2023 IEEE International Conference on Robotics and Automation (ICRA 2023)
May 29 - June 2, 2023. London, UK

DOTIE - Detecting Objects through Temporal Isolation of Events using a Spiking Architecture

Manish Nagaraj, Chamika Mihiranga Liyanagedera and Kaushik Roy

***Abstract*— Vision-based autonomous navigation systems rely on fast and accurate object detection algorithms to avoid obstacles. Algorithms and sensors designed for such systems need to be computationally efficient, due to the limited energy of the hardware used for deployment. Biologically inspired event cameras are a good candidate as a vision sensor for such systems due to their speed, energy efficiency, and robustness to varying lighting conditions. However, traditional computer vision algorithms fail to work on event-based outputs, as they lack photometric features such as light intensity and texture. In this work, we propose a novel technique that utilizes the temporal information inherently present in the events to efficiently detect moving objects. Our technique consists of a lightweight spiking neural architecture that is able to separate events based on the speed of the corresponding objects. These separated events are then further grouped spatially to determine object boundaries. This method of object detection is both asynchronous and robust to camera noise. In addition, it shows good performance in scenarios with events generated by static objects in the background, where existing event-based algorithms fail. We show that by utilizing our architecture, autonomous navigation systems can have minimal latency and energy overheads for performing object detection.**

I. Introduction

Autonomous navigation is emerging as an important area of research, with applications ranging from videography and surveillance in remote regions to transportation systems in populated urban areas. The Society of Automation Engineers (SAE) has identified six levels of automation for autonomous navigation [1], starting from systems with no automation (*level 0*) to systems that are fully automated and do not require any operator (*level 5*). One of the most basic and essential requirements of a system with automation (*level 1* and above), is the ability to detect and avoid obstacles. And most importantly, these systems must be able to perform object detection accurately at very high speeds.

Recent advancements in imaging technology have led to the development of biologically inspired *event cameras* [2]–[5]. While traditional *frame cameras* capture photometric features such as light intensity and texture with high spatial resolution at fixed intervals, event cameras are asynchronous and only capture the change in light intensities at each pixel. Although event cameras fail at capturing photometric features, they do not suffer from issues such as motion blur and can operate at a much higher frequency and under a wider range of illumination. Owing to their higher output frequency, event cameras can capture high resolution temporal information that are missed by frame cameras. Since object detection and subsequent applications such as collision avoidance need to be performed at a very high speed and over a wide range of illumination, event cameras become ideal candidates for this task.

While there has been a plethora of research on object and feature detection algorithms, most of these algorithms are designed for frame camera outputs. These algorithms fail on event camera outputs as they rely on photometric features that are only present in frame data. For example, Fig. 1. shows the performance of both, a preliminary edge detection algorithm - the canny filter [6], and a complex learning based object recognition algorithm - YOLOv3 [7], on a frame camera output and an event camera output of the same scene. While the algorithms can successfully function on the former, they fail to function on the latter.

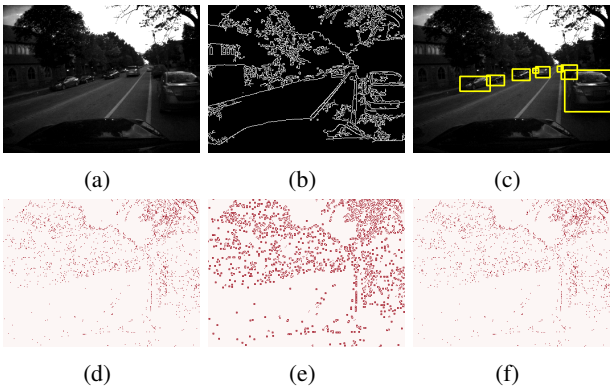


Fig. 1: Example to show the inability of traditional computer vision algorithms to perform on event camera outputs. (a) and (d) show the frame and event camera outputs captured on the same scene (taken from the MVSEC dataset [8]). (b) and (e) show the outputs of applying a canny filter on the respective images. (c) and (f) show the output of YOLOv3 on the respective images. We can see that both canny filtering and YOLOv3 fail to perform on the event camera output.

This motivates us to explore and utilize temporal features that are inherently present in the events, but are ignored in traditional computer vision algorithms. Leveraging such temporal information in events can help us in detecting and differentiating objects in a scene. For this purpose, we identify two properties important for object detection:

(i) Events generated by the same object are temporally close to each other.

*This work was supported in part by, Center for Brain-inspired Computing (C-BRIC), a DARPA sponsored JUMP center, Semiconductor Research Corporation (SRC), National Science Foundation, the DoD Vannevar Bush Fellowship, and IARPA MicroE4AI.

All authors are with Purdue University, West Lafayette, IN 47907, USA {mnagara, cliyanag, kaushik}@purdue.edu

2023 IEEE International Conference on Robotics and Automation (ICRA) | 979-8-3503-2365-8/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICRA48891.2023.10161164

(ii) Events generated by the same object are spatially close to each other.

Based on these two properties, we aim to group events along two dimensions - spatial and temporal, to isolate events based on their source objects and thus identify the object boundaries.

Biologically inspired spiking neurons, specifically *Leaky Integrate and Fire* (LIF) neurons [9], can leverage spatio-temporal information and are hence, well suited for achieving boundary detection along both these dimensions. These neurons mimic the brain activity and behave based on the temporal properties of the inputs to the neuron. The neuron generates an output spike only if the input events occur at a rate higher than a certain frequency. A neural architecture with these spiking neurons is sensitive to the temporal structure of input events. It is also possible to arrange the connectivity between these neurons in a network to enable support between events that are spatially close to each other.

Further, spiking neurons work in an asynchronous fashion, making them compatible with the asynchronous outputs of event cameras. These properties, along with the fact that spiking neural architectures can be more energy efficient (as shown in [10]–[12]) than their artificial neural counterparts, make them an ideal candidate for performing object detection with a low latency and energy overhead for autonomous navigation systems.

In this work, we develop a novel and energy efficient object detection technique to first isolate objects based on the speed of their movement. This is done using inputs from an event camera and a lightweight single layer spiking neural network. Once objects are separated based on their speed of movement, their corresponding events are then grouped together based on their spatial characteristics. For this, we utilize existing clustering techniques to further separate the events belonging to different objects, based on their spatial proximity. Here, we are taking advantage of the fact that the events from the same object, share similar temporal characteristics such as their speed of movement, and spatial characteristics such as being generated by pixels that are spatially close together. Our experiments show that this approach is a more efficient way to detect objects in terms of both latency and energy consumption. By isolating events based on the speed of movement of their corresponding objects, we are also able to eliminate the non-relevant information caused by noise and background static objects[1]. Further, this benefits the clustering techniques as they have a lower operational complexity due to the reduced number of samples (events) that need to be clustered.

We summarize the main contributions as follows:

1) We develop an object detection algorithm that solely relies on the event camera outputs and does not need any additional information from traditional frame cameras.
2) Unlike many existing works on object detection which accumulate events for a time duration to create a frame, we perform object detection asynchronously as the events are being generated by the event camera.
3) The proposed spike-based network for separating objects based on their motion consists of a single layer, resulting in a detection algorithm with lower latency and energy overhead.
4) The outputs of the proposed spiking architecture (which isolates objects based on their speed), can be used with any spatial clustering technique that does not require prior knowledge of the number or the size of clusters.
5) The spiking architecture is scene independent. This means that we do not have to train the parameters of the architecture based on the scene of deployment. These parameters directly correspond to the speed of the objects and can be fine-tuned prior to deployment.

[1]If there is a need to detect static objects, this can be done by clustering the residual events that do not propagate through the spiking architecture.

II. Related Work

A. Object Detection in the Event Camera Domain

Object detection is a topic that has been extensively studied in the computer vision community. There have been works ranging from simple feature detectors [6], [13], to more complex learning-based methods [14]. There has also been a significant interest in the learning community on neural networks that can not only detect objects, but can also classify them into different classes [7]. However, when it comes to autonomous navigation where object classification is not a priority, the latency and energy efficiency of the underlying algorithms take precedence.

As discussed earlier, traditional frame-based algorithms fail to operate on event camera outputs due to the absence of photometric characteristics such as texture and light intensity. However, owing to the numerous advantages of event cameras, including higher operating speed, wider dynamic range, and lower power consumption, there has been a substantial interest in the community to develop algorithms that are more suited towards this domain.

Initial event-based detection algorithms such as [15], [16] were focused on detecting patterns present in the event camera output. The authors in [17] used a simple blob detector to detect the inherent patterns present in the event data, and [18] used a plane fitting method on the distribution of events to identify corners. A recent work adapted Gaussian mixture modelling to detect patterns in the event data [19]. These methods, however, fail in scenarios where there are events generated by the background. As a solution, [20] proposed a motion compensation technique to eliminate events generated by the background, by estimating the system's ego-motion. The optimization involved, however, adds significant latency and computational overhead to the system.

To improve the detection accuracy, several recent research efforts were focused on utilizing information from both frame and event cameras [21], [22]. These hybrid methods detect features on the frames and track the objects through events. Since their detection relies on frame inputs, they cannot operate in scenarios with a wide dynamic range and are computationally expensive.

There have also been efforts in creating dense descriptors or "frames" by accumulating events for a pre-specified duration of time [23], [24]. These frames can then be used with traditional computer vision algorithms. But these implementations show lower accuracy due to the incompatibility of the inputs. To overcome the problem, many learning approaches such as [25]–[29] train Convolutional Neural Networks (CNNs) with such frames as inputs. However, such systems still add an overhead on the latency, and are not asynchronous in nature. Further, with the use of deep neural networks, there is an increased computational power overhead.

B. Spiking Neural Architectures for Handling Events

Spiking Neural Networks (SNNs) operate in an asynchronous fashion and can mitigate the latency issues experienced in learning-based techniques. The authors in [30] use an unsupervised learning technique to learn car trajectories on a freeway captured by an event camera. The works in [31]–[38] utilize SNNs to either detect objects, track them or classify them. Such works, however, either utilize deep SNNs that add a significant computational and latency overhead to the system or require intensive training techniques and lose accuracy when they are deployed in a scene different from the one used to train the network.

In this paper, we propose a network with only a single layer of spiking neurons to capture the temporal information present in event data. This mitigates the energy and latency overhead of the neural architecture. Further, this network does not need to be trained on an extensive amount of data, but rather the hyperparameters are simply fine-tuned prior to deployment. Unlike other works, we observe that there is no accuracy degradation when our algorithm is tested on different scenes of deployment.

C. Spatial Clustering of Events

Clustering techniques such as [39]–[41] prove useful in grouping samples with a low dimensional complexity. Since events can be considered as samples that have spatial and temporal dimensions, there have been attempts to utilize clustering techniques for grouping events and detecting the corresponding object boundaries. [42] uses Mean-shift clustering, proposed in [41], to cluster events, while [43] uses a graph spectral clustering technique to group events to determine object boundaries. However, such techniques assume that only the moving objects generate events. In the method proposed here, we first eliminate the events generated from the background and camera noise enabling the use of spatial clustering techniques. Since both the number and the size of the objects are unknown before deployment, we employ clustering techniques that are independent of such parameters [40].

III. Methodology

A. Separating Objects in the Temporal Domain

As mentioned in Section I, in order to detect objects efficiently, we aim to isolate objects along both the temporal and spatial domains. Spiking neurons are an excellent candidate as they are capable of operating along the temporal domain. The LIF neuron model [9] has two characteristic features. One is an internalized state called *membrane potential* ($U[t]$) and the other is a decay factor known as *leak factor* (β). The neuron is initialized with a *threshold value*, U_{thr}, and an initial membrane potential, $U[t_0]$. The weighted sum of the inputs ($WX[t]$) at each time step t, is accumulated in the membrane potential of the neuron as shown by Eqn (1). The accumulated membrane potential decays over time and the decay rate is controlled by the leak factor. The leak factor denotes how much of the membrane potential is retained for the next time step, i.e., the higher the leak factor, the slower the rate of decay. If the accumulated membrane potential of the neuron exceeds the threshold at any point, ($U[t_n] > U_{thr}$), the neuron emits an output spike and resets its membrane potential.

$$U[t_n] = \underbrace{\beta U[t_{n-1}]}_{\text{decaying membrane potential}} + \underbrace{WX[t]}_{\text{weighted sum of inputs}} \quad (1)$$

Such neurons are sensitive to the temporal information present in the input data. By tuning the hyperparameters (threshold and leak factor), it is possible to identify input spikes that are close to each other in time. This phenomenon is demonstrated in Fig. 2. If the time between input spikes is large, the membrane potential decays its value before it can reach the threshold. However, if these inputs occur more frequently, they are able to overcome the decay and increase the membrane potential towards the threshold. Thus, the neuron generates output spikes if the input events occur at a frequency higher than a certain value. On the other hand, event cameras generate events when the light intensity acting on the pixels change with time. The rate at which these events are generated will depend on how fast the scene or objects are moving. Hence, a faster moving object will be responsible for generating events at a much higher frequency. When these properties of spiking neurons and event data are combined together, we have a network that is sensitive to objects that are moving faster than a certain speed.

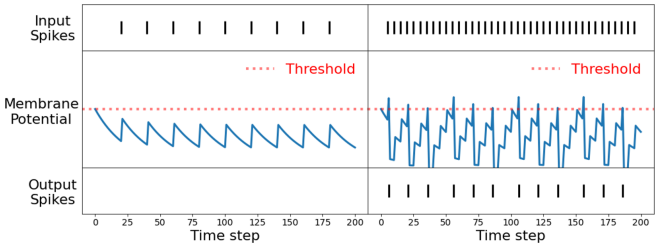
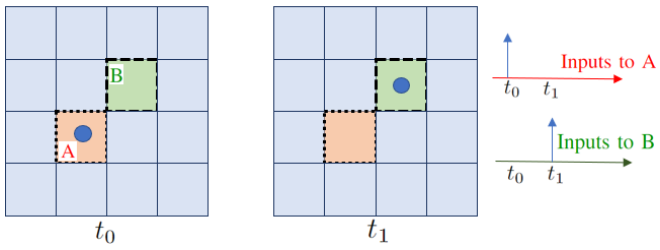


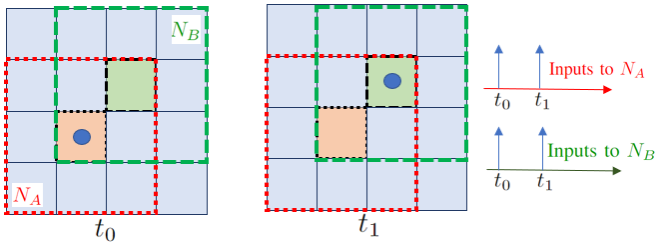
Fig. 2: Temporal sensitivity of a spiking neuron. The graphs on the left shows the neuron response to inputs occurring at a lower rate, the graphs on the right shows the same neuron respond to inputs occurring at a higher rate. Once the inputs occur at a rate higher than a certain frequency, the neuron is able to generate outputs.

While directly connecting each pixel of the event camera to a spiking neuron will measure the frequency of inputs at

that pixel, it does not necessarily measure the speed of an object. Let us consider the scenarios shown in Fig. 3. Here, an object generates an event at pixel A at time step t_0. If the object is moving at a fast speed, the event is generated in the neighboring pixel B at the next time step t_1. If we connect each pixel to a single spiking neuron, then both the neurons connected to pixels A and B only receive one input at each time step (Fig. 3a). Thus, the neural architecture cannot identify the fast-moving object. However, if we use a weighted sum from a neighborhood of pixels as inputs to neurons (Fig. 3b), then the neurons connected to neighborhoods N_A and N_B (centered around pixels A and B respectively) register two consecutive inputs. This leads to the spiking architecture successfully recognizing the object as fast moving. However, using a very large neighborhood would lead to events corresponding to different objects that are spatially apart, contribute to the same neuron, preventing object isolation. Hence, it is essential to choose an appropriate neighborhood size. We found that a neighborhood of (3×3) produced the best results.



(a) Each neuron is connected to a camera pixel directly.



(b) Each neuron is connected to a neighborhood of pixels.

Fig. 3: Two scenarios, with events generated identically at two consecutive time steps. In scenario (a), the events are registered as inputs only once at each neuron. In scenario (b), the events are registered in both time steps at both neurons.

Therefore, as demonstrated in Fig. 4, we consider a (3×3) weighted neighborhood of pixels as inputs to each neuron. The weights for the neighborhood were normalized to sum to 1. The central pixel was assigned a weight of 0.2, while the rest of the pixels were assigned equal weights of 0.1. This allowed our architecture to detect higher speed objects in a scene more efficiently.

Once we use the neural architecture to detect fast moving objects, we then need to isolate the input events that correspond to these objects before we separate them in the spatial domain. In order to do this, we recover the inputs around neighborhoods of the spike outputs. The size of this recovery neighborhood can be tuned and is not restricted to the size of

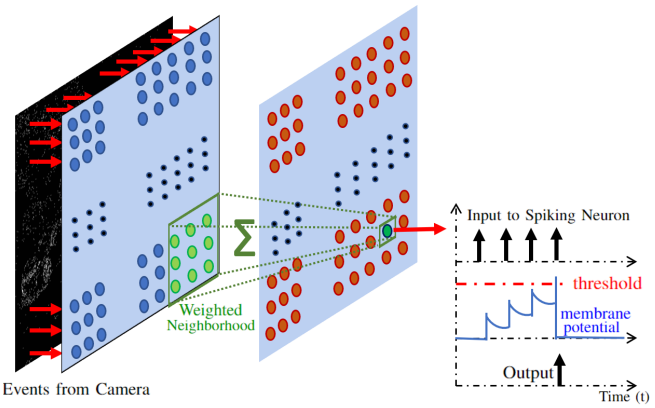
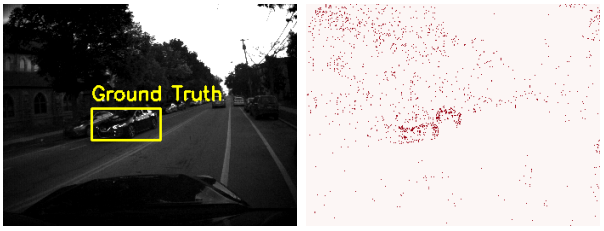
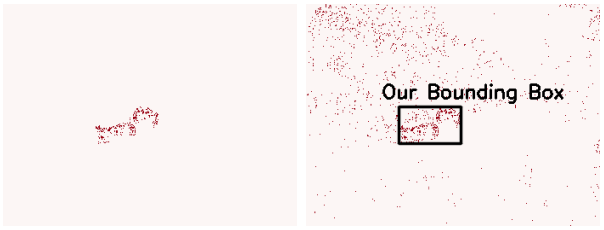


Fig. 4: Spiking architecture used to detect and isolate events belonging to objects moving at a faster speed. A weighted summation of inputs from pixels of the event camera are directly fed into the neurons. If the object is moving at a higher speed, it will prompt the neuron to generate a spiking output. This can be used as a detecting gate, that only propagates events from objects moving at a higher speed and filter out the remaining events.

the input neighborhood for neurons. Our spiking layer will act asynchronously with a delay of one time step (to check for the output spike and recover the inputs in the previous time step that caused this output). This method, unlike many existing works, does not require event data to be accumulated over a period of time before they can be used for processing. Fig. 5. demonstrates the operation of the proposed spiking architecture using a scene from the MVSEC dataset [8].



(a) Input grayscale image (b) Input events



(c) Isolated moving object (d) Resulting bounding box

Fig. 5: Demonstration of our algorithm on the MVSEC dataset. The events corresponding to the moving object are isolated and then grouped together based on spatial proximity.

B. Clustering in the Spatial Domain

In order to detect and draw accurate object boundaries, we use spatial clustering on the separated input events. Although there are many clustering techniques that are widely

available, most of them require us to either know the size of the clusters or the number of clusters in the data prior to clustering. Since we do not have any prior knowledge of the input scenario, we need a clustering technique that is independent of both these factors. For our experiments, we chose a density-based technique called Density Based Spatial Clustering (DBSC) [40]. The only parameters this technique requires are the minimum number of data points needed to form a cluster and the minimum data density of a cluster. Both of these parameters were fine tuned for the best results.

C. Extending to Multiple Speeds

So far, we have proposed a spiking neural block that can separate out objects moving from those that are stationary. While this is useful and already shows improvements at object detection in real world scenarios, we can further separate objects based on their speed. Fig. 6. shows the overview of the architecture that can be used to separate objects based on their speeds.

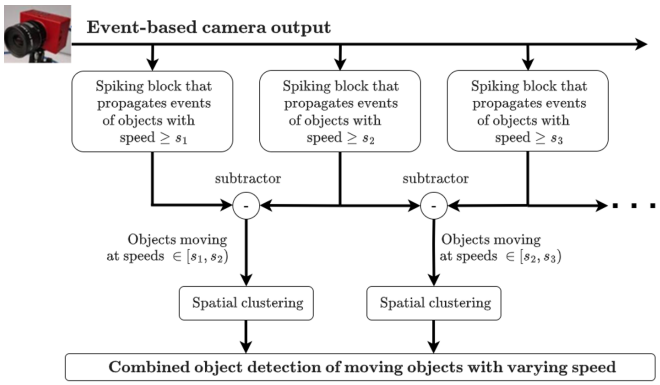


Fig. 6: Overview of the architecture to detect multiple objects moving at varying speeds. Each of the spiking blocks is fine-tuned to a specific range of speeds. Events corresponding to objects moving at a speed within this range are propagated. These input events can then be spatially clustered in order to detect object boundaries.

Each branch of this architecture will isolate objects moving faster than a certain speed. By considering the difference between the outputs of branches that have successive threshold speeds s_1 and s_2, we can isolate objects that are moving with a speed in the range $[s_1, s_2)$. Multiple such ranges could be added and fine-tuned before deployment, based on the application. These branches operate in parallel and will not add any latency. Since each branch only consists of a single layer of spiking neurons, the overall energy efficiency of the system is still preserved.

IV. Experiments and Results

In this section, we both qualitatively and quantitatively show the accuracy and energy efficiency of our proposed technique. Most existing works evaluate their algorithms on private datasets that are generated by the authors. These datasets are not available to the public and do not resemble event camera outputs that are generated in real life scenarios. In order to measure the accuracy and energy efficiency of our algorithm in real scenarios, we chose the MVSEC dataset [8] to show our results. This is a publicly available dataset and contains data collected by both an event camera, and a traditional frame camera mounted on a car.

A. Accuracy

As stated earlier, most existing works can only operate in scenarios without any background or camera noise. Our algorithm however can work well under these conditions. This is because, the spiking architecture acts as a temporal filter and can eliminate both noisy inputs from the camera as well as the events generated from the static objects in the background.

Since the MVSEC dataset does not contain ground truth bounding boxes of objects in the foreground, we used YOLOv3 [7], a traditional frame-based state of the art algorithm to generate bounding boxes on the corresponding grayscale image. We used this as the ground truth and then compared the *intersection over union* (IoU) of the objects present in the foreground (not static). Since the grayscale images are generated at a much lower rate, we only calculate IoUs of the event-based detection algorithms at these time frames. It is to be noted, however, that our detection algorithm is asynchronous and can operate at the rate events are generated by the event camera.

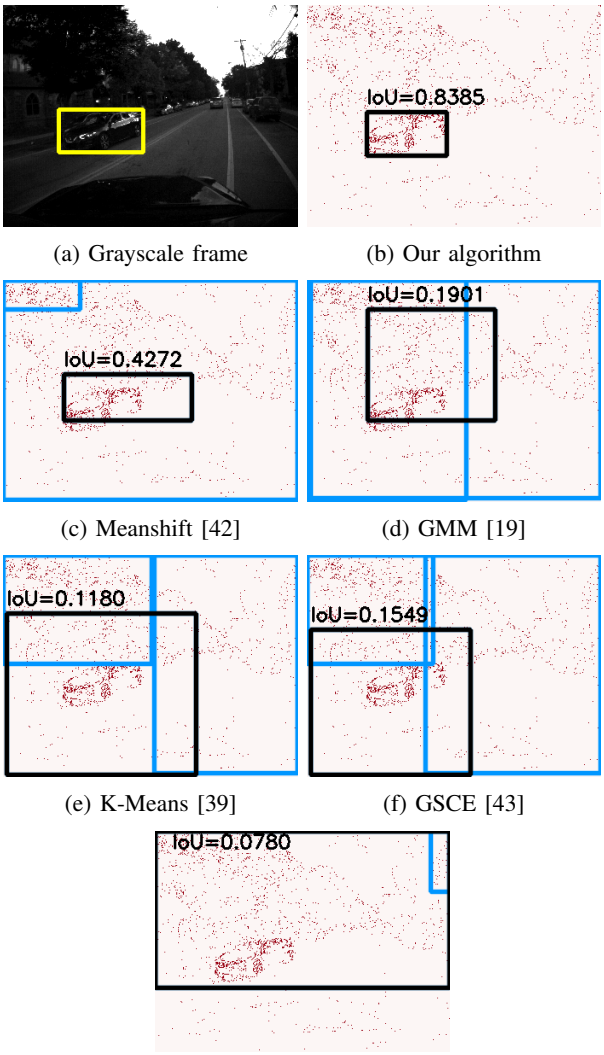
We consider a bounding box with an $IoU \geq 0.5$ with respect to the ground truth as a *true positive* (TP) and one with an $IoU < 0.5$ as a *false negative* (FN). If a bounding box was generated when no corresponding ground truth was detected, we considered this a *false positive* (FP). The *precision* and *recall* were calculated based on their standard definitions using TP, FN, and FP. Table I shows the comparison of these accuracy metrics of detection algorithms on the *outdoor day 2* segment of the MVSEC dataset.

TABLE I: Table showing the comparison of accuracy metrics of existing object detection algorithms to our algorithm. Experiments were conducted on the *outdoor day 2* segment of the MVSEC dataset.

Algorithm	Mean IoU	Recall	Precision
GMM [19]	0.2154	0.08	1.00
Meanshift [42]	0.3986	0.23	1.00
K-Means [39]	0.0997	0.00	0.00
GSCE [43]	0.1071	0.00	0.00
DBSCAN only [40]	0.1244	0.00	0.00
Ours (+ DBSCAN)	**0.8593**	**1.00**	**1.00**

Fig. 7. shows a more qualitative comparison of the bounding boxes predicted by each of these algorithms. We observe that utilizing clustering techniques ([39], [40]) directly on the event camera output fails to detect objects accurately. This is because, the events generated by the background and camera noise can occur spatially close to the events generated by the foreground (or moving objects). Other techniques such as [42] and [19] are able to minimize the number of false positives, but perform poorly producing a large number of

false negatives (boxes with $IoU < 0.5$). These algorithms also generate multiple bounding boxes even if there is only one object in the foreground. Our algorithm outperforms all the other existing algorithms at all instances of the dataset, and minimizes the number of false positives and false negatives while producing bounding boxes with a good IoU (≥ 0.5). For a fairer comparison, we consider the bounding box with the highest IoU with respect to the ground truth for computing the accuracy metrics in algorithms that generate multiple bounding boxes.



(a) Grayscale frame (b) Our algorithm

(c) Meanshift [42] (d) GMM [19]

(e) K-Means [39] (f) GSCE [43]

(g) DBSCAN only [40]

Fig. 7: Comparison of our object detection algorithm to existing algorithms. In case of algorithms that generate multiple bounding boxes, the box with the highest IoU with respect to the ground truth is considered. These boxes are highlighted in black, and the other bounding boxes are highlighted in blue[2].

B. Computational Efficiency

One of the main benefits of the proposed approach of object detection is the computational efficiency in terms of both energy and latency. Spiking neurons accumulate inputs in the membrane potential and only emit an output spike if the accumulated membrane potential exceeds a set threshold. These *accumulate* (AC) operations require only a fraction of the energy needed for computation when compared to the traditional *multiply and accumulate* (MAC) operations that are used in artificial neural networks. It has been shown in [12], that for 32-bit floating-point computations performed on a $45nm$ CMOS technology, the energy for AC operations (E_{AC}) is $0.9pJ$ and the energy for MAC operations (E_{MAC}) is $4.6pJ$. Further, due to the nature of events, these AC operations occur in a sparse fashion.

We can estimate the energy consumed for computation by the spiking layer by multiplying the number of such synaptic operations #OPS$_{\text{SNN}}$ with E_{AC}. Formally, the total computational energy is given as:

$$\text{E}_{\text{spiking}} = \#\text{OPS}_{\text{SNN}} \times E_{AC} = M \times C \times F \times E_{AC} \quad (2)$$

where M is the number of spiking neurons, C is the number of synaptic connections (inputs to each neuron) and F is the mean pre-spiking input rate. By plugging in the appropriate values ($M = 260 \times 346, C = 3 \times 3, F = 0.0151$), we estimate our architecture consumes an average energy of $11.03nJ$, during inference on the MVSEC *outdoor day 2* segment. This is a huge reduction in energy consumption compared to average energy of $44.06mJ$ that a standard artificial neural network YOLOv3 [7] uses during inference on the same dataset. We estimated the latter value by multiplying the number of MAC operations in the entire network ($\sum_{layer} M_{layer} \times C_{layer} = 9.58 \times 10^9$) with the energy consumption of a single MAC operation.

Our proposed technique is also efficient in terms of latency. While traditional computer vision works such as [7] are limited by the input rate of frame cameras, our algorithm can operate at much faster rates owing to the high output rate of event cameras. We also do not need to accumulate events for a duration of time before processing, unlike many event-based detection algorithms [20], [23], [24]. This true asynchronous nature of the algorithm minimizes our processing overhead in terms of latency.

V. Conclusion

In this work, we present a novel object detection algorithm for autonomous navigation tasks using biologically inspired spiking neurons in combination with an event camera. By using the inherent temporal information present in the input data, the proposed spiking architecture isolates events based on the speed of objects. We show our algorithm excels at detecting object boundaries, even in scenarios with camera noise and background objects, where other existing event-based algorithms fail. The proposed methodology carries low energy overhead, owing to the sparsity of input data, low network complexity and the computationally efficient accumulate (as opposed to multiply and accumulate in standard deep learning) operations of spiking neurons. In addition, the spiking architecture developed is independent of the scene of deployment, and is scalable to multiple speeds.

[2]Note that when the algorithms fail to isolate the moving object (7c)-(7g), there could be bounding boxes that cover the entire frame.

REFERENCES

[1] S. S. Shadrin and A. A. Ivanova, "Analytical review of standard SAE J3016 taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles with latest updates," *Avtomobil'. Doroga. Infrastruktura.*, no. 3 (21), p. 10, 2019.

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

[3] P. Lichtsteiner, C. Posch, and T. Delbruck, "A 128 × 128 120 db 15μs latency asynchronous temporal contrast vision sensor," *IEEE journal of solid-state circuits*, vol. 43, no. 2, pp. 566–576, 2008.

[4] C. Posch, D. Matolin, and R. Wohlgenannt, "A qvga 143 db dynamic range frame-free pwm image sensor with lossless pixel-level video compression and time-domain cds," *IEEE Journal of Solid-State Circuits*, vol. 46, no. 1, pp. 259–275, 2010.

[5] C. Brandli, R. Berner, M. Yang, S.-C. Liu, and T. Delbruck, "A 240× 180 130 db 3 μs latency global shutter spatiotemporal vision sensor," *IEEE Journal of Solid-State Circuits*, vol. 49, no. 10, pp. 2333–2341, 2014.

[6] J. Canny, "A computational approach to edge detection," *IEEE Transactions on pattern analysis and machine intelligence*, no. 6, pp. 679–698, 1986.

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

[8] A. Zihao Zhu, D. Thakur, T. Ozaslan, B. Pfrommer, V. Kumar, and K. Daniilidis, "The multi vehicle stereo event camera dataset: An event camera dataset for 3d perception," *arXiv e-prints*, pp. arXiv–1801, 2018.

[9] A. Delorme, J. Gautrais, R. Van Rullen, and S. Thorpe, "Spikenet: A simulator for modeling large networks of integrate and fire neurons," *Neurocomputing*, vol. 26, pp. 989–996, 1999.

[10] B. Han and K. Roy, "Deep spiking neural network: Energy efficiency through time based coding," in *European Conference on Computer Vision*. Springer, 2020, pp. 388–404.

[11] A. S. Kucik and G. Meoni, "Investigating spiking neural networks for energy-efficient on-board ai applications. a case study in land cover and land use classification," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 2020–2030.

[12] M. Horowitz, "1.1 computing's energy problem (and what we can do about it)," in *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*. IEEE, 2014, pp. 10–14.

[13] C. Harris, M. Stephens, *et al.*, "A combined corner and edge detector," in *Alvey vision conference*, vol. 15, no. 50. Citeseer, 1988, pp. 10–5244.

[14] S. Xie and Z. Tu, "Holistically-nested edge detection," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1395–1403.

[15] T. Delbruck and P. Lichtsteiner, "Fast sensory motor control based on event-based hybrid neuromorphic-procedural system," in *2007 IEEE international symposium on circuits and systems*. IEEE, 2007, pp. 845–848.

[16] T. Delbruck and M. Lang, "Robotic goalie with 3 ms reaction time at 4% cpu load using event-based dynamic vision sensor," *Frontiers in neuroscience*, vol. 7, p. 223, 2013.

[17] X. Lagorce, C. Meyer, S.-H. Ieng, D. Filliat, and R. Benosman, "Asynchronous event-based multikernel algorithm for high-speed visual features tracking," *IEEE transactions on neural networks and learning systems*, vol. 26, no. 8, pp. 1710–1720, 2014.

[18] X. Clady, S.-H. Ieng, and R. Benosman, "Asynchronous event-based corner detection and matching," *Neural Networks*, vol. 66, pp. 91–106, 2015.

[19] E. Pikatkowska, A. N. Belbachir, S. Schraml, and M. Gelautz, "Spatiotemporal multiple persons tracking using dynamic vision sensor," in *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*. IEEE, 2012, pp. 35–40.

[20] A. Mitrokhin, C. Fermuller, C. Parameshwara, and Y. Aloimonos, "Event-based moving object detection and tracking," *arXiv preprint arXiv:1803.04523*, 2018.

[21] D. Tedaldi, G. Gallego, E. Mueggler, and D. Scaramuzza, "Feature detection and tracking with the dynamic and active-pixel vision sensor (davis)," in *2016 Second International Conference on Event-based Control, Communication, and Signal Processing (EBCCSP)*. IEEE, 2016, pp. 1–7.

[22] D. Gehrig, H. Rebecq, G. Gallego, and D. Scaramuzza, "Eklt: Asynchronous photometric feature tracking using events and frames," *International Journal of Computer Vision*, vol. 128, no. 3, pp. 601–618, 2020.

[23] V. Vasco, A. Glover, and C. Bartolozzi, "Fast event-based harris corner detection exploiting the advantages of event-driven cameras," in *2016 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2016, pp. 4144–4149.

[24] E. Mueggler, C. Bartolozzi, and D. Scaramuzza, "Fast event-based corner detection," *British Machine Vision Conference (BMVC)*, 2017.

[25] X. Clady, J.-M. Maro, S. Barré, and R. B. Benosman, "A motion-based feature for event-based pattern recognition," *Frontiers in neuroscience*, vol. 10, p. 594, 2017.

[26] B. Ramesh, H. Yang, G. Orchard, N. A. Le Thi, S. Zhang, and C. Xiang, "Dart: distribution aware retinal transform for event-based cameras," *IEEE transactions on pattern analysis and machine intelligence*, vol. 42, no. 11, pp. 2767–2780, 2019.

[27] A. Sironi, M. Brambilla, N. Bourdis, X. Lagorce, and R. Benosman, "Hats: Histograms of averaged time surfaces for robust event-based object classification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 1731–1740.

[28] J. Li, F. Shi, W.-H. Liu, D. Zou, Q. Wang, P. K. Park, and H. Ryu, "Adaptive temporal pooling for object detection using dynamic vision sensor." in *BMVC*, 2017.

[29] M. Cannici, M. Ciccone, A. Romanoni, and M. Matteucci, "Asynchronous convolutional networks for object detection in neuromorphic cameras," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2019, pp. 0–0.

[30] O. Bichler, D. Querlioz, S. J. Thorpe, J.-P. Bourgoin, and C. Gamrat, "Extraction of temporally correlated features from dynamic vision sensors with spike-timing-dependent plasticity," *Neural networks*, vol. 32, pp. 339–348, 2012.

[31] P. O'Connor, D. Neil, S.-C. Liu, T. Delbruck, and M. Pfeiffer, "Real-time classification and sensor fusion with a spiking deep belief network," *Frontiers in neuroscience*, vol. 7, p. 178, 2013.

[32] D. Marti, M. Rigotti, M. Seok, and S. Fusi, "Energy-efficient neuromorphic classifiers," *Neural computation*, vol. 28, no. 10, pp. 2011–2044, 2016.

[33] Y. Cao, Y. Chen, and D. Khosla, "Spiking deep convolutional neural networks for energy-efficient object recognition," *International Journal of Computer Vision*, vol. 113, no. 1, pp. 54–66, 2015.

[34] G. Orchard, C. Meyer, R. Etienne-Cummings, C. Posch, N. Thakor, and R. Benosman, "Hfirst: A temporal approach to object recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 37, no. 10, pp. 2028–2040, 2015.

[35] S. Kim, S. Park, B. Na, and S. Yoon, "Spiking-yolo: spiking neural network for energy-efficient object detection," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, no. 07, 2020.

[36] P. Machado, A. Oikonomou, J. F. Ferreira, and T. M. Mcginnity, "Hsmd: An object motion detection algorithm using a hybrid spiking neural network architecture," *IEEE Access*, vol. 9, pp. 125 258–125 268, 2021.

[37] N. Kasabov, K. Dhoble, N. Nuntalid, and G. Indiveri, "Dynamic evolving spiking neural networks for on-line spatio-and spectro-temporal pattern recognition," *Neural Networks*, vol. 41, pp. 188–201, 2013.

[38] Z. Jiang, Z. Bing, K. Huang, and A. Knoll, "Retina-based pipe-like object tracking implemented through spiking neural network on a snake robot," *Frontiers in neurorobotics*, vol. 13, p. 29, 2019.

[39] J. A. Hartigan and M. A. Wong, "Algorithm as 136: A k-means clustering algorithm," *Journal of the royal statistical society. series c (applied statistics)*, vol. 28, no. 1, pp. 100–108, 1979.

[40] L. Duan, L. Xu, F. Guo, J. Lee, and B. Yan, "A local-density based spatial clustering algorithm with noise," *Information systems*, vol. 32, no. 7, pp. 978–986, 2007.

[41] K. G. Derpanis, "Mean shift clustering," *Lecture Notes*, vol. 32, 2005.

[42] G. Chen, H. Cao, M. Aafaque, J. Chen, C. Ye, F. Röhrbein, J. Conradt, K. Chen, Z. Bing, X. Liu, *et al.*, "Neuromorphic vision based multivehicle detection and tracking for intelligent transportation system," *Journal of advanced transportation*, vol. 2018, 2018.

[43] A. Mondal, J. H. Giraldo, T. Bouwmans, A. S. Chowdhury, *et al.*, "Moving object detection for event-based vision using graph spectral clustering," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 876–884.