCAN) algorithm [14]. DBSCAN describes the spatial density of a location by the number of points in a given neighborhood radius. The points in high-density regions are clustered together. If the active events of one AU have at least two clusters, the AU will be split.

Hierarchy-based splitting algorithm utilizes the Hierarchical Agglomerative Clustering (HAC) algorithm [29], which builds a hierarchy of clusters to split events. HAC aggregates events from clusters, starting from one event per cluster. Two nearest clusters are merged into one at each iteration, until all the events are amalgamated into one cluster, forming a hierarchy of clusters. If the last two clusters are too far apart to merge, the events will be split.

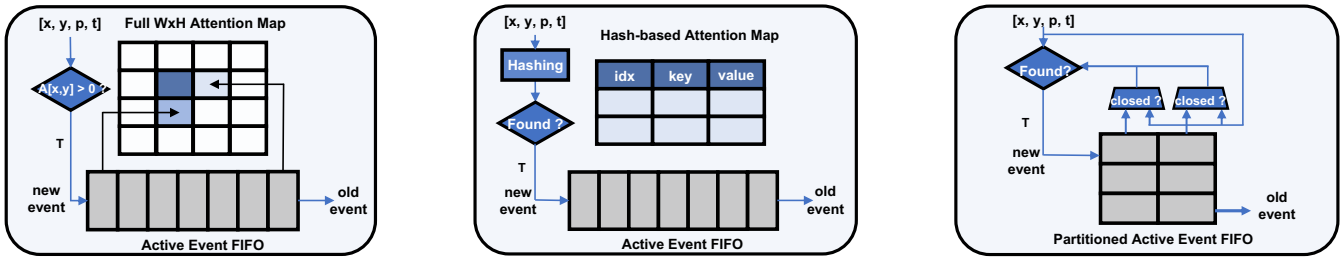
3.2 Hardware Designs

3.2.1 Overall Hardware Architecture. As shown in Fig. 4, the proposed hardware architecture of REMOT takes event stream as input. Each event will be broadcast to all hardware AUs to check whether the event lies within their attention region. The AU that considers the new event interested will push the event into its Active Events FIFO and update its attention map. If the Active Events FIFO is full, the oldest event will be popped out and the corresponding attention region will be shrunk. In addition, a global input event FIFO is used to buffer events in case that a burst of events arrive at the same time. The size of this FIFO is set to be larger than the peak event rates (in 1 millisecond interval) as the transmission latency of the DVS is usually within 1ms. The hardware also provides interfaces that allow the high-level software controller on PS to read and manipulate the status of AUs, e.g. the attention map and Active Event FIFO, to enable merge and split operations.

The proposed attention-guided tracking algorithm allows different hardware implementations to achieve different design tradeoffs between accuracy, throughput, resource, and power consumption. Here in REMOT, we introduce three implementations: Full-amap, Hash-amap, and Fifo-only.

3.2.2 Full-amap. The Full-amap is a straightforward implementation similar to the software implementation without additional hardware optimization. Each AU is composed of a $W \times H$ AMAP and an Active Event FIFO, where W and H is the width and height of the camera. The hardware expand and shrink behaviors follow the same procedures as the software implementation described in Section 3.1.1. Block RAM (BRAM) is used to implement AMAP on FPGA, resulting in around $d \times d$ cycles to update for every expand or shrink operation.

The Full-amap is considered as a costly hardware baseline without taking advantage of the inherent spatial sparsity of the DVS.



(a) Full-amap Implementation (b) Hash-amap Implementation (c) Fifo-only Implementation

Figure 5: Three different hardware implementations of an attention unit.

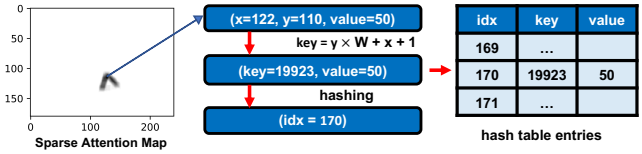


Figure 6: Hashing diagram of the attention map in Hash-amap implementation.

The BRAM usage for an AU in Full-amap can be estimated as:

$$B_{\text{Full-amap}} = \lceil (W \times H \times 16 + D_{FIFO} \times 64)/16\text{Kb} \rceil \quad (2)$$

where D_{FIFO} is the depth of the 64 bits Active Event FIFO and attention map uses 16 bits precision. In addition, the throughput of the Full-amap implementation is mainly determined by expand and shrink size d, which can be estimated using:

$$T_{\text{Full-amap}} = freq/(2(d^2 + C)) \quad (3)$$

where $freq$ is the clock frequency of the Programmable Logic (PL) and C is a constant overhead including the pipeline latency.

3.2.3 Hash-amap. As mentioned above, the inherent spatial sparsity in DVS brings great opportunities for hardware optimization. Thus, hash table becomes a good candidate to implement a sparse attention map in AU. By only storing non-zero attention region in a hash table, large on-chip memory is saved for each AU. In this way, more AUs can be deployed under the same resource constraints.

Fig 6 shows the diagram of the hashing scheme. For each location $[x, y]$, the hash key can be generated using:

$$key = y \times W + x + 1 \quad (4)$$

where W is the width of the camera screen. This key generation function calculates the flattened 1D array index of $[x, y]$ in the original 2D $W \times H$ space. It ensures that a unique hash key can be assigned to each location in the $W \times H$ space. Other mapping functions that can guarantee this uniqueness is also acceptable. In addition, we choose the binary multiplicative hashing function [13] to obtain the index of the hash table entry to store a key-value pair. For a hash table with 2^l entries and w bits hash key, the hashing function can be described as:

$$idx = (a \times key)[w-1 : w-l] \quad (5)$$

where a is a w bits constant number and $[w-1 : w-l]$ refers to $w-l$ to $w-1$ bits of the $2w$ bits multiplication result. In this way, we obtain an l bits index to store a key-value pair in the hash table.

For a sparse attention map in Fig 6, the hash table only stores the non-zero locations, which also corresponds to the ROA of the AU. When a new event comes, it will query the hash table using Eq.4 and Eq.5. If finding the matched key, the event will be considered interested and $d \times d$ locations will be updated similar to the Full-amap implementation.

Using hash table to store the sparse attention map also brings in the problem of hash collision, which means that two different non-zero locations might be mapped into the same hash table entry. To simplify the design and achieve a higher throughput, we use chaining as the collision handling strategy by allocating multiple slots for each hash table entry. If a collision happens, the new key-value pair goes to the next available slot. Otherwise, this key-value pair will simply be dropped.

The throughput of the Hash-amap implementation is similar to the Full-amap in Eq.3. However, the BRAM consumption can be largely reduced by using hash table, which can be written as:

$$B_{\text{Hash-amap}} = \lceil ((w + 16) \times 2^l \times s + D_{FIFO} \times 64)/16\text{Kb} \rceil \quad (6)$$

where s is the number of slots in one hash table entry. Similarly, the precisions of the attention map and FIFO are 16 and 64 bits. The hash key precision w is set to 18 in our implementation.

3.2.4 Fifo-only. Both Full-amap and Hash-amap implementation keep a record of the attention map to determine whether the new coming event is interested. They achieve constant query time but spend around d^2 cycles to update the ROA. The Fifo-only implementation reformulates the attention-based algorithm in a different way and leads to a different design tradeoff.

The Fifo-only implementation compares the location of a new coming event with all the existing events in the Active Event FIFO. If its location lies within the expanding region of any active event, the new event will be considered interested and be pushed into the Active Event FIFO. The algorithm can be formulated as:

$$interested = \exists\, e \in F, s.t. |e_x - x| \le d \;\&\; |e_y - y| \le d \quad (7)$$

where F is the Active Event FIFO. Different from the attention-map-based methods, the complexity of query is O(n) and the complexity of updating FIFO is O(1) for Fifo-only, where n refers to the depth of the Active Event FIFO. Therefore, the throughput of Fifo-only is bounded by how fast it can traverse the entire FIFO. This generally

means that the conventional hardware implementation of a FIFO with one push/pop operation per-cycle is incapable for this design. For example, if the FIFO depth is 1024, it takes around 1024 cycles to determine whether a new event is interested or not. Assuming the PL fabric running at 100MHz, the throughput will be less than 0.1 Meps, which is slower than the real-time requirement. Thus, as depicted in Fig. 5, we can achieve parallel access of the FIFO by partitioning. Given a target throughput T, we can estimate the partition factor of the FIFO and the overall BRAM consumption using:

$$\begin{aligned} P &= \lceil D_{FIFO}/(T/freq) \rceil \\ B_{\text{Fifo-only}} &= P \times \lceil 64 \times D_{FIFO}/(P \times 16\text{Kb}) \rceil \end{aligned} \tag{8}$$

which leads to a different throughput-resources tradeoff compared to the attention-map-based implementations.

4 RESULTS

4.1 Datasets and Evaluation Metrics

We evaluated the proposed algorithms on three event camera datasets. The first one is a 10 s segment from the open-source data *shapes_6dof* [28], and the other two datasets, *inbound traffic*, and *outbound traffic*, are captured by ourselves that show inbound and outbound traffic respectively. The data were captured using a DAVIS 346 camera [7] that produced simultaneous events and image frames. Table 3 shows the average and peak event rates for the datasets. The ground truth bounding boxes for object tracking were manually labeled using the frame-based images.

To evaluate tracking performance, we use HOTA (Higher Order Tracking Accuracy) [26], which is the default metric for multi-object tracking in many frame-based object tracking benchmarks including MOTChallenge MOT20 [12] and KITTI MOTS [17]. HOTA is a unified metric that evaluates both detection and association accuracy of an algorithm as follows.

Detection accuracy measures the alignment between the predicted bounding boxes and the ground-truth bounding boxes:

$$\text{DetA} = \int_{0<\alpha\leq 1} \text{DetA}_\alpha = \int_{0<\alpha\leq 1} \frac{|\text{TP}_\alpha|}{|\text{TP}_\alpha| + |\text{FP}_\alpha| + |\text{FN}_\alpha|} \tag{9}$$

where $|\text{TP}_\alpha|$, $|\text{FP}_\alpha|$ and $|\text{FN}_\alpha|$ refer to the numbers of true positives, false positives, and false negatives, with Intersection over Union (IoU) between predicted and ground-truth bounding boxes larger than the minimum match threshold α.

Association accuracy measures alignment between predicted track and the ground-truth track:

$$\text{AssA} = \int_{0<\alpha\leq 1} \text{AssA}_\alpha = \frac{1}{|\text{TP}_\alpha|} \sum_{c\in\text{TP}} \frac{|\text{TPA}^c_\alpha|}{|\text{TPA}^c_\alpha| + |\text{FPA}^c_\alpha| + |\text{FNA}^c_\alpha|} \tag{10}$$

where c is a given true positive, α is minimum IoU, and $|\text{TPA}^c_\alpha|$, $|\text{FNA}^c_\alpha|$, $|\text{FPA}^c_\alpha|$ correspond to the size of true positive association, false negative association and false positive association.

HOTA unites detection accuracy and association accuracy:

$$\text{HOTA} = \int_{0<\alpha\leq 1} \text{HOTA}_\alpha = \int_{0<\alpha\leq 1} \sqrt{\text{DetA}_\alpha \cdot \text{AssA}_\alpha} \tag{11}$$

4.2 Hardware Implementation Results

4.2.1 Hardware Experiment setting. In this section, we present the results on the hardware implementation of REMOT. A series of HW/SW experiments were carried out to demonstrate the multiple design tradeoffs in accuracy, throughput, resources and power. To demonstrate the flexibility and scalability of REMOT, we implemented multiple REMOT configurations on two embedded FPGA platforms: PYNQ-Z2 (Zynq 7Z020) and Ultra96 (Zynq ZU3EG). In addition, a software-only baseline of the proposed algorithm was implemented on the processor of Ultra96 (Arm CORTEX-A53) for performance comparison.

4.2.2 Performance comparison between FPGA and CPU. Fig. 7b shows a general picture of the different performance models of CPU and FPGA for low-level event processing (expand and shrink). Both FPGA and CPU results were measured on Ultra96. The Fifo-only implementation was used for FPGA results and the throughput was measured on the development board after synthesis, place and route with different number of AUs deployed. According to Fig. 7b, the parallel hardware AUs on FPGA achieve a comparable performance across different AU numbers while the processing throughput of CPU decreases dramatically as the number of AU increases. Even though the throughput of FPGA does drop slightly owing to lower PL clock frequencies as the resource utilization increases for more AUs, the speed-up continues to grow and achieves up to 34×. The results demonstrate our hardware architecture in REMOT can lead to a scalable performance in the low-level parallel event processing. In terms of power, the CPU implementation has a relatively static power consumption of 4.64W while the FPGA implementation with 13 AUs runs at 5.68W, resulting in 25.9× improvement in terms of power efficiency (Meps / W).

At the same time, more AUs means more capabilities to potentially track objects at the same time. Fig. 7a shows the tracking results on different datasets versus the maximum AU allowed. It points out the fact that if the CPU-only implementation wants to achieve higher tracking accuracy by using more AUs, it cannot meet the real-time throughput requirement (> 0.3Meps) for event processing.

Generally, for a given dataset, the accuracy will saturate at some points depending on how many objects would appear at the same time. In our case, 10 AUs would be sufficient for all three datasets. However, this result might not genuinely reflect the situation for other scenarios, e.g. a heavier traffic scene. The purpose of Fig. 7a is to illustate the important accuracy-resource tradeoff affected by the maximum number of AUs allowed. If more AU can be deployed under given resource constraints, a higher multi-object tracking accuracy can be expected to some extent.

4.2.3 Tradeoffs in different Hardware Implementations. In this section, we present some detailed discussions on implementation results with different design tradeoffs.

Tradeoffs in throughput. For the attention-map-based implementations (Full-amap and Hash-amap), throughput is bounded by sequentially updating attention region. Fig. 7d demonstrates this accuracy-throughput tradeoff with different expand and shrink size d shown in Fig. 4. The results were obtained based on the *shapes_6dof* dataset, while the throughput is calculated using Eq. 3 assuming a 300MHz clock frequency. As shown in Fig. 7d, the optimal value of d is 11 for the *shapes_6dof* dataset, leading to 1.04 Meps throughput.

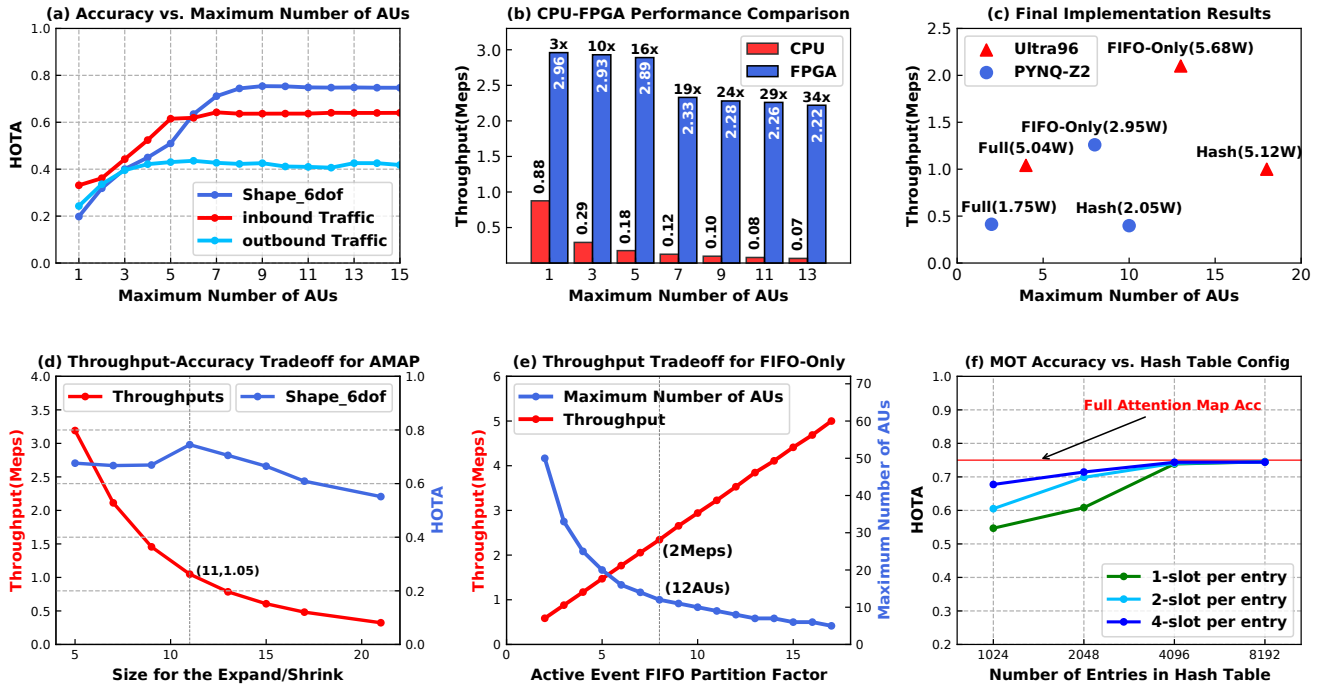


Figure 7: Hardware Implementation. (a) shows MOT accuracy with a different maximum number of AUs on three datasets; (b) compares performances between CPU and FPGA; (c) presents final performances for the three implementations on Ultra96 and PYNQ-Z2; (d) captures throughput-accuracy tradeoff for HASH-AMAP and FULL-AMAP implementations; (e) shows throughput-resource tradeoff for FIFO-ONLY; (f) illustrates accuracy degradation for different hash-table configurations.

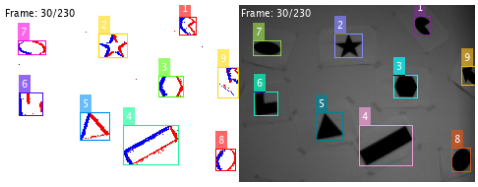
The FIFO-ONLY implementation has a different throughput tradeoff compared with attention-map-based design, which is determined by how fast it can traverse through the entire FIFO as described in Eq. 8. The throughput grows linearly with the FIFO partition factor while the BRAM usage for the FIFO also increases. Fig. 7e shows the theoretical throughput and the maximum number of AUs that can be deployed on Ultra96 with respect to different FIFO partition factors. The depth of the Active Event FIFO is set to 1024 in the experiments. As marked by the dashed line in Fig. 7e, if the partition factor is 8, we can embed around 12 AUs with 2 Meps throughput, closed to the final implementation result shown in Fig. 7c.

Tradeoff in HASH-AMAP design. Another accuracy-resource tradeoff exists in HASH-AMAP implementation. As discussed in Section 3, the HASH-AMAP leverages the intrinsic sparsity in attention-map to save on-chip memory consumption. However, it also brings in a new problem of hash collision that can potentially flaw the attention map. Fig. 7f shows the accuracy degradation for different hash table configurations on *shapes_6dof*. The accuracy would hardly drop when the number of entries exceeds 4096 compared to a full attention map. The results also show that using both more slots and hash table entries can benefit the accuracy. However, since the total hash table size is the number of entries times the number of slots per entry, increasing the entries number seems to bring more marginal benefits as shown in Fig. 7f. However, this is an empirical conclusion that only reflects the overall effects of our hashing function, data accessing pattern, and collision handling strategy. In the final implementations with a conservative configuration (8192×1), the HASH-AMAP enables around 4× more AUs to be deployed compared to the FULL-AMAP as shown in Fig. 7c. In this way, more potential objects can be tracked with more hardware AUs deployed.

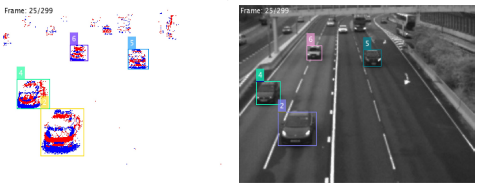
4.2.4 Scalability and Power Consumption. In the final deployment, we devise three different implementations on both PYNQ-Z2 and Ultra96 and measure their throughput and power consumption. The power was measured using a power source connected to the development board, which reflects the total power consumption for the system. Specifically, we set the voltage of the power source and observed the stable current value of the power source in a continuous input test. The internal AU configurations are identical for Ultra96 and PYNQ-Z2, leaving the available hardware resources to determine the maximum number of AUs. The final hardware AU quantities as well as the corresponding throughput and power consumption are shown in Fig. 7c. Table 2 lists the detailed resource utilization of different implementations. In general, our design shows high scalability. The Ultra96 development board has 54% more on-chip BRAM compared to PYNQ-Z2, resulting in around 60% increase in the maximum number of AUs deployed. The throughput on Ultra96 is also higher than the corresponding version on PYNQ-Z2 for a higher clock frequency after place and route.

Table 2: Resource utilization when maximum number of hardware AUs are instantiated.

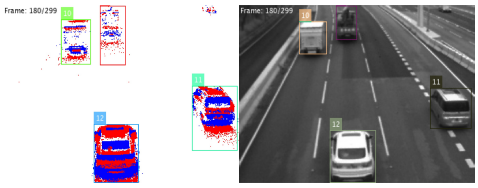
	DSP	LUTs	BRAM	FF	Freq(MHz)
PYNQ-Z2	220	53200	280	106400	
Full-amap	1%	20%	77%	14%	125
Hash-amap	5%	28%	80%	20%	125
Fifo-only	0%	20%	96%	13%	125
Ultra96	360	70560	432	141120	
Full-amap	1%	14%	96%	9%	300
Hash-amap	5%	26%	87%	5%	300
Fifo-only	0%	22%	99%	14%	250



(a) *shapes_6dof*



(b) *inbound traffic*



(c) *outbound traffic*

Figure 8: Visualization of tracking results. The left-hand side of each figure shows the events accumulated from the 40 ms prior to the corresponding image frame and tracking results from REMOT. The right-hand side shows the corresponding image frame with the ground truth bounding boxes.

4.3 MOT Performance

4.3.1 Multi-Object Tracking Accuracy. Table 3 summarizes the tracking accuracy of REMOT as measured by the HOTA metrics while Fig. 8 shows the visualization of tracking results. Specifically, the results were produced under the distance-based merging algorithm and DBSCAN splitting algorithm. Overall, REMOT performed best with the relatively simple *shapes_6dof* benchmark followed by the more complex real-world benchmark of *inbound traffic* and *outbound traffic*. Real-world challenges such as the presence of shadows, which the AUs regard as part of a vehicle but the human-produced ground truth labels did not, caused mismatched bounding box calculations. Furthermore, vehicles movements, such as when individual cars begin to merge in outbound traffic or when cars emerged from afar in inbound traffic, challenges our current simplistic AU merge, split, and expand actions.

Table 3: Tracking results and data rates on different datasets

	shapes_6dof	inbound traffic	outbound traffic
Detection Accuracy (%)	70.4	50.9	39.0
Association Accuracy (%)	76.3	58.0	47.8
HOTA (%)	73.2	54.3	43.1
Average Event Rate (Meps)	0.30	0.26	0.22
Peak Event Rate (Meps)	2.16	1.03	0.84

Table 4: Comparison results with other methods on the *shapes_6dof* dataset.

	Metrics	AP (%)	AR (%)	HOTA (%)
[5]	E-MS	61.2	66.8	40.2
[9]	ETD	80.9	99.8	N.A.
[8]	RMRNet	86.6	98.0	N.A.
	REMOT	76.5	94.3	73.2

We further compared the performance of REMOT against 3 related works that addressed similar event-based MOT challenges and have reported results using *shapes_6dof*, as shown in Table 4. Without access to the source code of ETD [9] and RMRNet [8], we instead evaluated REMOT using the specialized metrics employed in these 2 papers, namely, AP (average precision, also known as average overlap rate in [8]) and AR (average robustness) with our segment from *shapes_6dof*. The HOTA value for E-MS [5] was produced by evaluating their released source code using our segment from *shapes_6dof*. Minimum enclosing bounding boxes were created based on the segmented clusters of events per frame and were labeled with the corresponding segment color for association. The AR and AP values for E-MS were reproduced from [8]. Results show that REMOT performs better than E-MS across all 3 metrics, but is shy of achieving the same performance as ETD and RMRNet in the AP and AR metrics. Nevertheless, REMOT was able to achieve such accuracy in real time while consuming only a fraction of power using low-end FPGA-based HW/SW systems.

4.3.2 Flexibility of the high-level merge/split algorithms. As described in Section 3, REMOT decouples the low-level event processing from the high-level vision decision in a hierarchical way. The high-level merge/split algorithms implemented on the processing system can be extremely flexible and modular. For example, Fig. 9 shows the decomposed accuracies of *shapes_6dof* dataset under two different split algorithms. The solid lines are the accuracies using the DBSCAN splitting algorithm, and the dash lines correspond to the HAC splitting algorithm, both combined with the distance-based merging algorithm. According to Fig. 9, the software under different configurations can exhibit different performances in detection and association.

Fig. 10 also demonstrates the performance of the two merging algorithms, distance-based and ratio-based, assuming the same DBSCAN splitting algorithm. The color of Fig. 10 indicates the HOTA value. The y-axes of the figures are the expanding and shrinking size d. The x-axis of Fig. 10a is the maximum distance to merge

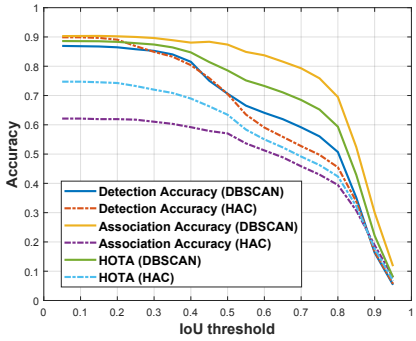


Figure 9: Decomposed accuracies using two splitting algorithms of DBSCAN and HAC, with the changes of minimum IoU between tracked and ground-truth bounding boxes.

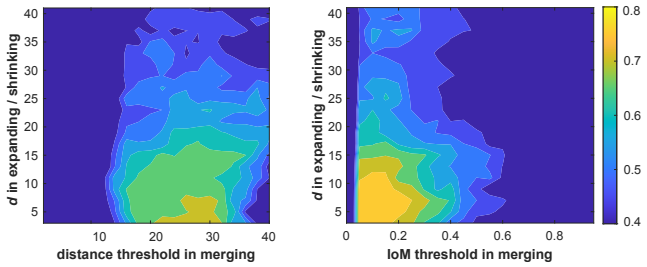


Figure 10: Accuracy varies as the changes of expanding/shrinking parameter (vertical) and merging parameter (horizontal). Both y-axes are the size d of expanding and shrinking. The x-axis of (a) is the maximum distance of merging, and x-axis of (b) is the minimum IoM ratio of merging.

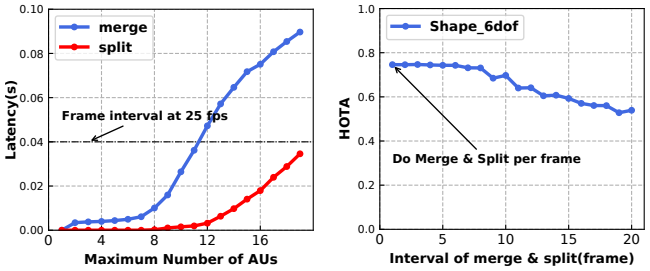


Figure 11: Latencies of merging and splitting get longer as the maximum number of AUs increases (a). Accuracy drops slightly as the time interval of merging and splitting operations is prolonged (b).

neighboring AUs, and the x-axis of the Fig. 10b is the minimum IoM ratio to merge AUs. Since the size d is a hardened hardware parameter, Fig. 10 brings in additional variability on the software side to explore a better overall tracking performance.

In addition, the decoupled high-level vision algorithms implemented on the PS can process at a lower rate than conventional frame-based vision algorithms. Normally, frame-based tracking algorithm would process each frame one after another. The real-time requirement for these algorithms is that the processing frame rate should be higher than the frame rate of the camera, e.g. 25 fps. However, in our REMOT, since the low-level hardware AU is continuously tracking the dynamic scene in real time, the high-level merge/split software does not necessarily need to operate at a fixed frame rate similar to a frame-based camera. Fig. 11a shows the average latencies of merge and split operations, measured on Ultra96 using the *shapes_6dof* dataset. As the maximum number of AUs grows, it takes longer to perform a merge/split operation for all AUs. If using 25 fps as requirement, the maximum number of AUs cannot exceed 11. However, Fig. 11b shows that the tracking accuracy would hardly decrease even when the merge/split happens every 6 frames (240 ms). This brings much room for tolerance to a low-end processor on edge platforms. It again demonstrates the virtue of the proposed HW/SW hierarchy in REMOT.

5 LIMITATIONS & FUTURE WORK

While our current design of REMOT works well with our target dataset, it has several limitations we plan to address in the future. First, the current MOT algorithm is designed for use with a stationary DVS camera where the static background is inherently removed. In cases with a moving camera, our MOT algorithm will need to be enhanced with techniques distinguishing between target objects and the background. Also, our MOT algorithms currently cannot handle occlusion well when the tracks of two objects cross. Finally, our proposed algorithms include many heuristic parameters that are sensitive to specific scenarios. In the future, we intend to develop learning-based algorithms that take advantages of the partial information collected by each AU to guide their merge and split operations so they can be aware of occlusions and adapt to changing scenarios. We intend to accelerate these data-driven operations from software to a new hardware AU manager close to the AUs.

6 CONCLUSIONS

In this paper, we have presented REMOT, a hardware-software architecture and a family of attention-guided multi-object tracking algorithms that run on this system. By partitioning the MOT task to operate in both hardware and software, we demonstrated that real-time performance can be achieved even on modest edge FPGA platforms when tested on real-world DVS datasets. The partitioned architecture allowed efficient low-level event processing throughput at up to 2.22 Meps while consuming 5.68W system power, which was 33.6 times higher throughput and 25.9 times more power-efficient than an equivalent software-only implementation. The proposed architecture is flexible, modular, and scalable, which allows future improved attention-guided MOT algorithms to be developed.

ACKNOWLEDGMENTS

This work was supported in part by the Croucher Innovation Award, Croucher Foundation, and ACCESS – AI Chip Center for Emerging Smart Systems, sponsored by Innovation and Technology Fund (ITF), Hong Kong SAR.

REFERENCES

[1] Jyotibdha Acharya, Andres Ussa Caycedo, Vandana Reddy Padala, Rishi Raj Singh Sidhu, Garrick Orchard, Bharath Ramesh, and Arindam Basu. 2019. EBBIOT: A Low-complexity Tracking Algorithm for Surveillance in IoVT using Stationary Neuromorphic Vision Sensors. In *2019 32nd IEEE International System-on-Chip Conference (SOCC)*. 318–323.

[2] Alessandro Aimar, Hesham Mostafa, Enrico Calabrese, Antonio Rios-Navarro, Ricardo Tapiador-Morales, Iulia-Alexandra Lungu, Moritz B. Milde, Federico Corradi, Alejandro Linares-Barranco, Shih-Chii Liu, and Tobi Delbruck. 2019. NullHop: A Flexible Convolutional Neural Network Accelerator Based on Sparse Representations of Feature Maps. *IEEE Transactions on Neural Networks and Learning Systems* 30, 3 (2019), 644–656.

[3] Boris Babenko, Ming-Hsuan Yang, and Serge Belongie. 2009. Visual Tracking with Online Multiple Instance Learning. *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 983–990.

[4] Boris Babenko, Ming-Hsuan Yang, and Serge Belongie. 2010. Robust Object Tracking with Online Multiple Instance Learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 33, 8 (2010), 1619–1632.

[5] Francisco Barranco, Cornelia Fermuller, and Eduardo Ros. 2018. Real-time clustering and multi-target tracking using event-based sensors. *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 5764–5769.

[6] Erik Bochinski, Volker Eiselein, and Thomas Sikora. 2017. High-speed tracking-by-detection without using image information. *2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, 1–6.

[7] Christian Brandli, Raphael Berner, Minhao Yang, Shih-Chii Liu, and Tobi Delbruck. 2014. A 240× 180 130 db 3 μs latency global shutter spatiotemporal vision sensor. *IEEE Journal of Solid-State Circuits* 49, 10 (2014), 2333–2341.

[8] Haosheng Chen, David Suter, Qiangqiang Wu, and Hanzi Wang. 2020. End-to-end learning of object motion estimation from retinal events for event-based object tracking. *Proceedings of the AAAI Conference on Artificial Intelligence* 34, 07, 10534–10541.

[9] Haosheng Chen, Qiangqiang Wu, Yanjie Liang, Xinbo Gao, and Hanzi Wang. 2019. Asynchronous Tracking-by-Detection on Adaptive Time Surfaces for Event-based Object Tracking. *Proceedings of the 27th ACM International Conference on Multimedia*, 473–481.

[10] Gregory K Cohen, Garrick Orchard, Sio-Hoi Leng, Jonathan Tapson, Ryad B Benosman, and André Van Schaik. 2016. Skimming digits: neuromorphic classification of spike-encoded images. *Frontiers in neuroscience* 10 (2016), 184.

[11] Mike Davies, Narayan Srinivasa, Tsung-Han Lin, Gautham Chinya, Yongqiang Cao, Sri Harsha Choday, Georgios Dimou, Prasad Joshi, Nabil Imam, Shweta Jain, et al. 2018. Loihi: A neuromorphic manycore processor with on-chip learning. *IEEE Micro* 38, 1 (2018), 82–99.

[12] P. Dendorfer, H. Rezatofighi, A. Milan, J. Shi, D. Cremers, I. Reid, S. Roth, K. Schindler, and L. Leal-Taixé. 2020. MOT20: A benchmark for multi object tracking in crowded scenes. *arXiv:2003.09003[cs]* (March 2020). http://arxiv.org/abs/1906.04567 arXiv: 2003.09003.

[13] Martin Dietzfelbinger, Torben Hagerup, Jyrki Katajainen, and Martti Penttonen. 1997. A reliable randomized algorithm for the closest-pair problem. *Journal of Algorithms* 25, 1 (1997), 19–51.

[14] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. *kdd* 96, 34, 226–231.

[15] Guillermo Gallego, Tobi Delbruck, Garrick Michael Orchard, Chiara Bartolozzi, Brian Taba, Andrea Censi, Stefan Leutenegger, Andrew Davison, Jorg Conradt, Kostas Daniilidis, and Davide Scaramuzza. 2020. Event-based Vision: A Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2020), 1–26.

[16] Daniel Gehrig, Henri Rebecq, Guillermo Gallego, and Davide Scaramuzza. 2020. EKLT: Asynchronous Photometric Feature Tracking Using Events and Frames. *International Journal of Computer Vision* 128, 3 (2020), 601–618.

[17] Andreas Geiger, Philip Lenz, and Raquel Urtasun. 2012. Are we ready for autonomous driving? the kitti vision benchmark suite. *2012 IEEE Conference on Computer Vision and Pattern Recognition*, 3354–3361.

[18] Wulfram Gerstner and Werner M Kistler. 2002. *Spiking neuron models: Single neurons, populations, plasticity*. Cambridge university press.

[19] Rui Jiang, Xiaozheng Mou, Shunshun Shi, Yueyin Zhou, Qinyi Wang, Meng Dong, and Shoushun Chen. 2020. Object tracking on event cameras with offline–online learning. *CAAI Transactions on Intelligence Technology* 5, 3 (2020), 165–171.

[20] Xavier Lagorce, Garrick Orchard, Francesco Galluppi, Bertram E. Shi, and Ryad B. Benosman. 2017. HOTS: A Hierarchy of Event-Based Time-Surfaces for Pattern Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39, 7 (2017), 1346–1359.

[21] Hongmin Li and Luping Shi. 2019. Robust Event-Based Object Tracking Combining Correlation Filter and CNN Representation Representation. *Frontiers in Neurorobotics* 13 (2019), 82.

[22] A. Linares-Barranco, F. Gómez-Rodríguez, V. Villanueva, L. Longinotti, and T. Delbrück. 2015. A USB3.0 FPGA event-based filtering and tracking framework for dynamic vision sensors. *2015 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2417–2420.

[23] Alejandro Linares-Barranco, Fernando Perez-Peña, Diederik Paul Moeys, Francisco Gomez-Rodriguez, Gabriel Jimenez-Moreno, Shih-Chii Liu, and Tobi Delbruck. 2019. Low latency event-based filtering and feature extraction for dynamic vision sensors in real-time FPGA applications. *IEEE Access* 7 (2019), 134926–134942.

[24] Alejandro Linares-Barranco, Antonio Rios-Navarro, Salvador Canas-Moreno, Enrique Piñero-Fuentes, Ricardo Tapiador-Morales, and Tobi Delbruck. 2021. Dynamic vision sensor integration on FPGA-based CNN accelerators for high-speed visual classification. *International Conference on Neuromorphic Systems* (2021), 1–7.

[25] Qianhui Liu, Haibo Ruan, Dong Xing, Huajin Tang, and Gang Pan. 2020. Effective AER Object Classification Using Segmented Probability-Maximization Learning in Spiking Neural Networks. *Proceedings of the AAAI Conference on Artificial Intelligence* 34, 02 (2020), 1308–1315.

[26] Jonathon Luiten, Aljoša Ošep, Patrick Dendorfer, Philip Torr, Andreas Geiger, Laura Leal-Taixé, and Bastian Leibe. 2021. HOTA: A Higher Order Metric for Evaluating Multi-object Tracking. *International Journal of Computer Vision* 129, 2 (2021), 548–578.

[27] Paul A Merolla, John V Arthur, Rodrigo Alvarez-Icaza, Andrew S Cassidy, Jun Sawada, Filipp Akopyan, Bryan L Jackson, Nabil Imam, Chen Guo, Yutaka Nakamura, et al. 2014. A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science* 345, 6197 (2014), 668–673.

[28] Elias Mueggler, Henri Rebecq, Guillermo Gallego, Tobi Delbruck, and Davide Scaramuzza. 2017. The event-camera dataset and simulator: Event-based data for pose estimation, visual odometry, and SLAM. *The International Journal of Robotics Research* 36, 2 (2017), 142–149.

[29] Daniel Müllner. 2011. Modern hierarchical, agglomerative clustering algorithms. *arXiv preprint arXiv:1109.2378* (2011).

[30] Fernando Cladera Ojeda, Anthony Bisulco, Daniel Kepple, Volkan Isler, and Daniel D Lee. 2020. On-device event filtering with binary neural networks for pedestrian detection using neuromorphic vision sensors. *2020 IEEE International Conference on Image Processing (ICIP)*, 3084–3088.

[31] Etienne Perot, Pierre de Tournemire, Davide Nitti, Jonathan Masci, and Amos Sironi. 2020. Learning to Detect Objects with a 1 Megapixel Event Camera. *Advances in Neural Information Processing Systems* 33 (2020), 16639–16652.

[32] Zenon W Pylyshyn and Ron W Storm. 1988. Tracking multiple independent targets: Evidence for a parallel tracking mechanism. *Spatial vision* 3, 3 (1988), 179–197.

[33] Bharath Ramesh, Andrés Ussa, Luca Della Vedova, Hong Yang, and Garrick Orchard. 2018. PCA-RECT: An energy-efficient object detection approach for event cameras. *Asian Conference on Computer Vision*, 434–449.

[34] Alpha Renner, Matthew Evanusa, and Yulia Sandamirskaya. 2019. Event-Based Attention and Tracking on Neuromorphic Hardware. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 1709–1716.

[35] Rafael Serrano-Gotarredona, Matthias Oster, Patrick Lichtsteiner, Alejandro Linares-Barranco, Rafael Paz-Vicente, Francisco Gómez-Rodríguez, Luis Camuñas-Mesa, Raphael Berner, Manuel Rivas-Pérez, Tobi Delbruck, et al. 2009. CAVIAR: A 45k neuron, 5M synapse, 12G connects/s AER hardware sensory–processing–learning–actuating system for high-speed visual object recognition and tracking. *IEEE Transactions on Neural Networks* 20, 9 (2009), 1417–1438.

[36] Amos Sironi, Manuele Brambilla, Nicolas Bourdis, Xavier Lagorce, and Ryad Benosman. 2018. HATS: Histograms of Averaged Time Surfaces for Robust Event-Based Object Classification. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[37] Ricardo Tapiador-Morales, Jean-Matthieu Maro, Angel Jimenez-Fernandez, Gabriel Jimenez-Moreno, Ryad Benosman, and Alejandro Linares-Barranco. 2020. Event-Based Gesture Recognition through a Hierarchy of Time-Surfaces for FPGA. *Sensors* 20, 12 (2020), 3404.

[38] Andrés Ussa, Chockalingam Senthil Rajen, Deepak Singla, Jyotibdha Acharya, Gideon Fu Chuanrong, Arindam Basu, and Bharath Ramesh. 2020. A Hybrid Neuromorphic Object Tracking and Classification Framework for Real-time Systems. *arXiv preprint arXiv:2007.11404* (2020).