Transactions Briefs

Dynamic Neural Fields Accelerator Design for a Millimeter-Scale Tracking System

Yuyang Li, Vijay Shankaran Vivekanand, Rajkumar Kubendran, and Inhee Lee

***Abstract*— This brief introduces a compact-size hardware accelerator for dynamic neural fields (DNF) used in object tracking. To address the substantial computational workload and memory occupancy associated with conventional DNFs, three key approaches are implemented: kernel size reduction and abstraction, the replacement of sigmoidal functions with comparison operations, and the approximation of rectangular-shaped objects. The design is realized in a 28-nm CMOS process, resulting in a layout with an area of 0.53 mm^2. Simulation results demonstrate that the accelerator processes 256 × 256 dynamic vision sensor (DVS) frames at 211 frames per second (fps), with a power consumption of 1.68 mW under such conditions.**

***Index Terms*— Dynamic neural fields (DNF), dynamic vision sensor (DVS), object tracking.**

I. Introduction

The successful miniaturization of an active sensing system with an integrated battery down to the millimeter scale has opened up promising opportunities in various fields, including biomedical applications [1], ecology research [2], and surveillance [3]. Specifically, incorporating a miniature image sensor and cutting-edge machine learning techniques into this millimeter-scale system, which features a bare-die-stacked structure without individual packaging, paves the way for creating compact artificial intelligence of things (AIoT) vision systems [4]. However, challenges persist with the traditional frame-based image-sensing approach. These challenges, including high latency and excessive power consumption, impede progress in developing more advanced systems, such as enhancing the control of miniature robots [5] through millimeter-scale object tracking sensors.

To address these challenges, an alternative approach involves the use of event-based dynamic vision sensors (DVSs) [6], [7]. Typically developed for centimeter-scale embedded systems, these sensors provide pixel position and brightness change direction, only when there is a change in a pixel. This approach significantly reduces data size and latency, unlocking new potential for millimeter-scale vision systems. Yet, the unique output format of event-based image sensors necessitates a novel type of data processing unit to convert the output data into desired information, such as tracking trajectories. While feedforward artificial neural networks [8] can efficiently process event-based sensor output for fast-moving objects, they encounter difficulties in certain conditions, such as distinguishing similar objects from the target. In addition, event-based sensors generate reduced outputs when a target slows down or comes to a stop. As an alternative, a recurrent neural network known as bio-inspired dynamic neural fields (DNF) [9], a well-established model in computational neuroscience and cognitive science, has been explored for object tracking [10], [11]. However, a data processing unit suitable for a millimeter-scale system with a small footprint and low power consumption has not yet been explored.

Manuscript received 11 January 2024; revised 8 April 2024; accepted 30 April 2024. Date of publication 1 July 2024; date of current version 27 September 2024. This work was supported in part by the University of Pittsburgh Center for Research Computing through the resources provided. *(Corresponding author: Inhee Lee.)*

The authors are with the Department of Electrical and Computer Engineering, University of Pittsburgh, Pittsburgh, PA 15261 USA (e-mail: inhee.lee@pitt.edu).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TVLSI.2024.3416725.

Digital Object Identifier 10.1109/TVLSI.2024.3416725

Hence, this work proposes a DNF accelerator that can effectively interface with and process the output data from an event-based image sensor for a millimeter-scale motion-tracking vision system. By simplifying the kernel, the activation, and the object shape, the proposed accelerator reduces the computation workload and chip area. The designed layout shows an area of 0.53 mm^2, and the system consumes 1.68 mW at a frame rate of 211 frames per second (fps).

II. Conventional Approach

A. Conventional DNF Computing

Fig. 1(a) illustrates a typical application of DNFs, involving the tracking of selected vehicles in moving traffic. A tiny tracking device can be integrated into miniature robots (e.g., drones [12]) for better monitoring performance. The DVS camera distinguishes moving vehicles from the background scenery and masks the static region. DVS events are then used in the DNF as input to track the position of a selected target. The standard DNF process simulates processes observed in animal brains. Equation (1) delineates its evolving process [9]

$$\tau \cdot \dot{u}(x,t) = -u(x,t) + h + \int f(u(x',t))\omega(x-x')dx' + S(x,t) \quad (1)$$

where τ is a timing constant determining the response speed, and h represents the negative resting level. It comprises three 2-D matrices: the activation $u(x,t)$, the kernel $\omega(x-x')$, and the external DVS input $S(x,t)$. The activation reflects neurons and their focused areas, continuously updating. The kernel is a 2-D Gaussian function expressed mathematically as

$$\omega_{i,j} = C_{\text{exc}} \cdot e^{-\frac{d_{i,j}^2}{2\cdot\sigma_{\text{exc}}}} - C_{\text{inh}} \cdot e^{-\frac{d_{i,j}^2}{2\cdot\sigma_{\text{inh}}}} \quad (2)$$

where d is the distance between any position and the center; C_{exc} and C_{inh} represent the strength of excitatory and inhibitory connectivity, respectively; and σ_{exc} and σ_{inh} represent their ranges, respectively. $f(u(x',t))$ denotes the activation function, typically a sigmoid function [9].

To better align with the synchronized digital processing flow, the continuous updating operation is separated into a base of frames. The events with time stamps in a specific time range are binned into one 2-D DVS map using their coordinates and without polarity (all the events are treated as "+1"). The entire process can be divided into three main stages, which are shown in Fig. 1(b): convolution, normalization, and finding the center while updating the activation. During convolution, the current activation and kernel undergo a 2-D convolution to generate the interaction. The interaction encompasses both excitation and inhibition [9] toward input events for subsequent frames. The region near the target is highlighted (denoted by "+"), while the areas far from the target object are de-emphasized (denoted by "−"). The maximum interaction value is recorded in this stage. In the normalization phase, the entire convolution output is scaled to a range from −1 to +1, with the maximum noted in convolution. The subsequent stage updates the current activation using the interaction and the input DVS image. The discrete updating process of the activation can be expressed as

$$u_{n+1} = u_n + \frac{dt}{\tau}\left(-u_n + \sum_{i,j} f(u_{n,i,j}) \cdot \omega_{i,j} + S_{n+1}\right). \quad (3)$$

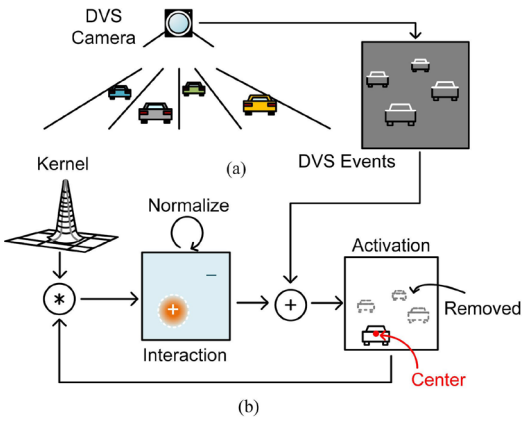

Fig. 1. Working mechanism of DVS camera and DNF tracking. (a) DVS camera used for traffic monitoring. (b) DNF tracking using DVS as input.

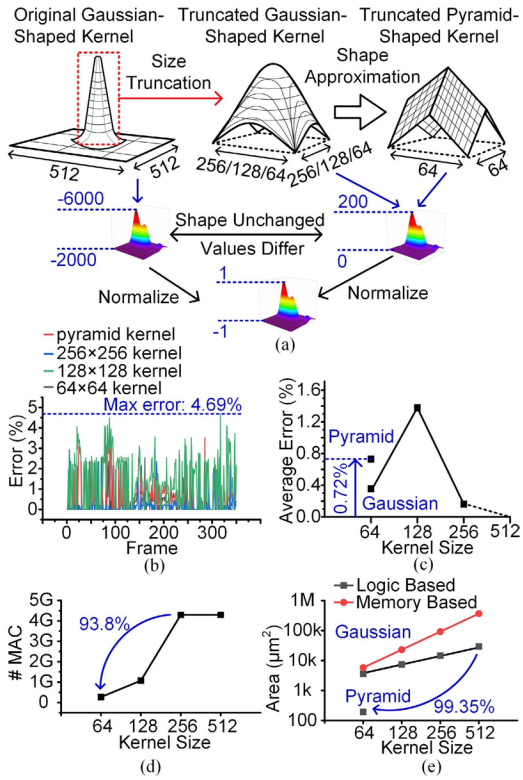

Fig. 2. Kernel size reduction and shape abstraction. (a) Shape change in the kernel. (b) Error of each frame compared with the original kernel with different kernels. (c) Average error with different kernels. (d) Number of MAC operations needed across kernel size. (e) Required circuit area across kernel size and shape, implemented with logic and memory.

B. Target Application and Challenges

To ensure efficient processing of the DNF model, three significant challenges need to be addressed. First, convolution poses a substantial computational and memory challenge. When working with a 256 ×256 activation matrix and a 512 ×512 kernel, the convolution operation demands 320 kB of memory space to store the input and output data. This memory requirement alone occupies 0.5 mm^2 in a 28-nm CMOS process. Second, performing the convolution entails over 4 billion multiply–accumulate (MAC) operations, necessitating high-throughput parallel computing. The third challenge arises from normalization. Normalizing the convolution output involves finding its maximum value. Thus, such a task can only be performed after the entire convolution concludes. The un-normalized interaction array requires being stored in memory, consuming extra time and area.

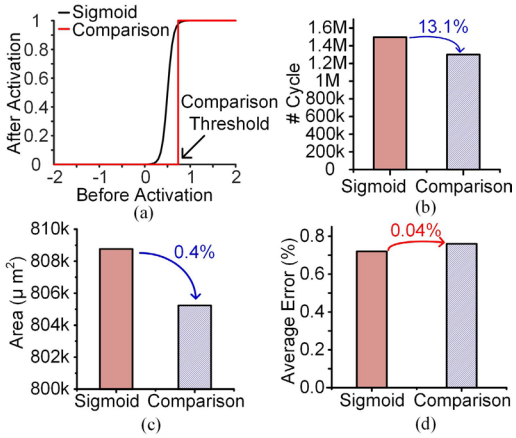

Fig. 3. Function and performance comparison of the sigmoid and comparator-based activation. (a) Curve of comparison and sigmoid function. (b) Processing time. (c) Area. (d) Average error of tracking.

III. Proposed DNF Accelerator Architecture

To address the challenges mentioned earlier, we simplify the DNF model from multiple perspectives without sacrificing notable tracking performance: kernel size, activation, and shape.

A. Kernel Size Reduction

Conventionally, the subtraction of two 2-D Gaussian functions in (2) makes the kernel flat near the edges and convex in the center [11]. To reduce the computation workload, the kernel size needs to be reduced. Truncating the kernel size from 512 to 64 leads to a reduction of 93.8% in the number of MAC operations. The left and middle sections of Fig. 2(a) indicate the reduction of the kernel size. Such truncation removes the flat part that contributes to the bias of the output but keeps the convex center so that the shape of the output is unchanged. It is worth noting that the value range of the convolution result changes as the kernel size is reduced, while the normalization adjusts it back to [−1, 1]. Thus, it is of higher priority to guarantee the shape is close to that of the original kernel rather than making the range close. Then, the normalization masks the difference in the value, as is shown in the lower part of Fig. 2(a). In addition, the kernel shape expressed in (2) represents a bell-shaped 2-D Gaussian function. Generating the values for the shape requires excess resources for both computation using logic and storing the kernel in memory. To eliminate the complex computation, we apply approximation to the kernel function by transforming the shape from a bell to a pyramid, as shown in the right part of Fig. 2(a). Fig. 2(b) and (c) depicts the per-frame and average errors with different kernel specifications. Here, we define the error as the distance between the object centers tracked by the original and proposed DNF computing methods. Our objective is to develop energy-efficient hardware for processing the original DNF, rather than further advancing the algorithm itself. The DNF is applied to tracking the objects from the "shapes_translation" DVS dataset [13]. Tracking the same object across 350 continuous frames that last 15 s, the tracking center shifts maximally by 4.7% and averagely by 0.72% for the pyramid-shaped, 64 × 64 kernel compared with the original kernel. It meets the requirement of a 10% maximum and 1% average error. The simplification of the kernel reduces the number of MAC by 93.8% [Fig. 2(d)]. In addition, the pyramid kernel saves 99.35% of area compared with the Gaussian-shaped kernel using the logic processing blocks.

B. Activation Simplification

A sigmoid function is commonly added at the end of the update to add nonlinearity to the activation [9] and filters the elements that

are not strong enough. However, it also requires specific arithmetic blocks, including exponential and division. To simplify the operation, we replace the sigmoid function with a comparison, as shown in Fig. 3(a). Fig. 3(b)–(d) shows comparison of its performance with the sigmoid function. The comparator-based activation saves 13.1% of processing time. Compared with an 8-bit look-up-table-based sigmoid function unit, the comparator cuts down the memory and saves 0.4% of circuit area while only generating 0.04% of the tracking shift for the pyramid-shaped kernel and sigmoid activation function.

C. Shape Abstraction

The center of mass and the centroid can represent the position of the target. The center of mass uses MAC operation and division that costs extra cycles while finding the centroid only requires finding the minimum bounding rectangle, which can be accomplished by comparison [14]. Fig. 4(a) shows the process that finds the centroid. The elements on the activation are compared with a threshold, and the positions on the farthest left, right, top, and bottom are used as the edges of the target object, and the edge coordinates are then averaged for the centroid position. This comparison can be merged with activation.

Under such a method, the centroid found for the target is equivalent to the centroid of the rectangle formed by its four borders. Since the position of the centroid is used for tracking, the influence of the object shape is minor. Hence, all the values within the rectangle formed by the borders are considered as "1"s. This abstraction removes the memory that stores the activation map. Since only the coordinates of the four boundaries are needed, for a 256 × 256 array size, 4 bytes of registers can hold the values.

With an abstracted object and the pyramid-shaped kernel, the convolution can be accomplished without the iterative MAC operation. Fig. 4(b) describes the new process of convolution. The kernel sweeps across the activation map, generating an overlapped region. Due to the object being a rectangle and all the values inside it being 1, the MAC is equivalent to the sum of the overlapped region of the kernel. Next, we consider the 2-D kernel as a collection of 1-D functions. Each 1-D function is a piecewise linear function (a piecewise arithmetic progression in the discrete domain). Its sum is derived by the equation given in Fig. 4(b). Then, the sum of multiple 1-D arrays can be converted into a second sum of arithmetic progressions, since each 1-D map has the same shape but a linearly increasing bias. Such computation only requires five multipliers and 13 adders and emulates a 64 × 64 MAC process in a single cycle. Compared with this, a conventional MAC unit needs 4096 multipliers and 4095 adders to achieve the same workload in one cycle, which is 455× larger.

For a rectangle object and a pyramid-shaped kernel, the maximum value is fixed at the position where their center overlaps. Consequently, the maximum value is known before the convolution, as the left part of Fig. 4(c) depicts. The normalization, as well as the comparison for finding the boundaries, is merged into the loop of the convolution. The memory to store the entire temporal convolution result is replaced by a register that holds the convolution results of the current position. Moreover, the interaction forms a 2-D parabolic shape only when the object and kernel overlap. Any input events outside such a region cannot trigger the activation to be "1." Thus, there is no need to calculate the convolution and normalization outside this region. With a rectangular shape, the area that the kernel and object overlap is also a rectangle, and its position can be calculated. The elements outside it are skipped, as is shown in the right part of Fig. 4(c).

Fig. 4(d) shows comparison of the performance of the shape abstraction with the original method. Merely using the sum of arithmetic progression to replace the MAC in the convolution brings

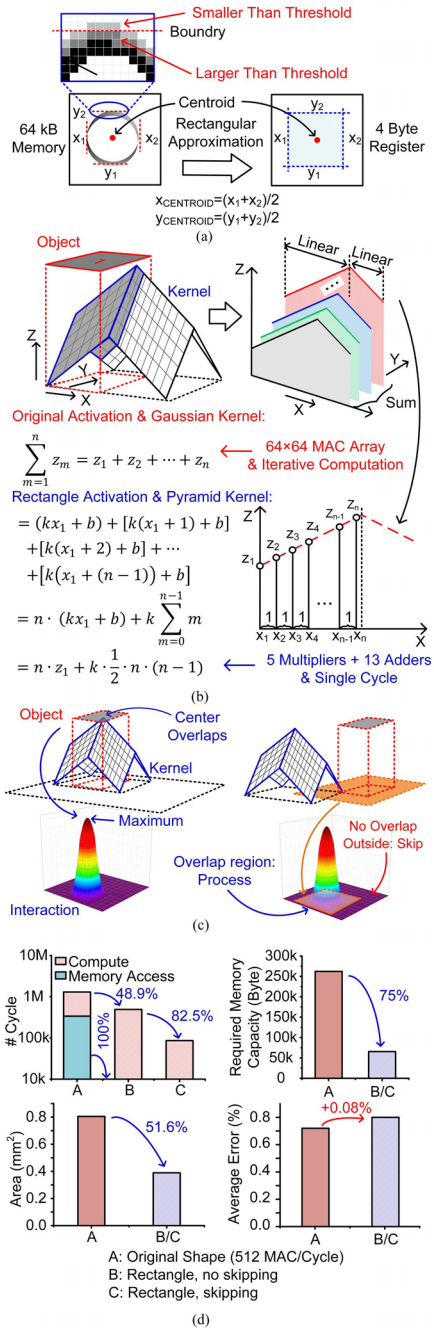

Fig. 4. Diagram of shape approximation and performance comparison. (a) Method of finding the centroid and replacing the shape with a rectangle. (b) Convolution simplification for the pyramid-shaped kernel and rectangle. (c) Method of finding maximum and skipping nonoverlapped region. (d) Performance comparison.

a 48.9% reduction in processing time. Adding the skipping of the positions outside the range generates an additional 82.5% time-saving. The proposed shape approximation reduces 75% of the memory and 51.6% of the total area while adding 0.08% of the tracking error.

IV. Circuit Implementation

Fig. 5 describes the block diagram of the proposed DNF accelerator, applying the techniques introduced in Section III. The system comprises an event-to-frame converter (EFC), a DVS frame FIFO (DFF), four DNF processing engines (PEs), and a finite-state-machine-based controller (CTL). Each PE consists of a range comparator (RGC), a convolution sum calculation unit (SUM),

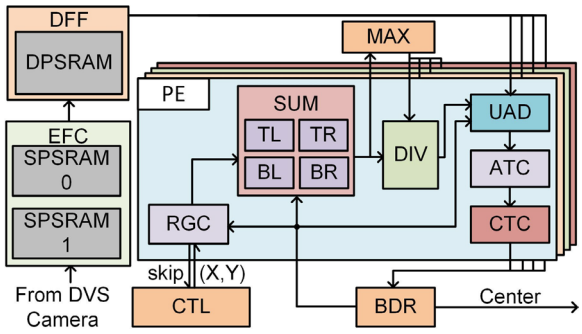

Fig. 5. Diagram of the proposed system architecture.

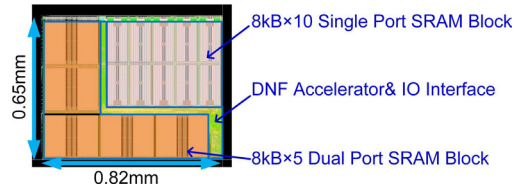

Fig. 6. Layout of the proposed DNF accelerator.

a divider (DIV), the update adder (UAD), an activation comparator (ATC), the center-finding comparator (CTC), and the boundary register (BDR).

To support DVS camera resolution up to 640 × 480 [7], EFC connected to the DVS camera converts events into frames. It integrates 80-kB single-port SRAM blocks partitioned into two groups (SPSRAM0 and SPSRAM1). The two groups alternately undertake the task of receiving address-event representation (AER) [15] signals and storing the converted DVS frames into DFF. DFF includes 40-kB dual-port SRAM blocks (DPSRAM), storing one entire DVS frame. CTL counts the horizontal and vertical coordinates that traverse the entire 2-D activation map. The current position is first sent to RGC, which monitors whether the received position is within the range of the effect area. If inside the effective area, the position is sent to SUM for the calculation of the sum of the arithmetic progression. Since the pyramid-shaped kernel is piecewise linear, four identical blocks are included in SUM, each calculating a quarter of the kernel [top-left (TL), top-right (TR), bottom-left (BL), and bottom-right (BR)]. The convolution result is divided by the maximum value in DIV for normalization, and UAD adds the current activation value, input DVS event, and the normalized interaction to obtain the updated activation. ATC activates the UAD output to either 0 or 1, and then the CTC monitors the position as the boundary of the rectangle and updates BDR. The first PE also receives the center position, used for computing the maximum convolution result. Its value is stored in a maximum value register (MAX), and other data paths read this value for normalization. Targeting a circuit area less than 0.5 mm^2, the proposed design includes four PEs and obtains the synthesized area of 0.42 mm^2.

V. Experimental Results

The prototype design is implemented in a 28-nm CMOS process with an area of 0.53 mm^2 as shown in Fig. 6. The netlist generated from place-and-route is used for transistor-level simulation to ensure accurate power analysis. To maintain acceptable simulation time, parasitic extraction is not applied. The logic cells are simulated using models at the TT corner with the Synopsys CustomSim Simulator. In addition, the power consumption of memory cells is derived from the datasheet created by the memory compiler.

Fig. 7(a) demonstrates the tracking of moving vehicles with the DVS dataset from the Metavision traffic_monitoring [16]. The frames are scaled to 256 × 256. The evolution of the DNF shows that the

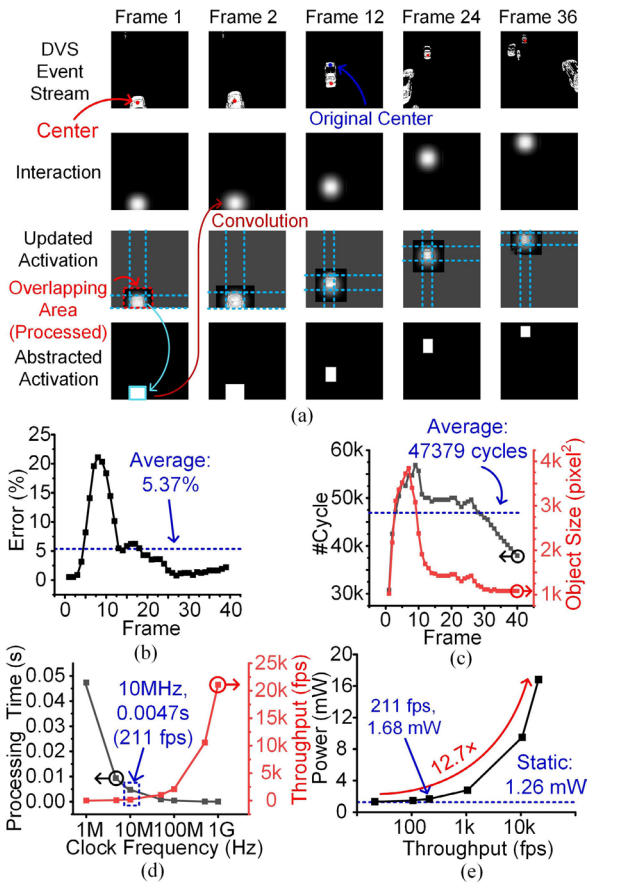

Fig. 7. Tracking of the vehicle in the moving traffic and the performance and consumption. (a) Exemplary images. (b) Error. (c) Number of cycles and target size. (d) Processing time and throughput across frequency. (e) Power across throughput.

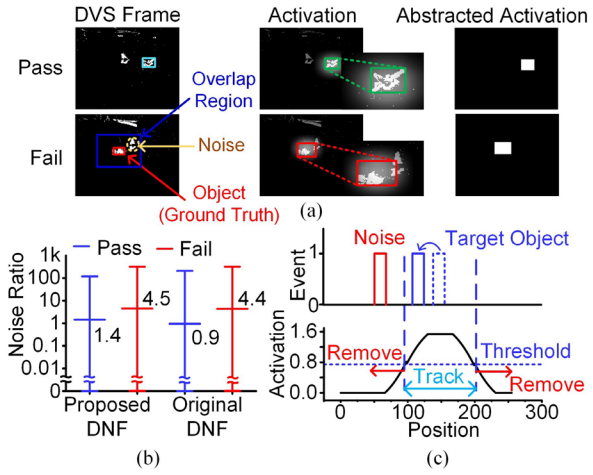

Fig. 8. Effect of input noise to the activation. (a) Passing and failure cases and the definition of noise. (b) Distribution of noise ratio for passing and failure conditions. (c) Effect of threshold toward the range of object and noise.

convolution of the current activation and the pyramid kernel results in a circle interaction with the highest values at the center point. The overlapped area forms a rectangle, encompassing the range within which the object can move while excluding other objects. The highlighted target object has its four edges detected and is abstracted into a rectangle again. The tracking error compared with the original DNF method (512 × 512 Gaussian kernel, activated with sigmoid function and keeping origin shape) is shown in Fig. 7(b). The hardware accelerator generates 5.37% of center shifts on average. At the beginning frames, the object is close to the camera so that it is

TABLE I
PERFORMANCE COMPARISON

	TCAS-II 2023 [18]	TCAS-II 2023 [19]	AICAS 2020 [11]	This Work
Supply Voltage (V)	1.1	0.89 - 1.2	0.5 - 1.25	1
Approach	DNN	DVS+SNN	DVS+DNF	DVS+DNF
Area (mm^2)	23	107.2	60	0.53
Throughput (fps)	69.8	100	2000	211.1
Power (mW)	71.2	208.1	N/A	1.68

large and moves fast. The tracking centers focus on different parts of the same vehicle, which leads to larger errors near the tenth frame, such as Frame 12 of Fig. 7(a). However, both the tracking centers are within the target object as shown in the second image of Fig. 7(a). Fig. 7(c) implies the number of cycles and object size. More positions are skipped as the size reduces when the object moves further. The average cycle is 47.4k, which brings 211 fps under a 10-MHz clock as indicated by Fig. 7(d). The EFC combines 2.4k events into a single frame at 211 fps. Objects in the frame appear clearer if more events are captured at a slower frame rate. Fig. 7(e) shows the average power consumption across throughput. The power increases by 12.7× as throughput increases from 21 to 21k fps. The static power is 1.26 mW, which takes the major portion at low frequency. At a frame rate of 211 fps, the accelerator consumes 1.68 mW of power.

Fig. 8 displays the effect of noise in the DVS images. About 23 test cases out of 32 from the FE-108 dataset [17] are applied. The unused test cases are due to the large file size and exceed the simulation capability. The events within the ground-truth box are considered as the object and those outside the box while within the overlap area are calculated as the noise. The ratio of noise and object event is calculated to represent the noise intensity. When the tracking results and ground-truth overlap by 50%, the result is considered as passing the test of the current frame and otherwise considered as failure. Per the noise being close to the object event, part of the noise object may get involved and detected as the target, bringing a larger detected area and resulting in the failure, as shown in Fig. 8(a). Fig. 8(b) shows the distribution of noise ratio level leading to passing and failure for both the proposed and original DNFs. As failure cases result from stronger noise, the proposed DNF tolerates a similar level of noise intensity compared with the original. Fig. 8(c) shows a row of 1-D activation in the 2-D array. Assuming all the input events are 1, the threshold determines the region that is recognized as the target object. With a higher threshold, the range that the object can move becomes narrower, limiting the acceptable speed. On the other hand, setting a lower threshold allows faster moving speed but leads to more noise events being recognized as the target object.

Table I shows comparison of the proposed DNF tracking system with the state-of-the-art approaches. Among the approaches, this work occupies the smallest area and consumes the lowest power, which is significantly important for miniature systems. Basing on deep neural network (DNN), the work proposed in [18] detects objects from conventional RGB images and does not require DVS cameras, but the DNN inference consumes a long processing time, limiting the frame rate. The design of [19] uses SNN to predict the position that uses historical data as input for higher accuracy. However, it requires irregular-shaped kernels and MAC operations so that hardware simplification similar to our proposed idea cannot be applied. The work of [11] using DNF achieves the highest throughput among the works being compared, but the design cannot be implemented in our target miniature system since the Loihi platform consumes a large area and requires control from a personal computer. In addition, the design integrates an extra layer to highlight all the objects and another self-sustained layer to track the target object.

VI. CONCLUSION

In this brief, we propose a hardware accelerator for tracking DVS inputs. The proposed accelerator offers improved speed with reduced area by optimizing the kernel size, replacing the sigmoid function with a comparison, and simplifying the object shape to a rectangle. The implemented layout indicates that the design occupies a silicon area of 0.53 mm^2. Simulation results show that the processing speed reaches 211 fps, with a power consumption of 1.68 mW.

REFERENCES

[1] M. H. Ghaed et al., "Circuits for a cubic-millimeter energy-autonomous wireless intraocular pressure monitor," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 60, no. 12, pp. 3152–3162, Dec. 2013.

[2] I. Lee et al., "MSAIL: Milligram-scale multi-modal sensor platform for monarch butterfly migration tracking," in *Proc. 27th Annu. Int. Conf. Mobile Comput. Netw.*, Oct. 2021, pp. 51–530.

[3] G. Kim et al., "A millimeter-scale wireless imaging system with continuous motion detection and energy harvesting," in *Proc. Symp. VLSI Circuits Dig. Tech. Papers*, Jun. 2014, pp. 1–2.

[4] A. Bejarano-Carbo et al., "Millimeter-scale ultra-low-power imaging system for intelligent edge monitoring," 2022.

[5] J. Haverinen, M. Parpala, and J. Roning, "A miniature mobile robot with a color stereo camera system for swarm robotics research," in *Proc. IEEE Int. Conf. Robot. Autom.*, Apr. 2005, pp. 2483–2486.

[6] R. Kubendran, A. Paul, and G. Cauwenberghs, "A 256×256 6.3pJ/pixel-event query-driven dynamic vision sensor with energy-conserving row-parallel event scanning," in *Proc. IEEE Custom Integr. Circuits Conf. (CICC)*, Apr. 2021, pp. 1–2.

[7] B. Son et al., "4.1 A 640×480 dynamic vision sensor with a 9μm pixel and 300Meps address-event representation," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2017, pp. 66–67.

[8] D. Han, J. Lee, J. Lee, S. Choi, and H.-J. Yoo, "A 141.4 mW low-power online deep neural network training processor for real-time object tracking in mobile devices," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2018, pp. 1–5.

[9] Y. Sandamirskaya, S. K. U. Zibner, S. Schneegans, and G. Schöner, "Using dynamic field theory to extend the embodiment stance toward higher cognition," *New Ideas Psychol.*, vol. 31, no. 3, pp. 322–339, Dec. 2013.

[10] J. N. P. Martel and Y. Sandamirskaya, "A neuromorphic approach for tracking using dynamic neural fields on a programmable vision-chip," in *Proc. 10th Int. Conf. Distrib. Smart Camera*, Sep. 2016, pp. 148–154.

[11] A. Renner, M. Evanusa, and Y. Sandamirskaya, "Event-based attention and tracking on neuromorphic hardware," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2019, pp. 1709–1716.

[12] W. Giernacki, M. Skwierczynski, W. Witwicki, P. Wronski, and P. Kozierski, "Crazyflie 2.0 quadrotor as a platform for research and education in robotics and control engineering," in *Proc. 22nd Int. Conf. Methods Models Autom. Robot. (MMAR)*, Aug. 2017, pp. 37–42.

[13] 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," *Int. J. Robot. Res.*, vol. 36, no. 2, pp. 142–149, Feb. 2017.

[14] M. Singh, B. S. Chauhan, and N. K. Sharma, "VLSI architecture of centroid tracking algorithms for video tracker," in *Proc. 17th Int. Conf. VLSI Design*, Jan. 2004.

[15] K. A. Boahen, "Point-to-point connectivity between neuromorphic chips using address events," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 47, no. 5, pp. 416–434, May 2000.

[16] *Prophesee Metavision Recordings and Datasets*. Accessed: Jun. 24, 2024. [Online]. Available: https://docs.prophesee.ai/stable/datasets.html#recordings-and-datasets

[17] H. Zhang et al., "In the blink of an eye: Event-based emotion recognition," in *Proc. ACM SIGGRAPH Conf.*, 2023, pp. 1–11.

[18] Y. Gong et al., "An energy-efficient visual object tracking processor exploiting domain-specific features," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 71, no. 5, pp. 2794–2798, May 2024.

[19] K. Liu et al., "Real-time target tracking system with spiking neural networks implemented on neuromorphic chips," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 70, no. 4, pp. 1590–1594, Apr. 2023.