2021 IEEE 3rd International Conference on Artificial Intelligence Circuits and Systems (AICAS) | 978-1-6654-1913-0/20/$31.00 ©2021 IEEE | DOI: 10.1109/AICAS51828.2021.9458497

Hardware-Algorithm Co-Design Enabling Efficient Event-based Object Detection

Brian Crafton[1], Andrew Paredes[1], Evan Gebhardt , and Arijit Raychowdhury[1]
[1]Georgia Institute of Technology, School of ECE, Atlanta, GA

***Abstract*—Event-based cameras are a promising alternative to traditional optical cameras for real time computer vision systems. They offer low power, high temporal resolution, and high dynamic range, making them an ideal candidate for resource-constrained edge computing. In this work, we evaluate system level designs using event-based cameras for computer vision on the edge. We present a new technique to improve the object detection performance of event-based cameras, and demonstrate a 1.88× to 2.17× improvement in energy efficiency.**

I. Introduction

In less than a decade, tremendous progress towards accelerating deep neural networks (DNN) has been made enabling orders of magnitude improvement in both performance and efficiency. DNNs have been compressed, pruned, and quantized to minimize the total size of the model and cost of a single inference [1]. Custom hardware accelerators have been developed [1] to maximize the reuse of all data such that expensive memory accesses and total data movement is minimized. At the same time, technology scaling, advanced packaging technologies, and emerging memories [2] all seek to contribute to improved energy efficiency for machine learning systems. These improvements have contributed to the immense interest from both industry and academia in moving applied machine learning and artificial intelligence closer to the edge, where data is collected.

At one point, a 200W GPU was required to implement a real time detection system [3]. Hence, the total power consumption of any computer vision system was dominated by the DNN. However commercially available ASICs or FGPAs can implement these systems in real time with less than 1W of power. In fact, we find that the camera for a low power real time object detection system now consumes more power than the hardware required to perform computer vision applications. For this reason, we must look to low power camera designs if we wish to continue to improve the energy efficiency of real time computer vision applications at the edge. One such design is the event based camera or dynamic vision sensor (DVS). Naturally, these new cameras come with their own challenges. Due to the lack of existing work and smaller pixel count, DVS perform worse than traditional optical cameras on tasks like object detection.

In this work, we design a low power object detection system using an event-based camera and explore trade-offs to improve the detection performance. We perform a system-level design and evaluate power, performance, and detection accuracy of an event-based system compared with traditional optical cameras. To benchmark, we use the recently released Gen1 Automotive Detection (GAD) dataset [4]. To improve the detection performance of DVS, we exploit the time component in the event stream by stacking several event sequences rather than processing static images. Furthermore we explore the tradeoff in frame rate versus power consumption and demonstrate that by using a DVS, we can improve energy efficiency by 1.88× to 2.17× over a standard camera for an object detection system while achieving competitive detection accuracy.

II. Background and Motivation

Event-based cameras promise a new visual sensor with exciting properties that have been absent in traditional optical cameras. Rather than capturing a series of dense digital images at a fixed frame rate to form a video, event-based cameras generate streams of visual events. Each event is triggered asynchronously, which occur when an significant illumination change occurs at a specific pixel.

Events are typically encoded as tuples containing the x and y pixel coordinates, a timestamp, and a polarity change (increase or decrease in illuminance):

$$e_i = (x_i, y_i, t_i, p_i)$$

Since the camera operates asynchronously, events are sampled as soon as they occur. Therefore there is high temporal resolution and thus latency on the order of microseconds rather than tens of milliseconds (30 to 60 FPS). In Figure 1, we provide an illustration of the data generated by a event-based camera and how it may be processed to resemble traditional optical cameras.

Event-based cameras promise ultra low power consumption even at relatively high sample rates and pixel densities. A recent demonstration [5] has shown a 1 mega-pixel camera capable of 300 million events per second consuming between

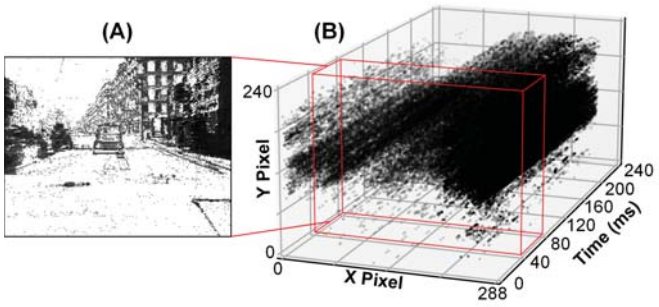

Fig. 1. Example data from event-based camera. (A) Event stream as point cloud. (B) Time slice of point cloud as an image.

32 mW and 84 mW of power. In comparison, a *GoPro* consumes over 2.5W while recording at 60 FPS. Lower power cameras compatible with commercially available hardware such as Nvidia Jetson Nano [6] consume as low as 200mW to 500mW depending on frame rate and resolution.

The power reduction enabled by event-based cameras motivates their use in systems that require very low power. With advances in DNN accelerators, we find that commercially available hardware like TPU Edge [7] or Nvidia Jetson Nano [6] consume less than or equal to the power dissipated in a low power optical camera. Furthermore, we can expect more improvements in energy efficiency. The Edge TPU [7], boasts 2 TOPs/W at 28nm technology and using standard LPDDR3. With a more advanced technology node and embedded DRAM, higher efficiency can be achieved enabling new applications on the edge.

One example application is object detection. Traditional object detection attempts to identify and locate objects in an image. Popular datasets like COCO [8] provide bounding box coordinates and class labels, where the objective is to predict a region where the object resides and the class the object belongs to. Since the popularization of deep learning, convolutional neural networks have been applied to this task with great success.

Despite this progress, event based object detection is still an open challenge. Recent work [9], [10] has provided techniques to convert event streams into frames so that DNNs and modern computer vision techniques can be used. In [9], streams of events (time surfaces) are turned into histograms that resemble images. However, these time surfaces depict only the moving parts of the image with emphasis on the edges of objects. In [10], a DNN is used in an attempt to recreate a grayscale image from an event stream. Hence, one DNN must first be trained to map event streams to more detailed grayscale images, while a second DNN is used to perform object detection from the grayscale images.

III. Event Based Object Detection

To make use of existing work in object detection, we must generate frames from our event stream that can be input to a CNN. This can be done by accumulating all the events in a certain time period and recording their pixel polarity values in a 2D frame. This formulation can be written as:

$$\forall e \in E(t, t+\tau) : F(e_y, e_x) \mathrel{+}= e_p$$

where $E(t, t+\tau)$ is the set of events in a time period and F is the 2D frame we construct to resemble a traditional image. However, this simple method suffers from several limitations. First, noisy events are not filtered using spatial and temporal correlation in the event stream. Second, the time component from the events is lost because we discard the timestamps and only record the coordinates and polarity of the event.

A. Filtering Noisy Events

Given that events occur in both space and time, we expect strong correlation in the spatial and temporal coordinates of events rather than random noise. However, noise and non-idealities in the sensor lead to random events that should be filtered out. Prior work [9] addresses this problem by analyzing the spatial coordinates of recent events and then filters out events that occur outside high correlation. Spatio-temporal windows (x, y, t) are established for each event. Next a convolution-like operation called the *time surface* [9] is applied to each event neighborhood to suppress noise. This technique is given by the following equation, which we adopt in our detection system.

$$\Phi_{xyt}(e_i) = \sum_{e_j \in N_{xyt}} e^{-(t_i - t_j)/\tau}$$

B. Frame Stacking

Object detection has traditionally been applied to images in popular datasets like COCO [8]. However, it has also been applied to videos, where detections are made for each frame. Naturally, several works have attempted to use recurrent neural networks [11] to exploit the temporal component of the video. This is because objects in the previous frame are likely to also occur in the current frame. Furthermore, the velocity and direction of objects can assist in detection and classification. To illustrate this idea, we divide a 100ms period from the GAD dataset into 12 frames and observe the change over time. We visualize this in Figure 2, where frames 1 and 12 from an event stream are displayed. From frame 1 to frame 12, it is clear that the car and pedestrian have moved from their prior locations. By providing the CNN several frames rather than a single frame, we allow it to extract and exploit these features for improved performance.

Building off this observation, we seek to model our event stream in a way that preserves the temporal component of the data. Thus for each event sequence, we generate up to 12 frames which we stack along the channel dimension. While this technique loses fine grain temporal resolution of events in the same frame, it allows us to process events as dense matrices (rather than point clouds) using efficient modern object detectors. We can control the fine grain temporal resolution by increasing or decreasing the number of frames an event stream is converted to. Naturally this results in increased memory and computation, and in Section V we explore this trade-off. We choose to process all the frames together as if the time dimension was the channel dimension of a standard CNN. For example, our first convolutional filter is sized $7\times7\times N_f\times64$,

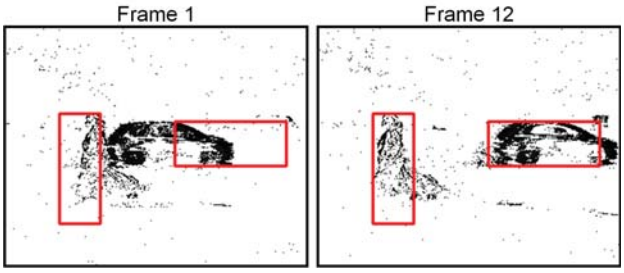

Fig. 2. Frames 1 and 12 from the event stream with bounding boxes. Encoding the event stream as 12 frames preserves the time dimension.

where N_f is the number of frames we have divided our event stream into. It should be noted that we could use a convolutional LSTM, however we avoid this because of the increased compute requirement. A convolutional LSTM would require that we run CNN N_f times for each detection, and given that we make 30 detections per second it is expensive for an edge detection system.

C. Dataset and Pre-Processing

A primary challenge in improving the viability of event based vision for object detection has been the lack of a large scale event based dataset. However several recent datasets have been released to serve as a benchmark and to evaluate the feasibility of event-based object detection. The GAD Dataset [4] is the first large dataset on event-based object detection with labeled pedestrians and cars. The dataset consists of 39 hours of recording with over 228k cars and 27k pedestrians.

To generate the image sequences and corresponding bounding boxes, we parsed through the 39 hours of video in the GAD Dataset. Bounding boxes are provided at 1Hz, 2Hz, or 4Hz throughout the video. For example, at 4Hz bounding boxes are provided every 250ms. For every bounding box, we constructed frames based on the events from the 100ms leading up to the bounding box. This resulted in 102,600 training image sequences and 12,700 validation image sequences. To improve generalization and avoid overfitting, we used dropout, batch normalization, and image augmentation. We randomly flipped, rotated, and cropped images to create 8 permutations per image to increase our training set.

D. Model and Loss Function

To implement our network, we use a variation of YOLOv3 [3], where the base of our network is a pre-trained ResNet18 model. After which we add two fully connected layers to predict a 12×10 grid of 7 anchor boxes each. Anchor boxes are chosen using the k-means algorithm [3] on the bounding boxes of the training set. Mean squared error is used for the bounding box error and confidence error, but categorical cross entropy is used for class error. Because there are only two classes (car and pedestrian), we predict a class at each anchor box. Hence, each anchor box predicts a 7-dimensional vector and the final output of our network is $10 \times 12 \times 7 \times 7$. After training, we perform 8-bit quantization to reduce computational cost.

IV. System Architecture

To implement our detection system and evaluate the value of an event-based camera, we design and evaluate the performance of a custom SoC through simulation using 16nm technology at 600 MHz. Our system level design can be divided into several different modules shown in Figure 3. In the following subsections we detail the hardware implementation of our design.

A. Processing Event Streams

To generate frames from the event stream, we design a custom hardware module called the aggregation unit. This module interfaces the input stream from the event based camera and stores the most recent frames in DRAM using a circular buffer. Given that there is typically high spatial and temporal locality between events, caching can be exploited to minimize DRAM accesses. Therefore, an SRAM bank is used by the aggregation unit to cache pixels and update the most recently received events. In addition to caching pixels, the aggregation unit performs filtering via time surfaces as described in Section II. To implement the time surface efficiently, we approximate the exponential with a lookup table and shift operations. To approximate power and performance, we implemented this module using Verilog and performed post-synthesis power analysis.

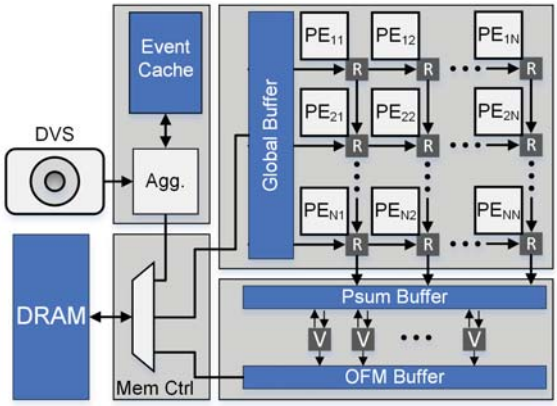

Fig. 3. System level design including SoC, event-based camera, and DRAM.

B. SoC Architecture

In addition to pre-processing the event stream, our SoC implementation requires several modules for performing object detection. To implement our object detector, we use a systolic array, a commonly used architecture for accelerating DNNs. Our systolic array design is a scaled down version of the Edge TPU [7] to meet our performance requirement, consisting of only 24×24 processing elements clocked at 600MHz. To enable real time performance, we simulate the systolic array on samples from our training set and compute the minimum array size required to minimize leakage power and area. In theory, our array size could be further reduced, however we find that for most of the matrix dimensions it is not possible to achieve peak throughput and thus 24×24 works well.

To store the model's weights, bounding box predictions, and input frames we use 1GB of LPDDR3 operating at 1600 MHz. To approximate power and performance for this DRAM we use *DRAM Power* [12], which yields roughly 65.2 mW for our model. To feed the array and store partial sums, we use 2MB of SRAM to cache the weights and input features for each layer. The results of convolution are cached collected in this bank and non-linear operations are applied using custom vector units. This custom vector unit performs SIMD operations on the results and is used to perform bias addition, ReLU, pooling, and quantization.

V. Results

To evaluate the advantages of using event-based cameras in low power detection systems, we compare power and perfor-

mance versus a system using an optical camera. Unfortunately, the GAD dataset does not have corresponding optical (RGB) video to compare the two for detection performance. Hence, we are only able to compare power and performance of the system and observe that the detection results are competitive to results achieved on other datasets. Furthermore, we expect that future datasets will contain RGB video for comparison and that further research will close the gap between algorithms for event-based vision algorithms and optical cameras.

To evaluate the advantages of frame stacking, we trained our model with varying numbers of frames and provide our results in Table I. Detection performance is measured as mean average precision (mAP) defined by the widely used COCO dataset. A correct detection requires predicting a bounding box with >50% IoU and the correct class (car or pedestrian). We find that by increasing the number of frames, and thus the temporal resolution, we achieve higher detection performance. At 1 frame we observe 32.0% mAP, while 12 frames yields 39.6%, for a 7.6% improvement. However, as we increase this past 8 to 12 frames, the detection improvement diminishes while requiring additional compute, memory capacity, and memory bandwidth. Given the brief time the GAD dataset has been available, there exists only one other work to compare our results with. We find that our 12 frame model yields equivalent performance to previous work [13], where 40% mAP was achieved using a convolutional LSTM. Our code and trained models are available at: https://github.com/bcrafton/event-based-vision.

To compare the efficiency of the event-based camera design with a system based on an optical camera, we compute the power consumption of each component in the system and visualize the breakdown in Figure 4. Each design uses the same systolic array and DRAM described in Section IV. All the event-based designs use the aggregation module for pre-processing event streams, and we assume 51 mW for the DVS based on the sampled event streams from the dataset and power specification of [5]. As reference, we present a power breakdown for a system using an optical (RGB) camera running the same workload. For camera and ISP (Image Signal Processor) power, we use the low power configuration for the

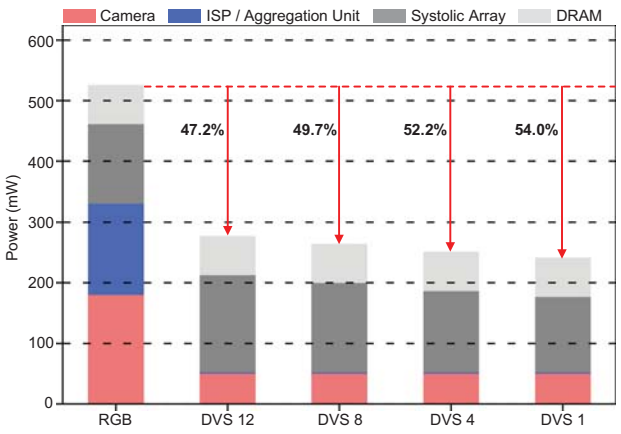

Fig. 4. SoC power consumption by method

TABLE I
DETECTION RESULTS ON GEN1 AUTOMOTIVE DETECTION

Camera	# Frames	mAP (%)	Energy (mJ)	Latency (ms)
Optical	1	-	4.37	33.7
DVS	1	32.0	4.15	32.0
DVS	4	37.3	4.48	34.5
DVS	8	38.6	4.91	37.9
DVS	12	39.6	5.34	41.2

popular AR1335 [14] and Nvidia Jetson ISP [6]. In Figure 4, we use the same key for ISP and aggregation unit since they serve the same function. It is clear from our power breakdown, that even at modest 16nm CMOS, the camera and ISP account for a large fraction of power dissipation and thus provide a opportunity to greatly reduce system power with a low power alternative.

VI. CONCLUSION

In this work we design and evaluate an object detection system based on event-based vision sensors. To make use of existing work in object detection and utilize the time component in event-based data, we convert event streams into sequences of images rather than a single image. This results in a 7.6% increase in mAP, while incurring a small energy and latency overhead. Furthermore, We find that using an event-based vision sensor reduces system power by 1.88× to 2.17× motivating its use in energy constrained edge systems.

REFERENCES

[1] V. Sze *et al.*, "Efficient processing of deep neural networks: A tutorial and survey," *Proceedings of the IEEE*, vol. 105, no. 12, pp. 2295–2329, 2017.

[2] B. Crafton *et al.*, "Merged logic and memory fabrics for accelerating machine learning workloads," *IEEE Design & Test*, 2020.

[3] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[4] P. d. T. Nitti *et al.*, "A large scale event-based detection dataset for automotive," *arXiv preprint arXiv:2001.08499*, 2020.

[5] T. Finateu *et al.*, "5.10 a 1280× 720 back-illuminated stacked temporal contrast event-based vision sensor with 4.86 μm pixels, 1.066 geps readout, programmable event-rate controller and compressive data-formatting pipeline," in *2020 IEEE International Solid-State Circuits Conference-(ISSCC)*, pp. 112–114, IEEE, 2020.

[6] NVIDIA, *NVIDIA Jetson TX2*, 2020.

[7] Google, *Google Edge TPU*, 2020.

[8] T.-Y. Lin *et al.*, "Microsoft coco: Common objects in context," in *European conference on computer vision*, pp. 740–755, Springer, 2014.

[9] A. Sironi *et al.*, "Hats: Histograms of averaged time surfaces for robust event-based object classification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1731–1740, 2018.

[10] H. Rebecq *et al.*, "Events-to-video: Bringing modern computer vision to event cameras," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3857–3866, 2019.

[11] G. Ning *et al.*, "Spatially supervised recurrent convolutional neural networks for visual object tracking," in *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1–4, IEEE, 2017.

[12] K. Chandrasekar *et al.*, "Improved power modeling of ddr sdrams," in *2011 14th Euromicro Conference on Digital System Design*, pp. 99–108, IEEE, 2011.

[13] E. Perot *et al.*, "Learning to detect objects with a 1 megapixel event camera," *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[14] O. Semiconductor, *AR1335 CMOS Image Sensor*, 2020.