SPi	Journal Code	Article ID	Dispatch: 15-FEB-19	CE: D, Subramani
	WIDM	1310	No. of Pages: 34	ME:

Received: 6 September 2018 | Revised: 23 January 2019 | Accepted: 24 January 2019

DOI: 10.1002/widm.1310

WILEY WIREs DATA MINING AND KNOWLEDGE DISCOVERY

ADVANCED REVIEW

Neuromorphic vision: Sensors to event-based algorithms

A. Lakshmi[1,2] | Anirban Chakraborty[3] | Chetan S. Thakur[2]

[1]Centre for Artificial Intelligence and Robotics, Defence Research and Development Organization, Bangalore, India

[2]Department of Electronic Systems Engineering, Indian Institute of Science, Bangalore, India

[3]Department of Computational and Data Sciences, Indian Institute of Science, Bangalore, India

Correspondence
Chetan Singh Thakur, Department of Electronic Systems Engineering, Indian Institute of Science, Bangalore 560012, India.
Email: csthakur@iisc.ac.in

Regardless of the marvels brought by the conventional frame-based cameras, they have significant drawbacks due to their redundancy in data and temporal latency. This causes problem in applications where low-latency transmission and high-speed processing are mandatory. Proceeding on this line of thought, the neurobiological principles of the biological retina have been adapted to accomplish data sparsity and high dynamic range at the pixel level. These bio-inspired neuromorphic vision sensors alleviate the more serious bottleneck of data redundancy by responding to *changes in illumination* rather than to illumination itself. This paper reviews in brief one such representative of neuromorphic sensors, the activity-driven event-based vision sensor, which mimics human eyes. Spatio-temporal encoding of event data permits incorporation of time correlation in addition to spatial correlation in vision processing, which enables more robustness. Henceforth, the conventional vision algorithms have to be reformulated to adapt to this new generation vision sensor data. It involves design of algorithms for sparse, asynchronous, and accurately timed information. Theories and new researches have begun emerging recently in the domain of event-based vision. The necessity to compile the vision research carried out in this sensor domain has turned out to be considerably more essential. Towards this, this paper reviews the state-of-the-art event-based vision algorithms by categorizing them into three major vision applications, object detection/recognition, object tracking, localization and mapping.

This article is categorized under:
Technologies > Machine Learning

1 | INTRODUCTION

In this era of rapid innovations and technological marvels, much effort is being put into developing visual sensors that aim to impersonate the working principles of the human retina. These efforts are motivated by and driven towards the ultimate objective of catering for applications such as high-speed robotics and parsing/analyzing high dynamic range scenes. There are situations where the conventional video cameras fall short. The colossal amount of data from a conventional camera overwhelms the processing of embedded systems and henceforth unavoidably turns into a bottleneck. This necessitates that some fundamental level of preprocessing has to be done at the acquisition level, as opposed to loading processing systems with redundant data. In any case, developing sensors for this novel modality requires information-processing technologies built from the domain of neuromorphic engineering. One such research effort prompted the development of neuromorphic vision sensors, also known as event-based vision sensors, which have shown incredible accomplishment in furnishing solutions to the issues encountered in conventional cameras. These sensors have led to a revolutionary way of recording information by asynchronously registering only changes in illumination with microsecond accuracy.

These sensors find utility extensively in high-speed vision-based applications, which are indispensable in various fields such as micro-robotics. The frame-based vision algorithms would not be effective in analyzing these event data-streams. As

they expect synchronous data, these would not enable us to leverage upon the aforementioned benefits obtainable from neuromorphic vision sensor data. Consequently, a shift in the paradigm of algorithms from conventional frame-based to event-based sensors is essential. The development of event-based vision algorithms is yet to reach the scale of development of their frame-based counterparts as well as the level of accuracy these algorithms have been able to achieve in the conventional video data domain. The main reason is the very limited commercial availability of these sensors and that these are accessible only as prototypes or proof-of-concept. Nevertheless, quite a number of interesting computer vision algorithms are recently being developed for these type of sensors, which are optimally designed to exploit the higher temporal resolution and data sparsity of these sensors.

With the development of present-day technologies, these sensors are starting to show up in the commercial field. This also necessitates a structured and detailed study of important vision algorithms already conceived or needed to be developed for analyzing the event data streams. However, there is hardly any such survey in the event domain unlike the plethora of similar works existing for its frame-based counterparts. Hence, we present a survey of vision algorithms available for event data. Despite the fact that our focus is to review event-based vision algorithms, we have additionally intended to provide a concise overview of silicon retinae, with more emphasis on a specific type of silicon retina known as the temporal difference silicon retina. Furthermore, we also briefly review the event datasets available in the public domain to evaluate the aforementioned event-based vision algorithms.

The structure of the paper is as follows. The paper introduces silicon retina and temporal difference silicon retina in a nutshell, followed by the details of the state-of-the-art of three important event-based vision applications, viz., object detection and recognition, object tracking as well as localization and mapping. As a closing remark, the paper gives a gist of open-source event datasets along with the available codes[1] and implementations of event-based vision sensor algorithms.

2 | SILICON RETINA

Information processing of neurobiological systems is very exceptionally one-of-a-kind in relation to present-day image-capturing devices. Biological vision systems dependably beat human-invented vision sensors. Conventional image sensors are operated by a synchronous clock signal, which has resulted in an enormous amount of monotonous and redundant data accumulation. This aspect has motivated the research in neuromorphic engineering to emulate vision sensors that mimic human eyes, which has led to the development of silicon retinae. Silicon retinae are event-based, unlike conventional sensors, which are frame-based. This makes it more similar to the biological model of the human eye, which is driven by events occurring in the real world. The first electronic model of the retina came into existence in the early 70's (Fukushima, Yamaguchi, Yasuda, & Nagata, 1970; Mahowald, 1994), following which numerous forms of silicon retinae widely known as event-based vision sensors came into existence, including the spatial contrast retina that encodes relative light intensity change with respect to the background (Barbaro, Burgi, Mortara, Nussbaum, & Heitger, 2002; Bardallo, Gotarredona, & Barranco, 2009; Ruedi et al., 2009), gradient-based sensors that encode static edges, temporal intensity retina (Mallik, Clapp, Choi, Cauwenberghs, & Cummings, 2005), and temporal difference retina (Kramer, 2002; Lichtsteiner, Posch, & Delbruck, 2006). This study reviews the rationale of temporal difference silicon retinae, which are commonly known as activity-driven event-based vision sensors.

Temporal difference silicon retina generates events in response to changes in illumination in the scene in temporal space. The following section discusses, in brief, the details of their architecture, types, history, advantages, companies involved, and software available.

2.1 | Event generation mechanism

In a conventional sensor, an image consists of pixels $I_{x,y}(t)$, whose value is sampled at fixed time intervals Δt. This results in a representation that is incompatible with compact representation. In temporal difference silicon retina, the pixels with significant luminance changes across time, $I'_{x,y}(t)$, are retained. An event at spatial location (x, y) and time t can be represented as

$$e_t = \begin{cases} sign\left\{I'_{x,y}(t)\right\} & if \mid I'_{x,y}(t) \mid > \triangle \\ 0 & if \mid I'_{x,y}(t) \mid < \triangle \end{cases}, \quad (1)$$

where $\triangle$ is the threshold. These events are spatio-temporal in nature.

2.2 | Event transmission mechanism

As temporal difference silicon retina is spike based, it utilizes a spike-based interfacing procedure known as address event representation (AER: Figure 1) to transmit asynchronous information over a narrow channel (Gotarredona, Andreou, & Barranco, 1999). In AER, each neuron is assigned a unique address by its encoder/arbiter. When an event is generated, the address of the neuron and relevant information are sent to the destination over the bus. The decoder in the receiver interprets the received information and assigns the received information to the particular destination. When the number of event-generating neurons surpasses the transmission capacity of the bus, multiplexing is employed.

2.3 | Event processing mechanism

This section describes the way in which spatio-temporal data of silicon retina could be interpreted. The temporal nature of these events is generally modeled with weights that are binary or an exponential decay function. Events modeled with binary weights can be represented with an activity buffer using built-in data structure support, first-in-first-out (FIFO). If a pixel's event exists for a longer duration than a predefined threshold $\bigtriangledown\ t$, it is discarded. As this threshold impacts the bandwidth required for transmission, it must be picked with utmost care in view of the application requirement. In the case of decay function model, the event that occurred at time t_i is made to lose its weight $\left(\exp^{\frac{t-t_i}{\tau}}\right)$ exponentially with time t, the decay being characterized by a time constant τ. The event is retained only if its weighted value surpasses a predefined threshold ϵ.

2.4 | Brief survey of temporal difference silicon retina

This section puts forth a brief survey of research and development of temporal difference silicon retina.

In 2002, Jorg Kramer developed an array of 48 × 48 pixels, temporal difference silicon retina (Kramer, 2002). Events were generated in this device when the intensity of the pixels exceeded a predefined threshold. Therefore, this device was unable to capture slowly moving objects. Lichtsteiner, Posch, and Delbruck (2008) presented a dynamic vision sensor 128 (DVS128) with the design of photodiodes that were superior to that of Kramer's. In 2010, Bardallo et al. (2009) introduced a vision sensor with a very low latency of 3.6 μs, which empowered it to capture objects rotating at a speed of 10,000 revolutions per second. In spite of the fact that it exhibits a reduction in the pixel area, the manufacturing technology might make this sensor to occupy more area than that of DVS128. In 2010, asynchronous time-based image sensor (ATIS) was proposed by Posch, Matolin, and Wohlgenannt (2011), which combined two sensors for each pixel. This enabled it to transmit new intensity values only after a change has happened, thereby making real-time data compression possible.

Increase in the practical application of DVS has driven many companies to get involved in research and development of the sensor. Variants of DVS such as DVS128 (Lichtsteiner et al., 2008), eDVS, mini-eDVS, DAVIS240 (240 × 180) (Brandli, Berner, Yang, Liu, & Delbruck, 2014a, 2014b), DAVIS346 (346 × 260) and color-DAVIS (Li et al., 2015) have been developed by iniVation Inc. Samsung Inc. developed DVS (Son, Suh, et al., 2017) with lesser pixel size, low power, good event quality, and low data rate. The data to be sent can be programmed. The sensor sends events when the programmed feature appears strongly in the field of view. The Australian Institute of Technology (AIT) has developed a dual line sensor, which is an optical CMOS dynamic vision sensor with two lines of 256 pixels each. It displays extreme high time resolution and data reduction as compared to conventional line sensors. In addition to this, it has also come up with an ATIS sensor as proposed by Posch et al. (2011). The other companies that develop event-based vision sensors are Hillhouse technology Inc., Insightness Inc., and Prophesee Inc. Some event-based vision sensors that are not commercially available, but whose prototypes have been developed are LWIR DVS (Posch, Matolin, Wohlgenannt, Maier, & Litzenberger, 2009), Smart DVS (Posch, Hoffstaetter, & Schoen, 2010), Sensitive DVS (Bardallo, Gotarredona, & Barranco, 2011; Gotarredona & Barranco, 2013) etc.

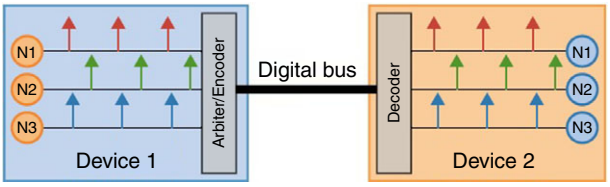

FIGURE 1 Inter device communication using AER. Figure courtesy (Gotarredona, Andreou, & Linarese, 1999)

2.5 | Architecture

DVS128 and ATIS are the two most prominent temporal difference silicon retinae that revolutionized the field. Hence, the architecture of these two sensors is discussed in more detail in subsequent sections.

2.5.1 | DVS128 architecture

DVS has 128 × 128 array of independent pixels. Each pixel responds independently and asynchronously to logarithmic change in light intensity with an ON spike to indicate the increase in intensity and an OFF spike to demonstrate the decrease in intensity. DVS communicates using AER circuits, with unique addresses for ON and OFF spikes. A simplified diagram of a single pixel of DVS is shown in Figure 2. The first stage comprises a logarithmic photo-detector, which emits a voltage proportional to the logarithm of the photo-current sensed by the circuit. This logarithmic response aids DVS to accomplish a good dynamic range. The photoreceptor is followed by a differencing amplifier that amplifies the change in the output of the first stage (V_p). The output of the differencing amplifier, V_{diff}, is fed to two comparators, which possess ON and OFF thresholds and generate an ON or OFF event, respectively, based on the sign of V_{diff}.

DVS has evolved into several versions such as in Brandli et al. (2014a, 2014b), wherein authors developed a Dynamic and Active Pixel Vision Sensor (DAVIS), which utilizes Active Pixel Sensor (APS) capability to enable overall intensity readout and Delbruck and Berner (2010), Bardallo et al. (2011), Yang, Liu, & Delbruck, (2015), where research has been carried out to increase the contrast sensitivity.

2.5.2 | ATIS architecture

Temporal contrast vision sensor has progressed to the latest sensor known as Asynchronous Time-based Image Sensor (ATIS) (Posch et al., 2011), which encodes the absolute change in illumination rather than +1/ −1 when activated by event occurrence. ATIS has a Quarter Video Graphics Array (QVGA) resolution of 304 × 240 asynchronous sensors. Each sensor comprises an illuminance change detector circuit and a conditional exposure measurement circuit. Figure 3 demonstrates the functional diagram of a single pixel. In response to illumination change of certain predefined threshold, the change detector circuit triggers the exposure measurement device. Pixel-specific change detector operation yields temporal redundancy suppression. The exposure measurement device encodes the change in illumination into the inter-spike timing of pulses. When triggered, the exposure measurement circuit begins to integrate its photo-current, emitting a spike when it reaches a lower

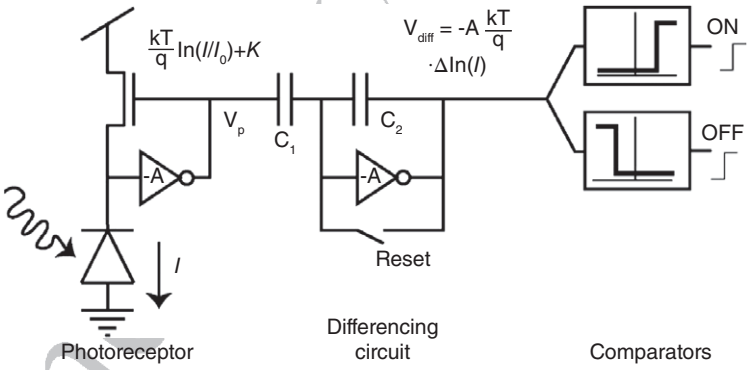

FIGURE 2 Architecture of a pixel of DVS. Figure courtesy (Lichtsteiner et al., 2008)

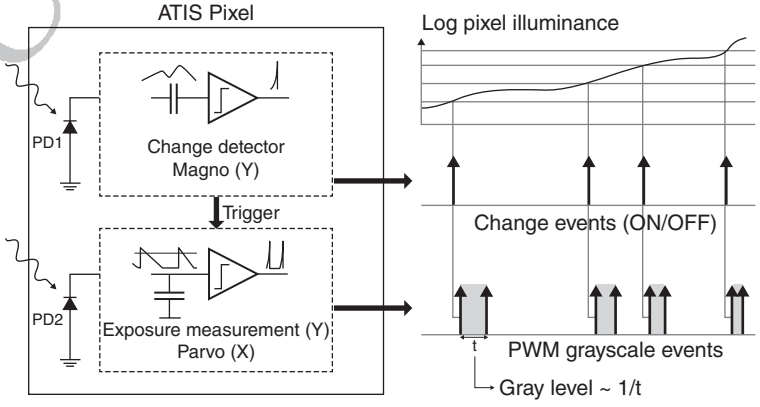

FIGURE 3 Single pixel architecture of ATIS. Figure courtesy (Posch et al., 2011)

threshold, trailed by the emission of a second spike once it reaches the upper threshold. Thus, the inter-spike time ends up being proportional to the pixel intensity.

The type of information encoding employed in the ATIS refrains from imposing the same integration time for the entire pixel array. This results in prominent benefit in terms of attaining high dynamic range. The ATIS can achieve an intra-scene dynamic range of 143 dB as the integration time could be varied from 350 ns to 4.5 s. Generally, the temporal resolution is traded off for a good dynamic range. However, the ATIS implements techniques used in standard voltage-mode imagers to sustain higher dynamic range without compromising on temporal resolution. The ATIS communicates using AER protocol, and the frames could be generated as and when needed by buffering the events that occurred over a given period of time.

2.6 | Advantages

Traditional vision cameras come with certain setbacks such as limited frame rate, limited dynamic range, and redundant data. Neuromorphic vision sensors (Figure 4 shows the output of an conventional camera and an event camera) try to address the above-mentioned disadvantages as given in the forthcoming sections.

2.6.1 | Frame rate

A frame camera continuously outputs frames even if there is no visible change in the field of view. An event camera, on the other hand, reports only the changes in the scene asynchronously. A frame-based camera recapitulates over all pixels every time, which increases its latency to changes in input. In an event-based camera, pixels respond to scene intensity changes individually, which leads to a very low latency often as low as 15 μs. Hence, the effective frame rate of an event camera is of the order of several kiloHertz.

2.6.2 | Dynamic range

Dynamic range indicates the property of camera that measures its ability to acquire images in low and bright light scenarios. Frame-based cameras typically achieve a dynamic range of 60 *dB*, whereas the DVS and ATIS have a dynamic range of 120 and 143 *dB*, respectively, as each pixel of an event-based camera is independent. Hence, event-based cameras find use extensively in scenes with extreme intensity contrast.

2.6.3 | Bandwidth

Frame-based cameras transmit information of all pixels irrespective of their information content. Event-based cameras achieve data rate compression at pixel level as they are triggered asynchronously in response to a local scene intensity changes.

2.6.4 | Power consumption

Power consumption tends to be a critical issue in portable digital devices especially in applications such as robotics. Despite the fact that event- and frame-based cameras require power in the range of *mW*, event-based processing consumes zero power in static scenes. This makes it superior to a frame-based camera as far as power consumption is concerned.

2.7 | Software

Various drivers/application softwares are available for the DVS, few of which are jAER, caer, and libcaer. This section provides a brief overview of these frameworks.

jAER (Dynamic Vision Software, 2017) stands for Java Address-Event Representation, which is an open-source framework with programs to view real-time or recorded event data and to develop real-time event-based algorithms. It has a built-in user interface known as jAERViewer, which allows users to plug-in any AER device through a USB interface and

FIGURE 4 Left: image from conventional camera, right: event image from event-based vision camera. Figure courtesy (Dynamic Vision Object Recog and Action Recog and Tracking Simulated Dataset, 2016)

subsequently view or log the event data. In addition to these basic operations, it also supports computer vision operations such as tracking and low-level feature extraction. jAER is compatible with Windows, Linux, and MacOS.

In contrast to jAER, caer (Dynamic Vision Software, 2018a) is a software framework written in C/C++, mainly targeted at embedded systems. This makes it ideal for research applications. It does not require a GUI, unlike jAER. It has a wide range of advantages such as small memory footprint, limited usage of CPU, no dependency on the graphical user interface, remote configurability and network accessibility. It is also compatible with Linux, Windows and MacOS operating systems.

Libcaer (Dynamic Vision Software, 2018b) is a small framework developed by the company iniVation in C to provide capabilities to access, configure, send and receive data from sensors such as DVS, DAVIS. It consists of minimum API for defining event formats and device control. It supports Linux and MacOS. A ROS package named as the ROS DVS package (Dynamic Vision Software, 2018c) has been developed on libcaer, which provides C++ drivers for DVS and DAVIS, APIs to read recorded data and APIs for intrinsic and stereo calibration.

Yet Another Robotics Program (YARP) (Dynamic Vision Software, 2018d) provides libraries to handle the DVS sensor along with few computer vision functionalities such as filters to remove salt and pepper noise, optical flow estimation, cluster tracking, corner detection, circle detection, particle filter, etc. Details of the applications developed on YARP can be found in Dynamic Vision Software (2018d).

3 | EVENT-BASED VISION ALGORITHMS

The event-based sensor algorithms have pulled in a great deal of attention in computer vision. For drone-like applications, power efficiency is a noteworthy prerequisite, which forms a characteristic preferred standpoint of event-based vision sensors. As these sensors encode illumination changes asynchronously, they exhibit a higher temporal resolution and higher sparsity of data. This makes them a natural choice for video analysis algorithms, which require examination of fast-moving scenes with low latency. The sparsity of the data recorded also paves way for algorithms that can provide real-time performance.

As the event cameras were not accessible economically until recently, video analysis algorithms are limited for these sensors. As the data structure is significantly different from that of conventional cameras, this call for a new set of algorithms to unlock their full potential. In this section, our aim is to provide a minimal survey of state-of-the-art object detection/recognition, tracking and localization and mapping algorithms for event data.

3.1 | Object detection/recognition

Object detection/recognition finds its application in drones, autonomous driving, and so on. It has encountered a high boost with the colossal development of deep learning algorithms. Incredible success has been accomplished in object recognition in frame-based cameras. However, object recognition in event-based cameras still remains an open area of research. As noninformative background data is not recorded in event-based sensors, the images (Figure 5) of nonstationary objects extracted from the event data represent salient regions, thus enhancing the accuracy of object recognition. The accompanying section brings out recent works on event-based object recognition. The methods discussed in this paper do not implement any techniques that compensate the background or camera motion, though there are few works available to compensate for ego-motion. The different methods that have matured over these years for event data object detection can be broadly categorized as follows

1. Feature based: Most of the prior works concentrate on detecting event based features followed by classification.

FIGURE 5 Moving object recorded by an ATIS camera

2. Artificial Neural Network based: Artificial Neural Network (ANN) based approaches have not been explored much for the event based data until recently as a result of the following two note-worthy reasons: (i) the output of the event based camera is different from that of the conventional camera and consequently cannot be fed directly into the successive frame based networks and (ii) lack of labeled training data. Recently datasets have started to appear for event cameras, few of which are N-MNIST, N-Caltech101, MNIST-DVS, CIFAR10-DVS, N-CARS dataset. Few of the datasets were obtained by recording the corresponding RGB dataset images using an event camera (the details of the dataset are discussed at the end of the paper). This led to the progress of research in this direction as well. The ANN for object recognition on event data falls in any one of the methodology: (i) Frame based Convolutional Neural Network (CNN) applied to event data with little or without any modification, (ii) CNN adapted to Spiking Neural Network (SNN), (iii) Spiking Neural Network (SNN) without back-propagation and (iv) SNN with back-propagation
3. Time Surface based: SNN is the best architecture that suits the asynchronous event data. The significant disadvantage that comes with SNN is their inability to be trained due to nondifferentiability. In order to mitigate this demerit of SNN, researchers have come up very recently with time surface based features for object recognition.

3.1.1 | Conventional feature based object detection/recognition

Engineering the most suitable features for object recognition has been a well-established area of research for data obtained using the conventional frame-based camera. Following the same footsteps, object recognition on event data can be achieved by feature detection, followed by classification. There are a handful of feature detection algorithms for event data. The most widely adopted features are corner features, edges and lines. This section describes the works that detect corners from event data, where the corners detected were not further explored for the application of object recognition. However, features could be extracted around detected corners and fed into a classifier for further classification.

Event-based corner detection was introduced in Clady, Ieng, and Benosman (2015), where optical flow orientation in the local neighborhood has been utilized to detect corners. However, the performance of the corner detector relies upon the accuracy in the estimation of optical flow. As flow estimation is a costly operation, the speed of corner detector is constrained by the speed of optical flow estimation.

The work that followed was Vasco, Glover, and Bartolozzi (2016), where frame based Harris corner detector (Harris & Stephens, 1988) has been adapted to event data. As DVS data does not provide intensity measurements, a local binarized surface (L pixels wide) was generated around the given event. The pixel position was either populated as a 1 or 0 depending on whether an event has occurred at that particular location in a fixed temporal window. As followed in conventional Harris corner detector, a symmetric matrix was constructed with the gradients computed on the binary surface. The probable presence of edges was indicated by the occurrence of two high eigenvalues of the estimated symmetric matrix. Despite the fact that this methodology exhibited good performance, it is not computationally efficient as it involves gradient computation.

The computational complexity of the previous methods was overcome in Mueggler, Bartolozzi, and Scaramuzza (2017), where a corner detection method inspired by conventional FAST corner detector (Rosten & Drummond, 2006) has been proposed. The method operated on Surface of Active Events (SAE) (Benosman, Clercq, Lagorce, Ieng, & Bartolozzi, 2014). Traditional FAST detector declares a pixel as a corner if a predetermined number of contiguous pixels around the pixel under consideration are darker or brighter than the latter. As SAE represents time stamps and not intensity values, the distribution of time stamp in a circular set of locations was considered for corner detection. If the set with newest time stamp was distributed as contiguous circular arc within a predefined angular range, the pixel under consideration was classified as a corner. This strategy is computationally more efficient than traditional Harris detector for a small trade-off in accuracy.

Inspired by the event-based FAST corner detector, (Alzugaray & Chli, 2018) proposed a corner detector algorithm which was 4× times faster than FAST corner detector and 50× faster than event-based Harris detector. It extended the native event FAST corner detector by employing an efficient iterative algorithm that tremendously reduced the number of operations required for corner detection.

The accompanying table provides an investigation of various algorithms described earlier and furnishes the best performance of the variants of each algorithm reported in their respective paper (Table 1).

3.1.2 | Neural network for object detection/recognition

The noteworthy advantages of event sensor are high temporal resolution and reduced data redundancy. The best suited neural network architecture for this kind of data is SNN, which is a biological counterpart of the artificial neural network. SNN is made up of spiking nodes and they are interconnected with edges called synapses. In this type of network, the neuron does not fire at every time instant, instead, it gets triggered only when its membrane potential reaches a specific threshold. When a neuron gets triggered, it sends a spike signal to its connected neuron which either increases or decreases the membrane potential

TABLE 1 Analyses of corner detection algorithms which could be used for object recognition

Methods	Merits	De-merits	Dataset	Event based or frame based	Accuracy
Clady et al. (2015)	Reduces noise by avoiding events such as a corner event and predicted corner event occurring within a given period of time	The accuracy depends on velocity estimation. At low speeds, the velocity estimation is not accurate and hence feature detection is error-prone	Not publicly available dataset (indoor and outdoor natural scenes). Indoor scenes had single contrasted object	Event based algorithm	Four different measures are used to quantify the error, (Spatial distance: 20% pixels <0.6 pixels, Velocity direction, velocity magnitude, Abs difference of vector)
Vasco et al. (2016)	Accuracy of feature detection does not depend on the speed of the object. False positives due to noise are mitigated as eigenvalues of the gradient matrix will be less for noise	Fixed number of events representation considered. This leads to different representation under cluttered and noncluttered scenes and hence if the threshold is not properly chosen, it might lead to overcrowding of corner detection around strong corners. Low response for corners with wider angle	Not publicly available dataset	Frame-based Harris corner detector extended to events	Average spatial distance error: <3 pixels
Mueggler et al. (2017)	It exhibits the advantages of conventional FAST corner detector. Effect of noise is reduced by avoiding gradient computation and by scrutinizing two different circles of pixels around the pixel under consideration	Multiple corner detection around the same corner. Fails to detect few corners depending on the motion of the camera	Mueggler et al. (2017)	Conventional frame based FAST detector extended to event data	Low texture: Number of matches is 88.9%, high texture: Number of matches is 96.8%
Alzugaray and Chli (2018)	Lesser computational time	Attracts more noisy corners. False corners removed with heuristics	Mueggler et al. (2017)	Algorithm derived from frame-based data	True positive rate: appr 10%, false positive rate: appr 10%

Note. The performance measure column gives the best performance among the variants of that particular algorithm/different databases used for that algorithm.

of the specific neuron. This encodes timing information in the working of the network, which makes it more suitable for asynchronous event data.

The major disadvantages of SNN are that (i) it cannot be easily adapted to detect bounding boxes and subsequently not suitable for object detection (however, it can be tuned for object classification) and (ii) SNN is not differentiable since spike signals are not differentiable and hence end to end training is certainly not a trivial task. Hence the history of the neural network for event data can be categorized into three different eras:

1. A naive approach: Integrate the events into quantized frame like data and use frame based CNN, which has well established training procedures. Training of CNN on event data is still an open research problem due to nonavailability of sufficient event data. However, there are few researches that endeavor to train on event data, but scalability is a major issue in this methodology.
2. An intermediate solution: Train CNN on frame based data and map these trained parameters into an equivalent SNN framework. In this methodology the complete statistics of the spiking inputs are not presented to the network during training time.
3. The best approach: A very recent development has come up with supervised learning techniques to train SNN directly using spike signals. However, this is a very complex task.

The following sections give details on the state-of-the-art ANN methods for event data object recognition.

Frame based CNN and its variants for event data

The present literature on Convolution nets focuses on the frame-based data. Frame-based CNN is a well-matured field. Hence, many researchers found it appropriate to apply the existing CNN architectures as used in classical vision on event data for object recognition. This work can be broadly classified into two types, (i) methods that utilize frame based CNN as such for event data and (ii) methods which have tried to rewire CNN architecture to take advantage of the event data structure.

Ghosh, Mishra, Orchard, and Thakor (2014) extended the conventional frame based CNN for the event based data. A spatio-temporal region of interest (ROI) was detected around objects. A temporal ROI was defined either as a fixed size window or a dynamic sized window that contained a fixed number of events. Spatial ROI was defined by location and size of the

object, which was estimated using the method described in Delbruck and Lichtsteiner (2007). A static image has been created for each identified spatio-temporal ROI and further classified using CNN implemented with Neuflow architecture (Farabet et al., 2010). The CNN was trained in MATLAB with event data.

Liu et al. (2016) proposed a technique for object recognition which involved ROI generation followed by the frame based CNN classification (Figure 6). ROI generation followed an event based tracker (Delbruck, 2008), which as and when a new event arrived, assigned the same to an existing cluster or created a new cluster based on spatio-temporal coherence of the events. As the background is cluttered due to the motion of the camera, the tracker has been used to generate ROI. It was followed by classification using conventional CNN available at.[2] The design of CNN was such that it has two convolution layers and two sub-sampling layers. The network has been fine-tuned with 373 positive samples (the objects to be recognized) and 1,500 negative samples (objects other than the desired objects) captured from DAVIS camera. CNN classification was followed by a particle filter based tracking in order to remove the false positives generated by CNN.

The method proposed by Cannici, Ciccone, Romanoni, and Matteucci (2018) has a slighter advantage over the other methods that use conventional frame based CNN for object recognition because the authors of this paper have tried to take advantage of the sparse nature of event data before feeding it into the frame-based CNN. Sparse events generated by event camera were integrated into a leaky surface before it was fed into the frame based CNN architecture. The leaky surface generation was an inspiration from the functioning of SNN. Whenever an event occurs, the corresponding pixel of the surface was incremented, thus maintaining time resolution in the generated surface. The architecture proposed was named as You Only Look Events (YOLE). In YOLE, the network architecture was an adaptation of LeNet (Lecun, Bottou, Bengio, & Haffner, 1998) and the training procedure on event data followed the trails of the popular YOLO framework (Redmon, Divvala, Girshick, & Farhadi, 2016).

The networks used in the methods discussed so far have made no attempt to take advantage of the merits of event data such as sparsity, time resolution etc., whereas the forthcoming methods have attempted to modify parts of CNN to accommodate the unique nature of event data.

Li et al. (2017) proposed a three-component modified frame based CNN, which consisted of a feature extraction module, a temporal pooling module and a detection module. The feature extraction part of CNN has been modified to adapt to the sparse nature of event data. As most part of the event data comprises high frequency and mid-frequency components, fewer kernels have been suggested to target these dominating frequencies present in the event data. As the event data is sparse, the initial layers were proposed to have a greater stride, thus not wasting the computational resource on noisy part of the data. The temporal pooling module is the main contribution of this paper. Event camera produces more events during faster motions than during slower motions. Hence, the proposed temporal pooling module accumulated data over a set of past frames depending on the motion content. It aggregated information from past frames over a period of variable time, that is, for more time for rapid motion and lesser time for slow motion. This results in motion invariant feature. The features extracted from these two modules have been trained in a manner similar to that of frame-based Faster RCNN detection framework. The reported performance score varies from 61.3 to 76% depending on the variant of the proposed method.

Cannici et al. (2018) proposed an architecture named as event-based Fully Convolutional Networks (eFCN). The YOLE architecture proposed by the same authors was not tailored to match event data, whereas eFCN has a convolution and pooling layer adapted to the sparse nature of event data. The event-based convolution layer maintained the previous state. As and when

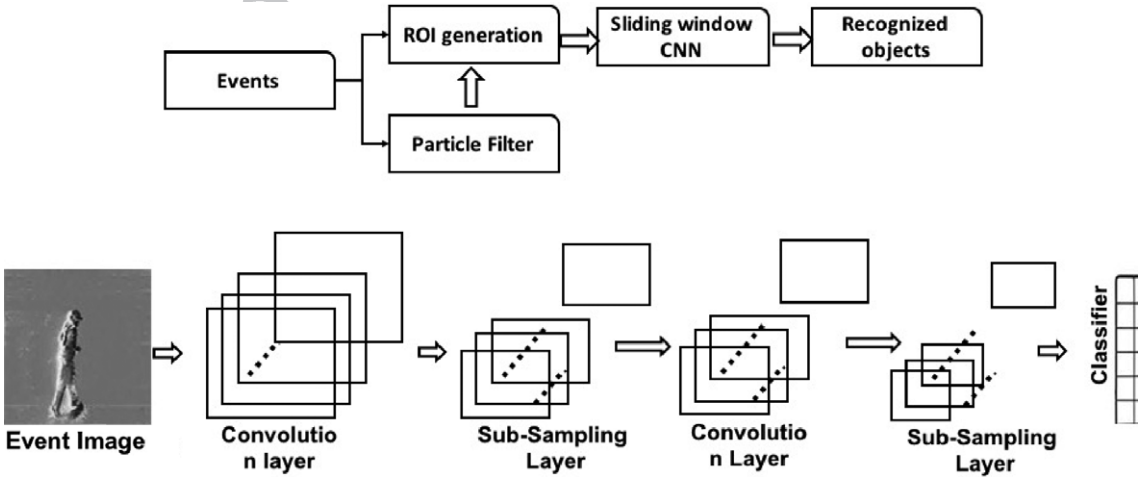

FIGURE 6 Framework of ROI generation followed by conventional CNN for object recognition. Top plot shows the overall architecture. Bottom plot shows the details of the CNN layers

an event approaches, eConv layer determined the part of the feature map that required an update and computed feature maps of the corresponding positions alone. As far as the architecture and training are concerned, the authors have followed the same strategy as that of YOLE (Table 2).

Convolutional neural network adaptation to SNN

In the recent past, spiking neural networks have become very famous due to their inherent nature in accelerating the computation and mimicking biologically similar neural network models. SNN is event-based, where the neurons update only as and when new information is available. In contrast to activation function in traditional artificial neural networks, the nodes in SNN receive inputs as binary numbers and integrate the input and fire an output only when the state of the neuron reaches a predefined threshold. This makes training of SNN a cumbersome task. The research field of training SNN is a recently developing area, whereas a number of works have shown that SNN can be obtained successfully via conversion from a conventional CNN. Hence, most of the SNN based object recognition tasks consist of a conventional CNN, fully trained with backpropagation and converted to simple spiking neurons during the testing phase.

The earlier work started with (Perez-Carrasco, Zhao, Serrano, & Chen, 2013), where frames were generated from DVS events during a fixed time interval, a frame-based conventional CNN was trained with this data and the parameters were mapped from the frame driven CNN to event data. The network was a six-layer feedforward ConvNet, with four convolution layers and two sub-sampling layers. The network was trained with DVS images and these trained weights were mapped to the event-driven version. The methodology followed to map these frame driven weights into the event-driven weights was the scaling of the former based on the three main factors of a typical spiking neuron, (i) characteristic time during which a given neuron receives a collection of events, (ii) refractory time which is the minimum time between two consecutive spikes that could be received, (iii) the threshold which forces the neuron to reset to its resting state. The method has been demonstrated on human silhouette orientation recognition and poker card symbol recognition. The major setback of this method is that it requires tuning of the network parameters for better accuracy.

Stromatias, Soto, SerranoGotarredona, and LinaresBarranco (2017) put forward an object recognition framework (Figure 7) composed of a frame-based feature extraction module and a spiking classifier made up of the event-driven neuron. This event-driven neuron was constructed as per the algorithm mentioned in Perez-Carrasco, Zhao, Serrano, and Chen (2013). The feature extraction module was composed of a one layer ConvNet, followed by a sub-sampling layer and a flatten module which rearranged the output of pooling layer into an one-dimensional vector. The flatten module was followed by the spiking classifier. The training of the architecture was applicable only to spiking classifier as the weights of other layers were fixed. ConvNet layer had 18 different orientation Gabor filters as its predefined weights. The flatten module had been trained using the unsupervised technique provided in Connor, Neil, Liu, Delbruck, and Pfeiffer (2016). The training of the spiking classifier was carried out in three steps, (i) For a set of events, a normalized histogram was constructed by counting the number of events at the output of the flatten module, (ii) a fully connected nonspiking classifier was trained using this normalized histogram, (iii) the scaled version of this learned classifier weight was transferred to spiking classifier.

It was followed by Cao, Chen, and Khosla (2014) who established a close link between the transfer function of a spiking neuron and activation of ReLU of conventional CNN. The usage of ReLU overcame the major issue involved in the representation of negative values and biases in spiking networks. ReLU was approximated by Integrate and Fire (IF) neuron with no

TABLE 2 Analyses of frame-based CNN and its variants of object detection algorithms

Methods	Merits	De-merits	Dataset	Accuracy
Ghosh et al. (2014)	The object detection does not depend on the speed of the object as the temporal ROI created accumulates a fixed number of events	Full potential of event-based data not used. Trained on frame based data	Not publicly available dataset and Simard et al. (2003)	99.10%
Liu et al. (2016)	Computation cost highly reduced by running CNN only on ROI generated by event data	Depends on frame data for object recognition	Robot predator and prey dataset	93%
Cannici et al. (2018)	SNN inspired frame integration procedure	Use of frame-based object detection network. Reduced performance reported with noise and lesser training data	N-Caltech-101, shifted N-MNIST, shifted MNIST-DVS, BlackBoard MNIST, POKER-DVS	94.9%
Li et al. (2017)	Motion invariant feature by adaptive temporal pooling of features from successive images. Reduced computational complexity by adopting large stride in initial layers to suit sparse nature of event data	The information carried by time-stamp of DVS events not utilized for object detection	Not publicly available dataset created from DVS	76.03%
Cannici et al. (2018)	Feature maps are tuned to utilize the advantage of event data, which makes it faster	On the process of redesigning the network to align with the nature of event data, the overall performance decreased	Shifted MNIST-DVS, BlackBoard MNIST	94.0%

Note. The performance measure column gives the best performance among the variants of that particular algorithm/different databases used for that algorithm.

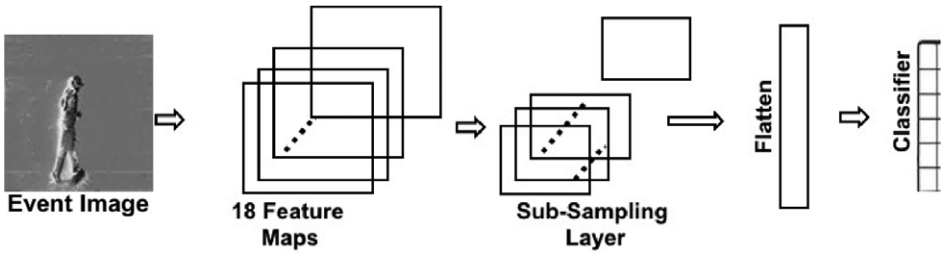

FIGURE 7 Framework of frame-based feature extraction module followed by spiking classifier for object recognition

refractory period. The output of ReLU and the number of spikes produced by IF neuron in a given period of time were considered to be proportional. Their method exhibits good performance in computer vision tasks that utilize average pooling layers. In spite of all these efforts, it was suffering from conversion loss, which was significantly improved in (Diehl et al., 2015), where the authors have used weight normalization to improve the performance. The two types of weight normalization that have been used are model-based normalization and data-driven normalization. The model-based normalization rescaled weights based on the maximum possible activation that could occur in a layer. In data based normalization, the training set was passed over the network after training and the weights were normalized based on the maximum possible activation that occurred within the training set. The improvement in performance via weight normalization was demonstrated by investigating the proposed method on MNIST dataset character recognition. The proposed conversion was giving less than 1% reduction in accuracy on small models, whereas on larger models, the conversion accuracy reduced drastically. This could not be used to convert any famous feature extraction architectures which involve layers such as max-pooling, softmax and batch normalization.

Neil, Pfeiffer, and Liu (2016) have extended the data normalization technique proposed above to reduce the computational complexity of the converted SNN. In order to decrease the response time, the number of spikes within a network has to be decreased. Especially in fully connected networks, the response time drastically reduces with the reduction in the number of spikes as each and every spike will lead to the trigger of multiple neurons down the network. In order to decrease the number of spikes, the paper followed sparse coding, a modified L_2 cost on activation and dropout. Sparsity was achieved through a regularization term which penalizes deviation from the target firing rate. As SNN firing rate is directly proportional to the activation of ANN ReLU, the proposed L_2 cost aims at penalizing high activations. High dropout may lead to loss of accuracy. Hence, the network was trained for initial few epochs without dropout. The probability of the dropout was slowly increased. It has been demonstrated that rate based ANN performance could be achieved using SNN with far lesser latency (Table 3).

SNN with predefined feature extractors

Training SNN is a cumbersome task. When the previous section explained stat-of-the-art of the algorithms that trained a conventional CNN and adapted the parameters to SNN, this section describes the recent works on SNN (Folowosele, Vogelstein, & EtienneCummings, 2011; Orchard, 2015) that suggest the use of predefined network weights such as Gabor filter weights.

HMAX model (Folowosele et al., 2011) is a hierarchical feed-forward model with alternating two main operations, Gaussian-like operations in S layer and nonlinear MAX operation in C layers. The proposed network has four layers, S1, C1, S2, and C2. The cells of S1 were designed to respond to four different orientations, 0, 45, 90, and 135°. It was followed by MAX computation by C1 cells, which was proportional to the MAX value of a group of S1 cells which were tuned to same orientation, but different receptive fields. This is similar to Winner-Take-All (WTA) layer, but the major difference being that MAX operation reveals only the amplitude of maximum value, not its identity. The cells of S2 layer were responsible for pooling operation, whose weights were determined by unsupervised learning. It was composed of two types of cells, 2×1 and 1×2. 2×1 cells were formed by pooling C1 cells which are present one on top of the other, whereas 1×2 cells pooled the C1 cells that were present adjacent to each other. It was followed by a C2 MAX operation layer. The HMAX model ended with a classification layer, which consisted of two cells, rectangle and triangle cell. The main setback of HMAX algorithm is that it does not operate in real time.

Orchard (2015) have proposed an extended hierarchical Spiking Neural Network (SNN) architecture for object recognition, called as HFirst (Figure 8), which could classify not just two classes (as in the previous paper), but nine different classes. The HFirst used spike timing to encode the strength of neuron activation, leading to a MAX operation based on temporal Winner-Take-All (WTA). The HFirst architecture had four layers named Simple 1 (S1), Simple 2 (S2), Complex 1 (C1), and Complex 2 (C2). The S1 layer consisted of 12 Gabor even filters (7×7 each) at 12 different orientations. It encoded the orientation information of edges at every pixel. Filter kernels are as given in Equation 2

TABLE 3 Analyses of object recognition algorithm that adapt CNN to SNN

Methods	Merits	De-merits	Dataset	Object detection or recognition only	Accuracy
Perez-Carrasco et al. (2013)	Generic method to convert frame driven neuron to event-driven neuron. Hence it can be applied to any ConvNet	Performance of the network highly depends on the proper tuning of a set of parameters such as refractory time	Not publicly available dataset	Recognition	Orientation detection (97.6%) and poker card symbol recognition (95.2%)
Stromatias et al. (2017)	The training method followed makes the implementation of the network easier. It gives good performance for both real and synthetic data. Work demonstrates a valid SNN classifier	Performance is highly dependant on leakage rate of the neuron.	Datasets derived from MNIST, N-MNIST and MNIST-DVS	Recognition	97.25%
Cao et al., (2014)	High energy efficiency when implemented in ultra-low power spike based neuromorphic hardware. Scalability as any CNN can be converted to tailored CNN and then to SNN	The computation of integrate and fire neuron is computed in double precision, which limits practical implementation of neuromorphic circuits	Tower dataset[14] and CIFAR-10 dataset Krizhevsky (2009)	Object detection	100% for few classes
Neil et al. (2016)	Computational complexity is greatly reduced by weight normalization and a novel drop out learning schedule while maintaining good accuracy	Weight normalization decreases the accuracy of SAE networks. In some cases, high drop-out followed in this paper lead to reduced accuracy	MNIST	Recognition	98%

Note. The performance measure column gives the best performance among the variants of that particular algorithm/different databases used for that algorithm.

$$F_\theta = \exp^{-\frac{x_0^2+\nu^2 y_0^2}{2\sigma^2}} \cos\left(\frac{2\pi}{\lambda} x_0\right), \tag{2}$$

where,

$$u_0 = u\cos\theta + v\sin\theta, \tag{3}$$
$$v_0 = -u\sin\theta + v\cos\theta,$$

where, x and y are horizontal and vertical pixel locations, θ varies from 0 to 165° in increments of 15°. λ and σ are variable parameters in the generation of synaptic weights. C1 layer had 12 neurons, each fed from all 12 S1 neurons. C1 neurons performed temporal MAX operation as discussed below. Each neuron of S2 layer was connected to all C1 neurons. The receptive fields of these neurons were learnt during the training phase. A separate S2 neuron was assigned for each class to be recognized. The highest synaptic weights were assigned to locations where the orientation of the character to be recognized matches with the orientation of the C1 neurons. The C2 layer was optional. The use of the C2 layer was to pool the responses from S2

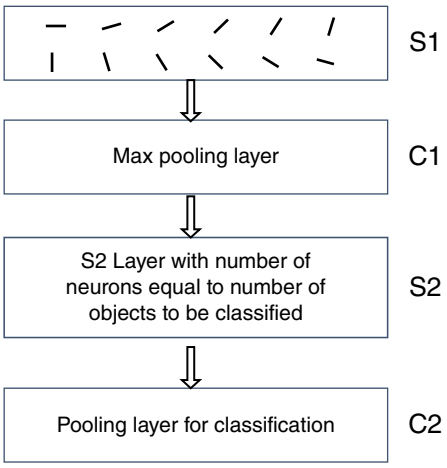

FIGURE 8 HFIRST architecture. S1 layer gives Gabor filter response, C1 layer is a MAX pooling layer. A separate S2 neuron for each class

TABLE 4 Analyses of object recognition algorithms which propose SNN with predefined feature extractors

Methods	Merits	De-merits	Dataset	Accuracy
Folowosele et al. (2011)	The model proposed is similar to visual cortex neural computation responsible for object recognition	It was demonstrated for binary classification. It does not operate in real time. It focuses only on "immediate object recognition" methodology of the human eye.	Images of basic shapes	84.6%
Orchard (2015)	Uses timing information from event data. Processes video data continuously. Can recognize multiple objects simultaneously. No computation is performed when there is not much activity in the scene	Training assumes a static view of objects. Hence the objects are tracked before feeding it into the network for training. This causes a drop in accuracy even with few milliseconds of timing jitter which may occur in real time scenarios	Poker data Perez-Carrasco et al. (2013), character (not publicly available) dataset	Appr 97.5% for poker card and appr. 84.9% for characters

Note. The performance measure column gives the best performance among the variants of that particular algorithm/different databases used for that algorithm.

neurons for classification. The probability of each class was estimated as $p(i) = \frac{n_i}{\sum_i n_i}$, where n_i is the number of spikes emitted by the S2 layer with respect to each class i.

The neuron model used in the HFirst architecture was Integrate-and-Fire (IF) neuron, as developed by Izhikevich (2004). The membrane voltage V_{mi} at time i of m^{th} neuron was updated only when it received an input spike after a predefined refractory interval. If the membrane voltage exceeded a predefined threshold, then the corresponding neuron emitted a spike that was fed to the next layer. The nonlinearity involved in the HFirst architecture was the MAX operation in C1 and C2 layers. The MAX operation was implemented as that of finding the neuron which responded early in the temporal domain. The neuron whose input spike pattern matched that of the weights will be activated heavily and tend to exceed the spiking threshold earlier. This mechanism was utilized to implement the MAX operation (Table 4).

SNN with back propagation for event data

Delbruck and Pfeiffer (2016) introduced a SNN architecture made up of Leaky Integrate and Fire (LIF) neuron and Winner-takes-all (WTA) circuits. The states of LIF neuron were updated asynchronously. The performance of the SNN architecture has been improved with the introduction of WTA, wherein as soon as a neuron produced an output, it would inhibit other neurons from spiking. The details of the transfer function of spiking neurons to allow backpropagation can be found in the paper. The generalization capability of SNN was improved with weight regularization and threshold regularization. The performance of previous versions of SNN has been improved to match that of conventional CNN.

3.1.3 | Time surface based object recognition

The best alternative to SNN is a form of representation of events built on a structure named as time surface. The most recent state-of-the-art works that tackle the problem of object recognition using time surfaces are Haessig and Benosman (2018), (Lagorce, Orchard, Galluppi, Shi, and Benosman (2017), and Sironi, Brambilla, Bourdis, Lagorce, and Benosman (2018a, 2018b). These methods generated time surface based feature for further classification. The time-surface S_i of an event e_i represents the temporal characteristics of all the events around any event e_i. It provides a spatio-temporal context around an event. The time-surface S_i of dimension $2R \times 2R$ (Equation 4) is obtained by applying an exponential decay (Figure 9) to the time of the last activity of the events in the neighborhood of the current event e_i at x_i, y_i.

$$S_i(u, v) = \exp^{-\frac{t_i - t_{u,v}}{\tau}}, \tag{4}$$

where, $u \in [x_i - R, x_i + R]$, $v \in [y_i - R, y_i + R]$, $t_{u,v}$ is the time stamp of the most recent event that occurred at u,v and t_i is the time stamp of the event at i and τ is the time constant of the exponential decay.

Haessig and Benosman (2018) learnt a basis function with the event time surfaces. The linear coefficients of the projection of the time surface of the incoming event on this set of learnt time surface basis formed the feature for further classification.

Projection Basis: In order to obtain a low dimensional feature for object recognition, the time surfaces were projected onto a set of basis function ϕ, which were estimated by formulating an optimization problem that minimized the error given in Equation (5). The error involved a sum of a reconstruction error term (first term of Equation (5)) and a coefficient sparseness term (second term of Equation (5)).

$$E = \sum_{x,y} \left[S(x, y) - \sum_{j=1}^{N} a_j \phi_j(x, y) \right]^2 + \lambda \sum_{j=1}^{N} \left| \frac{a_j}{\sigma} \right| . \tag{5}$$

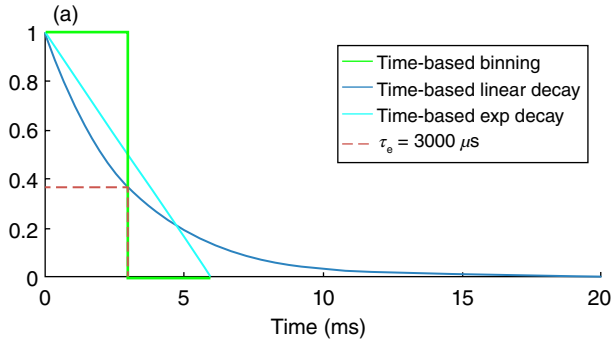

FIGURE 9 Various temporal weights that could be applied to events to generate memory surfaces. Figure courtesy (Afshar, Cohen, Hamilton, Tapson, & van Schaik 2016)

E represents the error, λ is the regularization constant, σ is a scaling constant and a_j is the projection constant onto the j^{th} basis.

The previous work was further enhanced by Lagorce et al. (2017) with a hierarchical representation of time surfaces. The architecture of object recognition involved generation of time surfaces, creation of time surface prototype, followed by a hierarchical representation of events based on these time surface prototypes. This hierarchical representation was fed to a classifier for object recognition. The time surface generation is as explained above. The time surface prototypes that were constructed on top of time surfaces are elementary representations of the time surfaces that occur in natural scenes. The time surface prototypes were generated by incremental clustering (Ballard & Jehee, 2012) of the time surfaces. Three different layers of time surface prototypes were generated. The time surface prototypes of different layers mainly differed in the space-time integration constants, which were approximately 50 ms for layer 1, 250 ms for layer 2 and 1.25 s for layer 3. The time surface prototype with a varying time constant is the heart of the proposed Hierarchy of Time Surface (HOTS) feature. When a new event arrived, a hierarchical representation of the same was created by comparing the time surface of the current event with time surface prototypes. Activations were produced in a layer when the time surface of current event matched the cluster center of the time surface prototype. The output of the previous layer formed the input to the succeeding layers. The activation of the final layer was considered to be the feature vector for object recognition. The major drawback of this algorithm is increased computational time which is the result of higher integration time in time surface generation and clustering process.

The computation time has been significantly reduced by Sironi et al. (2018a, 2018b) by proposing an effective feature known as the Histogram of Averaged Time Surfaces (HATS, Figure 10), which did not involve time consuming clustering process. In order to build feature representation, events $\{e_i\}_{i=1}^{I}$ with $e_i = (\mathbf{x}_i, t_i, p_i)$ were converted into local memory time-surfaces, which were in turn represented as HATS, where $\mathbf{x}_i = (x_i, y_i)$ are the coordinates of the pixels, t_i is the time stamp of the event and p_i is the polarity of the event. The pixel grid was divided into C cells of $k \times k$ size each. For each event e_i, the local memory time-surface τ_{e_i} was generated. A histogram h_c was generated for each cell considering τ_{e_i} of the events corresponding to that cell. The generation of τ_{e_i} and h_c are described briefly as follows:

Local Memory Time-Surface Generation: The Local Memory Time-Surface is an enhanced version of time-surfaces proposed in Lagorce et al. (2017). The Local Memory Time-Surface, τ_{e_i}, is a representative pattern of an event in spatial and temporal space, which was formulated as a spatio-temporal description on and around an event e_i, considering the history of the event in a temporal window of size $\bigtriangledown t$. Memory time surfaces were generally created by considering only the last event received in the neighborhood of the pixel under consideration, whereas local memory time surface took an average of the events received in the temporal time window $\bigtriangledown t$. Hence, it reduces the effect of noise and increases the robustness of the representation.

HATS Generation: Histogram generation followed the trail of conventional Histogram of Gradient (HOG) feature (Dalal & Triggs, 2005). For each cell C, the adjacent pixels were grouped and the corresponding histogram was formed by summing the Local Memory Time-Surfaces of each group of pixels of that particular cell. This HATS feature was used for further classification of objects (Table 5).

3.2 | Object tracking from event data

The two types of tracking are feature tracking and object tracking. Feature tracking finds its application in areas such as motion estimation, 3D reconstruction. Over the decade, object tracking has advanced as an important application in domains such as robotics. The objects of intrigue, for the most part, include pedestrian, vehicles and so forth. Object tracking has turned out to be widespread applications in surveillance and traffic monitoring. In this paper, we have limited our survey and discussion to object tracking.

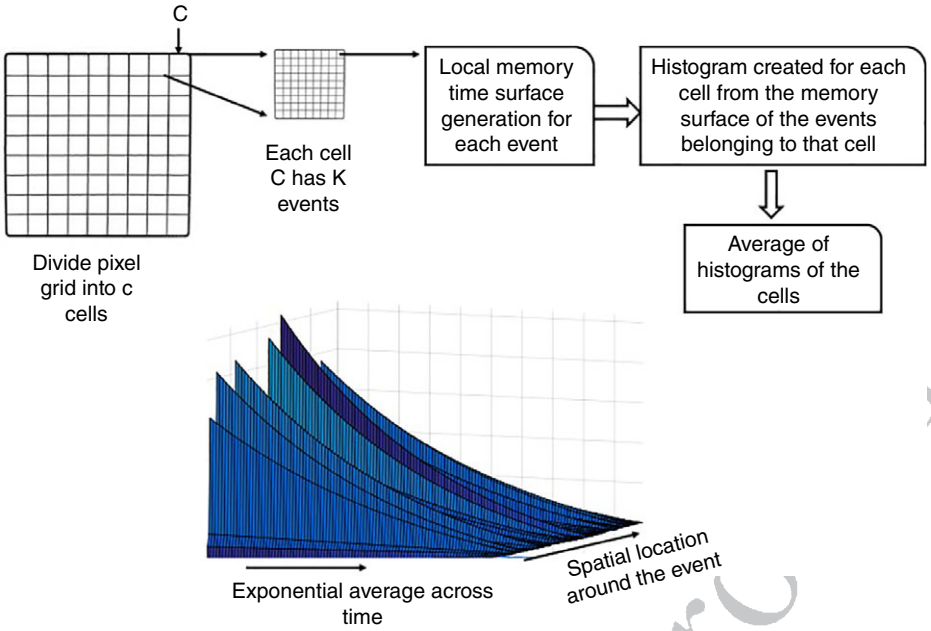

FIGURE 10 Top plot: HATS generation architecture, bottom plot: An example simulated local memory time surface. The pixel plane is divided into C cells. Local memory time surface is generated for each event in each cell. Histogram of each cell is generated by considering the events in the memory surface of that particular cell

For a system to track moving objects, it requires robust detection of the object of interest. An object of interest is formulated either as user input or as a preprocessing step of tracking, where objects are detected automatically. Challenging situations are scenarios where high-speed tracking is a mandate for real-time control of systems. In such scenarios, the event-based tracking algorithm discovers its utility increasingly than frame-based algorithms, in spite of the well-developed literature of the latter. In the recent past, an extensive literature has become available on event-based tracking. This segment gives an outline of the different tracking algorithms developed on event-based vision sensors. As a moving camera creates a lot of unwanted clutters, the method followed for object tracking under static and moving camera would differ a lot. Hence, the tracking techniques have been discussed as two different modules, one for tracking objects under static background and another under dynamic background.

3.2.1 | Tracking under static background

Tracking under static camera does not require background modeling as an event camera will pick only dynamic moving objects, unlike a conventional camera. Figure 11 shows a sequence of images captured from a static event camera. The methods discussed in this paper include event-driven approaches whose trackers were updated as per the characteristics of

TABLE 5 Analyses of time surface based algorithms for object recognition

Methods	Merits	De-merits	Dataset	Accuracy
Lagorce et al. (2017)	Time surfaces use spatial and time information of the event data, hence resulting in spatio-temporal features and giving good performance for dynamic objects. Hierarchical feature extraction allows information extraction from bigger spatial and time receptive fields to smaller spatial and time receptive fields	The features extracted heavily depend on speed and direction of movement of the objects. Parameter tuning is required to obtain better results. The receptive field of time is larger in the last layers of the hierarchical model as compared to that of the brain.	Card dataset SerranoGotarredona and Linares-Barranco (2015), letters and digits dataset Orchard et al. (2015), Face recognition dataset (Not publicly available)	100% for cards dataset, 100% for letters and digits dataset and 79% for faces dataset
Sironi et al. (2018a, 2018b)	Local memory time surfaces are robust to noise and small variations in the event data, as they are formed by utilizing the information carried by past events as well. The proposed algorithm can be easily parallelized	As it creates a histogram of time surfaces, it loses spatial information. Exponential weights are used for memory surface, it could be replaced by learnt weights to improve performance as indicated in the paper. Performance could be improved for low latency as well.	Cars dataset[15]	90.2%

Note. The performance measure column gives the best performance among the variants of that particular algorithm/different databases used for that algorithm.

FIGURE 11 Image sequence captured from a static camera. As the background is clean, tracking becomes an easier task

each and every new event. As per our analysis, the methods described below apply well established conventional vision algorithms to event data and does not utilize the full potential of the event data.

The earlier works of tracking mostly started with application specific requirements, such as pencil and pole balancing applications published in Conradt, Berner, Cook, and Delbruck (2009), Conradt et al. (2009) and robotic goalie (Delbruck & Lang, 2013). The pole balancing application involves balancing of a pencil mounted on an actuated table with two dynamic vision sensors mounted at right angles to each other. Each and every event generated by the movement of the pencil has been utilized to track the pencil in 3D position. The tracking algorithm proceeded in two stages: (i) The first stage was the recognition of the line corresponding to the pencil location with respect to the sensor plane, as and when a new event arrives. The estimated line was represented in Hough space. In Hough space, the lines are described as $y = m\mathbf{x} + b$, where $\mathbf{x} = (x, y)$ are the x and y coordinates of the event generated and m and b are the slope and intercept of the pencil and (ii) The second phase was to estimate the 3D position of the pencil from the estimated line. As the position of the sensors was fixed and known, the 3D position of the pencil was estimated by triangulating the pencil position as obtained from the two views.

Soccer goalie robot application involved tracking of the ball and tracking of the arm. An event-driven cluster tracker (Litzenberger et al., 2006) inspired by the conventional mean shift approach has been used for ball and arm tracking. The tracking algorithm proposed in this work was the result of influence experienced from that of conventional cameras. The algorithm proposed depicts an extension of the mean shift algorithm to event data. The events were continuously clustered and tracked as and when they occur. When a new event occurs at $\mathbf{x}$, it was assigned to a cluster whose center $\mathbf{x}_c$ was within seek distance of that cluster, R_c^k, from $\mathbf{x}$, that is, $R = \mid \mathbf{x}_c - \mathbf{x} \mid < R_c^k$. The cluster parameters of that particular cluster such as cluster center ($\mathbf{x}_c$), cluster size (R_c) and cluster seek distance $\left(R_c^k\right)$ were updated as follows:

$$\widehat{\mathbf{x}_c} = \mathbf{x}_c\alpha + \mathbf{x}(1-\alpha), \tag{6}$$

$$\widehat{R_c} = \max\left(R_{\min}, R_c\alpha + R(1-\alpha)\right), \tag{7}$$

$$\widehat{R_c^k} = \min\left(R_{\max}, R_c R_{\text{mult}}\right), \tag{8}$$

where, $0 < \alpha < 1$, $R_{\min}$, $1 < R_{\text{mult}} < 3$ are parameters of the algorithm. The max condition in R_c ensures that the cluster size does not exceed a set threshold. As the seek distance R_c^k for each cluster was estimated as a higher multiple of the cluster size, it allows the cluster to grow in size consistent to the size of the target. Though the clusters can change in size over time, they were restricted to be of circular shape.

The other work which has attempted event-based object tracking based on clusters was published in Schraml and Belbachir (2010). This algorithm differs from the previous paper in the way the events were assigned to clusters. The cluster assignment was depending on the 3D Manhattan distance between the incoming event and the clusters, estimated in space and time. This sort of cluster assignment reduces the noise effect as compared to the naive 2D Euclidian distance based assignment. The 3D position of the events was estimated with stereo dynamic vision sensor. The state-of-the-art FPGA implementation of event-based tracker with clusters can be found in LinaresBarranco, GomezRodriguez, Villanueva, Longinotti, and Delbruck (2015).

These cluster based techniques were suitable for embedded vision systems because of their low memory usage. However, these are application specific as the size of the clusters is tuned for that particular application. As these distance-based clustering results in hard boundaries, they are not well suited for multiple objects with occlusion. This has led to the development of tracking algorithms based on Gaussian mixture model (GMM) clustering techniques in works such as Piatkowska, Belbachir, Schraml, and Gelautz (2012), Lagorce, Meyer, Ieng, Filliat, and Benosman (2015).

Piatkowska et al. (2012) characterized the trajectories as k Gaussian clusters in space-time. As and when a new event occurred at time t, the probability of the event was estimated for the k clusters and further assigned to the cluster with minimum distance. After a time step of Δt, the parameters ($\mu_{i,\,\Delta t}$, $\sigma_{i,\,\Delta t}$, $w_{i,\,\Delta t}$) (mean, variance and weight) of each cluster i over Δt were estimated using standard Expectation Maximization update equation of GMM and the current model at time $t + \Delta t$ was updated as follows:

$$w_{i,t+\Delta t} = \alpha w_{i,\Delta t} + (1-\alpha) w_{i,t}. \tag{9}$$

$$\mu_{i,t+\Delta t} = \alpha \mu_{i,\Delta t} + (1-\alpha) \mu_{i,t}. \tag{10}$$

$$\sigma^2_{i,t+\Delta t} = \alpha \sigma^2_{i,\Delta t} + (1-\alpha)\left(\sigma^2_{i,t} + \mu^2_{i,t}\right) - \mu^2_{i,t+\Delta t}. \tag{11}$$

In this algorithm, the events were modeled by Gaussian clusters. This has been later extended by Lagorce et al. (2015), where the spatial distribution of events has been used to model bi-variate Gaussian. This was also an inspiration from the mean shift algorithm. The moving objects result in specific patterns in spatial distribution based on the specific object characteristics. This has been utilized to model each moving object as bi-variate spatial Gaussian cluster. The mean of the cluster represents the position of the object, whereas the covariance matrix is an indication of its size and orientation. As and when an event occurred, the probability of each cluster producing the particular event was estimated and the event was assigned to the cluster that had maximum probability. The parameters of the clusters converged iteratively with each incoming event and were updated as integration of past parameters with the current event information. The main advantage of this method is that in addition to tracking, it also furnishes information regarding the orientation and size of the objects tracked.

The method mentioned above was well suited for objects with simple shapes like ellipse, whereas real-world objects will be made up of several ellipses with an interconnecting geometric relation. The 3D to 2D projection of objects during image formation renders this geometric relation also to be varying. In order to take account of this, (Valeiras et al., 2015) proposed multiple Gaussian blob tracker for each and every object. Each object was modeled with multiple Gaussian blob tracker (Figure 12), for example, face was modeled with 26 trackers, one for each eye, eyebrows, nose, outline of the face, etc. These trackers were connected by springs, which were characterized by the maximum allowable change in length and stiffness constant. When there is deformation in the shape of the object during movement, the springs connecting them will either be elongated or contracted from its equilibrium position, which results in elastic energy. This elastic energy was utilized to determine whether a set of trackers belong to a particular object.

The works discussed so far may not perform better under object occlusions. The break through in tracking under occlusion happened with CamunasMesa, SerranoGotarredona, Ieng, Benosman, and LinaresBarranco (2018). The proposed algorithm combined object tracking method with 3D object reconstruction to improve the performance of the tracking algorithm under occlusion. The integration of 3D reconstruction with object tracking was such that each algorithm was benefitted by the feedback from the other. When an occlusion occurs, it results in confusion in cluster assignment. In that scenario, the algorithm

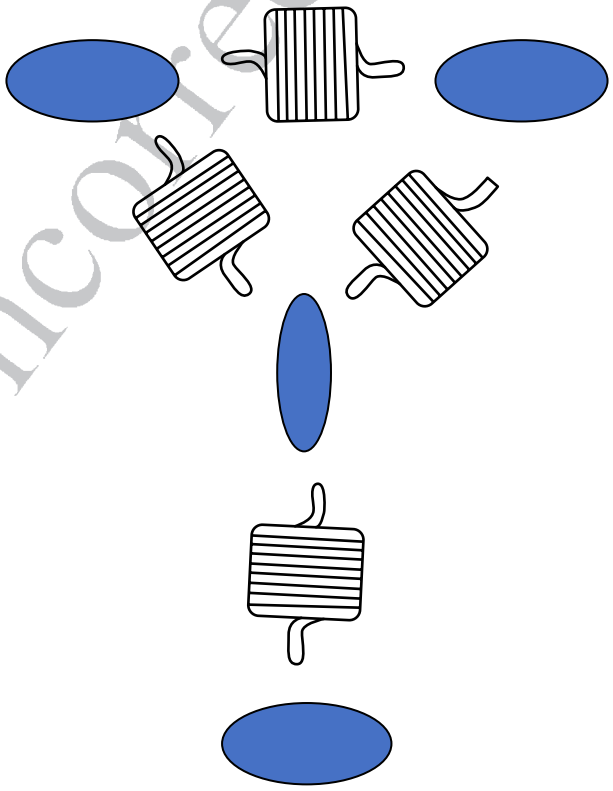

FIGURE 12 Gaussian blob tracker of face. Each part of the face such as eye, nose, mouth (Gaussian blobs of outline of the face is not shown here for clarity) is individually modeled as Gaussian cluster/blobs and a spring connectivity is maintained between them

assumed the current event to be belonging to all clusters with equal probability and verified the assumption based on the output from 3D reconstruction module. Though cluster tracking (Litzenberger et al., 2006) and 3D reconstruction (CamunasMesa, SerranoGotarredona, Ieng, Benosman, & Linares-Barranco, 2014) methods used in this paper were reported earlier in literature, the novelty of the proposed method lies in interlinking the output of 3D reconstruction and object tracking.

The cluster-based tracking algorithms are not suitable for the situations where there is a lot of clutter due to multiple moving objects. Cluster-based techniques will not be able to reject events that belong to clutter. Most of the tracking methods under cluttered scenarios employ probabilistic filter based approaches or filtering combined with Hough transform, etc.

Hough transform based circular objects (with the same radii) detection has been proposed in Ni, Pacoret, Benosman, Leng, and Regnier (2012). Though Hough transform depends on the specific shape of the object, the sequential implementation of the same makes it more preferable for event data. However, as Hough transform is sensitive to noise (Illingworth & Kittler, 1987), it is not an ideal candidate for the cluttered environment. Hence, Glover and Bartolozzi (2016) proposed a method called directed Hough transform. Instead of projecting all the points to Hough space, the authors have projected only the points which lie in the estimated direction of the circle center. This reduces the contribution from the event that belongs to the noncircular clutters. The direction of the center was estimated as the direction of the edge at that particular event. As edges are readily available from event data, direction estimation was a trivial task. This method reduced the effect of noise as against the techniques where Hough transform is used in its native form.

One another way that has been proposed in Barranco, Fermuller, and Ros (2018) to solve the problem of clutter is to combine mean shift clustering with particle filter tracking. This method does not assume the number of clusters or shape of the clusters. This makes it more preferable than other clustering based techniques, which require the number of clusters to be known a priori and assumes the shape of the clusters to be of a particular type. By augmenting clustering with particle filter tracking, this method overcomes the inability of the cluster based techniques to perform under clutter (Table 6).

3.2.2 | Tracking under dynamic background

Tracking in robotics involves object tracking along with ego-motion compensation. This is more challenging than static event camera, where the property of the event camera to produce no events under static background has been taken advantage to track moving objects. In moving event camera, background results in a lot of clutter (Figure 13). As distinguishing the signal generated by moving target from that of the ego-motion of the camera becomes difficult in scenarios where the camera is moving, there are only a couple of works which attempt to tackle object tracking in moving event camera. The earlier work started with Ni, Ieng, Posch, Regnier, and Benosman (2015), but the method was tested on simulated scenarios alone.

In order to develop a solution for real scenario object tracking with moving event camera, Liu et al. (2016) combined frame and event data. The possible ROI were detected using event-based cluster tracker. Due to the clutter of the background, the event based tracker may result in unwanted ROIs that belong to the background as well. In order to remove spurious ROIs, a frame-based CNN object classification was run on each ROI. The architecture of CNN comprised of two convolution layers and two sub-sampling layers. The filtered ROI of CNN was fed to particle filter based object tracker. CNN based classification was able to remove the spurious background clutters. However, the method relies on the frame based data as well, which depletes it from the advantages of using event camera alone.

In order to utilize the advantages of event data over conventional frame based data, tracking algorithm that depends on event data alone is mandatory. In Glover and Bartolozzi (2017), particle filter based approach was proposed to track objects in moving event camera. In a cluttered condition, a single event may not have sufficient information to update the particles. Hence, the events were accumulated for a time window under the assumption that the accumulated event will cancel the noise of background clutter. However, the time window should be lesser than the speed of the target. The time period of integration was different for each particle. However, the weight updates were done for all particles at same intervals of time as the particles have to be normalized before further processing. Though the tracking of objects seems to reject clutter, the time window greatly influences the performance of the tracker.

Another set of algorithms visualize the problem of object tracking in cluttered environments as ego-motion estimation followed by compensation to segment moving objects from the dynamic background. Especially in robotics, the moving platform is equipped with proprioceptive sensors such as Inertial Measurement Unit (IMU), which is used to estimate the ego-motion. However, the imprecise measurements of these sensors might lead to noisy motion compensation. This led to the development of algorithms which utilized event data itself to estimate the ego-motion, followed by moving object tracking (Vasco et al., 2017). The pipeline included feature detection, tracking and velocity estimation. In the off-line phase, a model was developed to represent the relation between the velocities of the vehicle and visual motion. During run time, the estimated visual motion was compared with the learnt model. If there is a significant difference in the estimated and computed motion, then it was considered to be generated by moving objects.

TABLE 6 Analyses of tracking algorithms under static background

Methods	Merits	De-merits	Dataset	Performance
Conradt et al. (2009) and Conradt et al. (2009)	Compact storage of estimate of the line (corresponding to pencil) in Hough space. The algorithm is suitable for hardware implementation	Not robust to noise	Demonstrated for pencil balancing application	Not reported in paper
Delbruck and Lang (2013) and Litzenberger et al. (2006)	As the algorithm is mean shift based, a-priori knowledge of number of clusters is not required. Suitable for low power and low memory systems (only a few parameters to be saved for each cluster)	The weights in the update equations have to be selected carefully to get good performance	Not publicly available dataset	Not reported in the paper
Schraml and Belbachir (2010)	Highly robust as it combines the advantages of density-based and distance based clustering algorithms. Suitable for embedded system implementation as the computational complexity depends only on the number of events and the number of clusters	Not suitable for multiple objects	Synthetic images and not publicly available dataset	Not reported in paper
Piatkowska et al. (2012)	Does not restrict the object shape to circle and problem of occlusion addressed. Enables multiple object tracking	Fails to detect objects which enter at the same time and are close to each other. Complex as compared to simple cluster based techniques	Not publicly available dataset	Recall of 0.9872 and precision of 0.7872
Lagorce et al. (2015)	Provides information regarding size and orientation of the objects tracked. Different kernels such as Gabor kernels are tested and provides a generalized kernel approach	Well suited for elliptical objects only. Involves computation of eigenvalues	Not publicly available dataset	Position accuracy: appr. 2.61 pixels, angle accuracy: appr. 1.59°
Valeiras et al. (2015)	As each object is modeled as multiple Gaussian blobs, it is well suited to track objects of any shape	Tracking is sensitive to the rotation in the image plane	Simulated shapes and face dataset (not publicly available)	Best mean error reported is 2.74%
CamunasMesa et al. (2018)	Information from object tracking and stereo matching fused to improve the performance. Works well for two objects whose trajectories cross each other multiple times. Not complex as it does not involve classification.	Depth estimation increases the computational complexity	Not publicly available dataset with pens, boxes, rotating straws and real-world scenario with two walking people	Success rate of 95.83%
Ni et al. (2012)	Provides an algorithm for microparticle tracking. Provides sub-pixel precision	Suitable only for objects which are circular in shape. Computationally expensive as it involves Hough transform	Not publicly available dataset of micro-sphere	Achieves sub-pixel accuracy
Glover and Bartolozzi (2016)	More robust to background clutter as it combines information obtained from Hough transform and optical flow	Works well for circular shaped objects (especially ball) only	Not publicly available datasets known as hand-move (HM) dataset and eye-move (EM) dataset	Performance is measured in terms of recall vs. precision, which is appr. 0.6 vs 0.7 for HM dataset and appr. 0.5 vs 0.4 for EM dataset
Barranco et al. (2018)	Works for multi-target tracking. Does not require number or shape of the objects to be detected	The minimization function of mean shift clustering includes Gaussian kernel, which increases the convergence time. As Kalman filter is used for tracking, it demands an accurate model of the system	Mueggler et al. (2017) and not publicly available dataset	Performance compared in terms of adjusted rank index (ARI: Around 0.941), normalized mutual information (NMI: Around 0.952), precision (around 0.981), recall (around 0.927) and F-measure (around 0.951)

Note. The performance measure column gives the best performance among the variants of that particular algorithm/different databases used for that algorithm.

The previous work needs an offline generation of a model for a given scene and hence it leads to the disadvantage of separate processes for 3D motion estimation and segmentation. This was followed by the work of Mitrokhin, Fermuller, Parameshwara, and Aloimonos (2018), where the ego-motion estimation and tracking were together considered as a single problem. This was the first work to tackle both the problems simultaneously. This was developed by being inspired from 3D point cloud

FIGURE 13 Object tracking for a moving camera. The image (available in the internet) is captured from DVS camera. As the camera was also moving, background generates a lot of clutters. Hence, tracking becomes a nontrivial task

processing techniques such as Kinect Fusion (Izadi et al., 2011). The tracking pipeline consisted of motion compensation (Figure 14) followed by motion inconsistency analysis. Motion compensation led to a new form of representation of events known as time-image. The time-image is a plane where each pixel contains the average timestamp of warped events that belong to that particular pixel (Figure 15). This reduces the effect of projecting events of different edges onto the same point which could be observed in works such as Gallego and Scaramuzza (2017), where the count of events was used rather than the time stamp of the events. The warping of pixels results in motion estimation, where the motion is modeled as a shift parallel to the image plane, motion towards the image plane and rotation around z-axis of the image plane. The events whose motion deviates from the estimated global motion were segmented, which results in tracking (Table 7).

3.3 | Vision-based localization

Localization, otherwise known as ego-motion estimation, is the process of estimating the pose of the robot in terms of translation and rotation from the associated images. The process of estimating the pose effectively plays a noteworthy role in the navigation of mobile robotics application. Localization for the conventional camera is a very much examined problem. However, the algorithms to be deployed for event-based cameras should be different from conventional algorithms in order to leverage the higher temporal resolution of event-based cameras. Event-based visual odometry will pave the way for the design of highly reactive robots, as each incoming event contributes to the localization. It is a cumbersome task to localize from an event-based data, as a solitary event may not have much information to perform the estimation. Despite the challenges, considerable progress has been made in this field. The localization approaches can be categorized into three types as follows.

1. Only Localization: These algorithms concentrate on localization without any attempt to generate the self-map or to correct the errors that accumulate over a period of time.

FIGURE 14 Left: image before motion compensation, right: image after motion compensation. Figure courtesy (Mitrokhin et al., 2018)

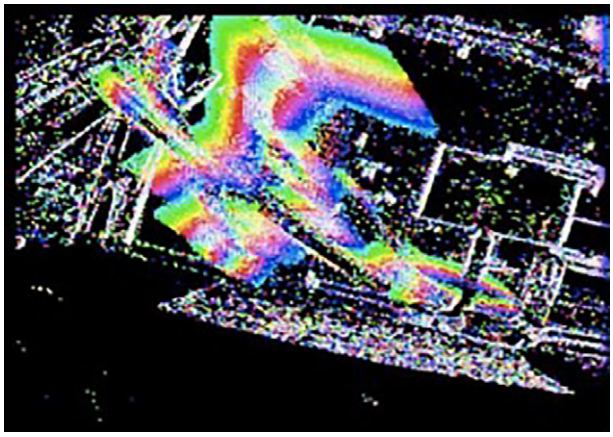

FIGURE 15 Motion compensated time image of a moving object drone. Blue represents event at time t_0 and green represents event at time $t_0 + \Delta t$. Though the moving object was occupying more space, motion compensation works well. Figure courtesy (Mitrokhin et al., 2018)

2. Localization and Mapping: Having a mechanism to self-generate map is mandatory in any real-time applications. Towards this, a lot of work has progressed in the field of localization and mapping for the event based camera. The two major assignments involve depth estimation and visual odometry estimation.
3. Methods with error correction module: Accurate pose estimation forms the foundation of any robotics application. A significant bottleneck of localization is the overwhelming nonconvexity of the optimization problem, thus leading to drift in trajectories over time. The two famous techniques followed to mitigate the effect of error are (i) Loop closure based techniques which employ methods such as bundle adjustment and pose graph optimization to minimize the drifts in visual odometry estimation and (ii) fusion of vision with IMU which results in reduction of trajectory ambiguities that would occur when using the camera or IMU alone.

3.3.1 | Visual odometry

The methods that deal with localization estimation alone can be categorized into works that handle constrained motions (3 degrees of freedom [DoF]) such as rotation only or planar alone and those that estimate the most complex case of freely moving camera (6 DoF).

TABLE 7 Analyses of tracking algorithms under dynamic background

Methods	Merits	De-merits	Dataset	Accuracy
Liu et al. (2016)	Region of interest generated by event camera is utilized for CNN recognition, thus reducing the computational complexity	Depends on frame data for tracking in event data	Predator-prey dataset	Performance measured in terms of accuracy vs. allowed error distance. For a distance threshold of 20 pixels, it achieves an accuracy of 90 mod
Glover and Bartolozzi (2017)	Gives good performance even when the relative velocity between the camera and the object varies.	The maximum target speed that could be tracked depends on the update rate of the particle filter, which is in turn dependant on the computation power available. Once the target is lost, no attempt is proposed for target recovery	EM and HM dataset Glover and Bartolozzi (2016)	Average error of 4.5 pixels for HM dataset and 2.1 pixels for EM dataset
Vasco et al. (2017)	Works for arbitrarily shaped objects	Off-line calibration to generate a model to correlate the velocity of the camera and resulting visual motion is required. When an object stops moving or changes direction, tracking of the particular object fails	Not publicly available dataset	Performance measured in terms of recall vs. precision, achieving a precision of 90 mod at low recall.
Mitrokhin et al. (2018)	No off-line calibration is required. No prior knowledge about the scene or motion of the object is required. Proposed a novel representation called time-image representation of events	Performance deteriorates under a noisy situation such as low light conditions. The algorithm requires enough of background to be visible for robust results	Extreme event dataset (EED)	Performance is measured in terms of success rates, which was 92.78% for "fast-moving drone" sequence

Note. The performance measure column gives the best performance among the variants of that particular algorithm/different databases used for that algorithm.

3 DoF motion estimation

The earlier work started with Weikersdorfer and Conradt (2012) for 2D motion estimation with event sensor alone. The 2D planar motion of the robot was estimated by applying particle filter on the features that the robot observes on the ceiling. For every event that arrived, a set of particles were updated based on a motion model and the score of each particle was updated based on the likelihood of occurrence of the current event with respect to the estimated pose. The particles were resampled if significant changes have occurred since the last event. A prerecorded ceiling map was provided to the robot during test time. The algorithm works with very high time resolution as it processes each and every event.

Parallel to the estimation of 2D planar motion estimation, there was huge progress in literature to estimate the rotation of the camera. The earlier preliminary work started with Cook, Gugelmann, Jug, Krautz, and Steger (2011). It became more matured with Kim et al. (2014), which simultaneously generated high-resolution panorama images of natural scenes in addition to the estimation of rotational motion. There was no attempt to estimate depth or translation. The approach involved two probabilistic filters, one to track the rotational motion of the camera event by event with respect to the scene mosaic and another to estimate the mosaic from the gradient image. The pixel-wise gradient estimation was carried out as follows: On receiving an event at a pixel location in the camera frame, the corresponding location in the panoramic map frame was estimated using the pose of tracking algorithm. Each pixel was initialized with a gradient of zero vectors. The gradient estimate was improvised using the brightness constancy assumption whenever a new event occurred. The brightness constancy states that the difference in the log gray level from the last event is equal to the log intensity, which has triggered the event. This estimated gradient image was utilized for the creation of scene mosaic by the method mentioned in Tumblin et al. (2005).

Gallego and Scaramuzza (2017) have proposed an event-based method to estimate the rotational motion of the camera searching for the motion parameters that maximize the strength of the motion-compensated edges. As the problem involved the estimation of rotational motion alone, there was no requirement of depth estimation. As a first step, event images were estimated from events. Under rotational motion, given a point x_0, its path in the image plane will be described by $x'(t) = W(x, \omega(t))$, where W is the warp function determined by the angular velocity, $\omega(t)$ alone. In calibrated coordinates, such a warp is defined by 2D homography.

$$W(x;\omega, t) \sim \exp(\hat{\omega}t)x, \tag{12}$$

where, $\hat{\omega}t$ is the rotation matrix $R(t)$ for constant angular velocity, $\hat{\omega}$ is the cross-product matrix and x is the point to be warped. With this concept, the event that occurred at the location, x_k in the image plane at time t_k is warped to the location x'_k as:

$$x'_k = W(x_k;\omega, t_k - t_0), \tag{13}$$

where, t_0 is the time of the first event in the reference set. The 3D rotation angle $(t_k - t_0)\omega$ is different for each event as it depends on the time of the event. An image, $I(x; \omega)$ of rotated events was produced by summing the value of rotated events at each pixel location. $I(x; \omega)$ was used to estimate the ω that aligns the events triggered by a particular scene point to their corresponding image point. This is done by contrast maximization of the rotated image.

With the advent of deep learning in multiple domains, Maqueda et al. (2018) proposed a method to estimate rotation from the event-based vision sensor using deep-learning. The overall architecture of the proposed algorithm consisted of a feature extraction layer followed by steering angle prediction layer. The events were converted into two synchronous frames by averaging the positive and negative polarity event data separately over a period of time. These event frames were fed to a deep neural network, ResNet, which learned a regression task to output the steering angle of the vehicle. As the ResNet has been mostly deployed in classification tasks, the convolutional layers of the ResNet have been used to extract features. The extracted features were subjected to a global average pooling in order to reduce the over-fitting by reducing the number of parameters. The global average pooling layer was followed by a 256-dimension (for ResNet18) or 1,024-dimension (for ResNet50) fully connected layer. The final layer of the network was a one-dimensional steering angle prediction layer, which was preceded by a nonlinear ReLU layer.

6DoF motion estimation

The last section was limited to 2D motion estimation or rotation estimation. In Censi and Scaramuzza (2014) proposed a 6DoF pose estimation that combined an event camera with CMOS sensor. The frame-based camera was synchronized with the event camera before the start of pose estimation. As and when a new event occurs, the 6DoF pose was estimated by estimating the likelihood of observing the present event given the motion model and the most recent low-frequency frame. As the time resolution of event data is very high, the estimation of every degree of the pose such as yaw, roll, pitch, translation was decoupled. The method gives freedom to implement it as a pure asynchronous algorithm or to accumulate pose over a series of events to acquire a robust pose estimate.

In the above paper, the authors have used CMOS sensor, which limits the speed capability of localization estimation. Hence, the authors have attempted to use DVS alone for 6DoF estimation in Mueggler et al. (2014), where the pose was estimated by minimizing point-to-line reprojection error. Estimation of 6-DoF motion occurs as and when an event arrives. As the pose was estimated asynchronously, there is no latency introduced, which is the major contribution of the paper. DVS mounted on a flying platform looked at a black square on a white background. The algorithm accumulated events until the border lines become detectable for Hough Transform and a four-sided shape was created with the detected four lines. On the arrival of a new event, the closeness of the event to any of the lines of the square was checked. If the event succeeded to be closer to any of the line, then the closest point in the line was replaced with this event. Following this, the pose was estimated by minimizing the error (Equation (14)) between line reprojections and the events belonging to it.

$$\xi^* = \arg\min_{\xi} \sum_{l=1}^{4} \sum_{i=1}^{N} \| d\left(\pi(L_l, \xi), e_{l,i}\right) \|, \tag{14}$$

where, L_l indicates the line belonging to the pattern, $\pi(L_l, \xi)$ is the projection of the line onto the image plane, ξ is the pose, $e_{l,i}$ denotes the event i belonging to line l and d is the distance between the event and the line. However, this method was developed for artificial black and white line maps.

Following their previous paper on localization, Mueggler et al. posed a novel trajectory estimation approach (Mueggler et al., 2015), where localization has been addressed as a continuous time problem estimated on a given map, as and when events were obtained from event-based vision sensors asynchronously. The trajectory estimation over an interval T was posed as a probabilistic formulation. The formulation seeks a DVS state $\boldsymbol{\xi}(t)$, which is, pose over the given interval, which maximizes $p(\boldsymbol{\xi}(t)/M, e_{1:N})$, where M is the given map and $e_{1:N}$ is the set of measured events $\{e_1 \ldots e_N\}$. $p(\boldsymbol{\xi}(t)/M, e_{1:N})$ is given as:

$$p(\boldsymbol{\xi}(t)/M, e_{1:N}) \propto p\left(\frac{e_{1:N}}{\boldsymbol{\xi}(t), M}\right) p(e(t)). \tag{15}$$

In the absence of the prior belief, the problem reduces to maximizing the likelihood of $p\left(\frac{e_{1:N}}{\boldsymbol{\xi}(t), M}\right)$.

Contrary to the previous work, Mueggler et al. proposed a low latency 6DoF estimation which is not restricted to artificial scenes (Gallego et al., 2016). Maps were assumed to be available during pose estimation. These maps were obtained by a previous run of dense standard reconstruction approaches of CMOS camera such as DTAM (Newcombe et al., 2011) or REMODE (Pizzoli et al., 2014). The pose was updated using Bayesian filtering approach as and when a new event occurred. The main contribution of this paper is that they have put forward a normal-uniform (normal for good measurement and uniform for noisy measurement) mixture model for likelihood function considering the fact that event data is prone to have outliers. Another noteworthy contribution is the approximation of the posterior with tractable exponential distribution which allows for closed form solution for pose estimation.

Following the recent advances in deep learning approaches, Nguyen et al. (2017) posed six-DoF localization task as a regression problem of the deep neural network (Figure 16). A pose was modeled as $[\boldsymbol{\xi}_p, \boldsymbol{\xi}_q]$, where $\boldsymbol{\xi}_p$ is the position and $\boldsymbol{\xi}_q$ is the quaternion representation of rotation. The pose regression network proposed was composed of two modules: CNN and SP-LSTM (Stacked Spatial LSTM). The CNN network was used as a feature extractor. The output of the CNN was fed to a drop-out layer to avoid overfitting, after which it was reshaped to 64×64 to feed it into the LSTM with 256 hidden units. Generally, the LSTM is used to learn temporal dependencies, whereas, in this work, the LSTM had been used to learn the spatial dependencies between 64 features, each with a 64-dimension feature vector. Two such LSTMs were used to capture hidden semantics of the data. The output of the LSTM was leading to a fully connected layer with seven neurons (that model the parameters of the pose) through a fully connected layer that has 512 neurons as output (Table 8).

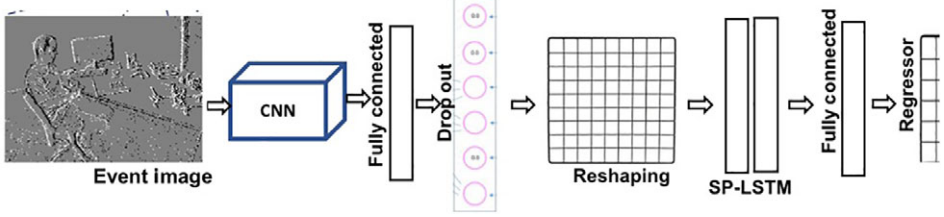

FIGURE 16 The network consists of a feature extraction CNN, a fully connected layer, followed by drop-out and reshaping and SP-LSTM. Finally it gives the six dimensional pose as regressed output

3.3.2 | Localization and mapping

Tracking problem has been well addressed in event data domain as it has lesser degrees of freedom as compared to tracking and mapping. As mapping with event data is a complex task, it started with 2D map generation and later developed into 3D map creation. Some methods also tried to estimate the map by relying on additional sensors such as RGB-D or conventional CMOS sensors, but it leads to the same bottlenecks of standard frame-based systems such as high latency.

In Weikersdorfer et al. (2014a) visual odometry algorithm of event camera was augmented with a depth sensor. This restricts the type of scenarios where event based localization could be deployed. The next stage of development in parallel localization and mapping was the introduction of 2D map estimation. The state-of-the-art works that generate 2D map while estimating 2D planar motion are Weikersdorfer et al. (2013) and Reinbacher et al. (2017a). The authors of Weikersdorfer and Conradt (2012) extended their work to 2D mapping and localization (Weikersdorfer et al., 2013). For each event e_k, the event-based tracking algorithm proposed initialized a particle set $P_k \in \Omega$ for the current system state (position and orientation of the sensor) and a score value $\in[0, 1]$ representing the likelihood of the state estimate. The major contribution of this paper over Weikersdorfer and Conradt (2012) is the introduction of dynamic 2D map estimation with event-based vision sensors. Each location on the map indicates the likelihood that the particular event will be generated by the event-based sensor that moves as per the estimated motion model. The position was estimated by projecting the events on the generated 2D map. Reinbacher et al. (2017a) also proposed a panoramic mapping and tracking. This method differs from the previous in tracking methodology. The 3D tracking that was followed in this paper is an inspiration from conventional direct alignment approaches such as Newcombe et al. (2011).

2D maps restrict the use of the localization module to planar scenarios. In order to extend the capability of the event based localization and mapping to complex scenarios, Weikersdorfer et al. (2014a) have augmented the event sensor with a depth-sensing camera. This augmented camera gave 3D coordinates of the events generated. With the map provided by the depth sensors, the authors tracked the 6DoF motion of the event camera with the particle filter based odometry estimation proposed by them in Weikersdorfer and Conradt (2012). This restricts the robot from generating the map without any additional sensors.

The section that follows will present state-of-the-art methods that generated 3D map with event camera. Kueng et al. (2016) proposed an event-based mapping and tracking. The proposed method alternated between two phases, mapping and tracking. During the mapping phase, features were detected and tracked with the hybrid feature detection and tracking method proposed in Tedaldi et al. (2016). The depth of these tracked features was estimated via triangulation of the current frame and its first detection, using the relative camera pose between the frames. The estimated depth was modeled as a random variable with a probabilistic distribution. The uncertainty of the depth was initialized to a very high value. As and when new measurements appeared, the variance and depth were updated through a Bayesian framework. When the variance decreased to a predefined threshold, the new 3D point was inserted into the map and became eligible for pose estimation utility. In tracking phase, with the given sparse 3D points, the current camera pose $\boldsymbol{\xi}_k$ was estimated by minimizing the reprojection error, $\arg\min_{\boldsymbol{\xi}} \sum_i \|\mathbf{x}_i - \pi(\boldsymbol{\xi}, \mathbf{X}_i)\|$, where $\mathbf{x}_i$ and $\mathbf{X}_i$ are 2D and 3D positions of i^{th} feature and π is the projection of 3D world points onto the image plane.

The previous method relied on frame based data for feature detection, though tracking was carried out in the event domain. This setback has been overcome in (Kim et al., 2016), which came up with a 3D motion estimation and reconstruction through three interconnected filters, one for tracking the motion, second for estimating the gradients, followed by log intensity reconstruction and third for estimating the inverse depth. The second and third filters were run on key-frame images, whereas the first filter runs on every image. Each and every filter is explained briefly as follows: (i) 3D Motion Estimation: An extended Kalman filter was used to estimate 6-DoF minimal representation of camera motion. For the motion prediction stage, 6-DoF constant position motion model was assumed. In the measurement update phase, the value of the measurement was estimated based on the given event, the current key-frame pose, and the current camera pose estimate. The measurement was used to update the 3D motion prediction. (ii) Log intensity reconstruction: EKF was employed on each pixel to estimate its gradient, which was in-turn utilized to reconstruct the log intensity image. The log intensity reconstruction was defined as an optimization problem that minimizes the error between the estimated gradients and the gradients of the reconstructed log intensity. (iii) Inverse depth estimation: Pixel-wise EKF was followed to estimate the inverse depth at each pixel by using the estimated pose and the reconstructed log intensity image.

The system proposed above included image reconstruction which increased the time complexity of the algorithm. In order to decrease the computational complexity of the algorithm, Rebecq et al. (2017) proposed a keyframe-based approach (Figure 17) which does not need image reconstruction and also does the costly map estimation operation only on key-frames. The framework had two interlinked modules, 3D tracking and mapping. Mapping of keyframes involved the generation of the disparity image, known as the Disparity Space Image (DSI). Keyframes were created whenever the distance between the current camera pose and that of the keyframe scaled by mean scene depth exceeded a threshold. In tracking, the global image

TABLE 8 Analyses of localization (only) algorithms that estimate either 3 DoF or 6 DoF motions

Methods	Merits	De-merits	Dataset	3DoF or 6DoF	Trajectory error
Weikersdorfer and Conradt (2012)	Purely event based. Does not employ any event integration. Hence operates with high time resolution accuracy	Assumes map to be made up of lines, which restricts its usage in real-world scenarios	Simulated and not publicly available		experimental dataset
3DoF	Average tracking error not provided. Tracking error vs distance traveled is provided in the paper				
Kim et al. (2014)	Builds mosaic of the scene without any additional sensing	No bootstrapping is used. This may result in convergence failure. The mosaic generation algorithm involves pixel-wise extended Kalman filter (EKF), which is computationally complex.	Not publicly available dataset	Rotation	Trajectory error not reported in the paper
Gallego and Scaramuzza (2017)	Predicts angular velocity from rotation without the requirement to estimate optical flow or image intensity. Reduces the effect of noise by smoothing the rotated events with Gaussian blur	Applicable only to rotational motions	Mueggler et al. (2017)	Rotation	Performance given in terms of angular velocity error which was having a maximum deviation of 80°/s
Maqueda et al. (2018)	Regression network used to predict angle, which allows modeling of angle as a continuous value	Transfer learning from conventional CNN to event-based data used for angle prediction. To use conventional CNN, events are accumulated to form frames	Binas et al. (2017a, 2017b)	Steering angle	Root mean square error of 4.58°
Censi and Scaramuzza (2014)	Low latency event-based algorithm	Requires input from frame-based camera as well and calibration of conventional and event camera	Not publicly available dataset	6 DoF	Angular drift of 1°
Mueggler et al. (2014)	Low latency algorithm that works on each and every event, thus making it suitable for quadrotor exhibiting high-speed maneuvers	Works with planar maps only	Simulated and real dataset	6 DoF	Mean position error of 10.8 cm and mean orientation error of 5.1°
Mueggler et al. (2015)	Trajectory is modeled as continuous time parameter to match the high speed of event camera	Map is assumed to be a set of 3D line segments.	Mueggler et al. (2014)	6 DoF	Position and orientation mean error for simulation is 5.47 cm and 1.74° position and orientation error for quad-rotor data is 4.6 cm and 1.8°
Gallego et al. (2016)	Event based pose update. The effect of noise is reduced by proposing a sensor likelihood function based on Gaussian mixture model. A closed form solution to filter update equation.	Assumes the existence of photometric depth map	Mueggler et al. (2017)	6 Dof	Position mean error of 3.97 cm and orientation mean error of 2.00°
Nguyen et al. (2017)	Does not require any map. The Pose is regressed through the network	Pose not estimated for every event. Events are stacked to get images and the algorithm works on these images	Mueggler et al. (2017)	6 DoF	Average position error of 0.028 m and average orientation error of 2.486°

Note. The performance measure column gives the best performance among the variants of that particular algorithm/different databases used for that algorithm.

alignment strategy (Forster et al., 2014) was proposed, which tried to minimize the geometric alignment between two edge images. The two edge images are an event image, I and a template M, which is the projection of a semi-dense 3D map of the scene using a known pose. The mapping module updates the local map from the events based on Event-based Multiview Stereo (EMVS) method proposed in Rebecq et al. (2016) (Table 9).

3.3.3 | Localization with error correction

In order to reduce the drift in trajectory estimation, the two prevalent state-of-the-art methods are loop closure and visual-inertial fusion.

Visual inertial fusion

It is a well-accepted fact that complementing the camera with an IMU aids in the resolution of trajectory ambiguities that would occur when using the camera or IMU alone. While there is a significant amount of literature available for the fusion of

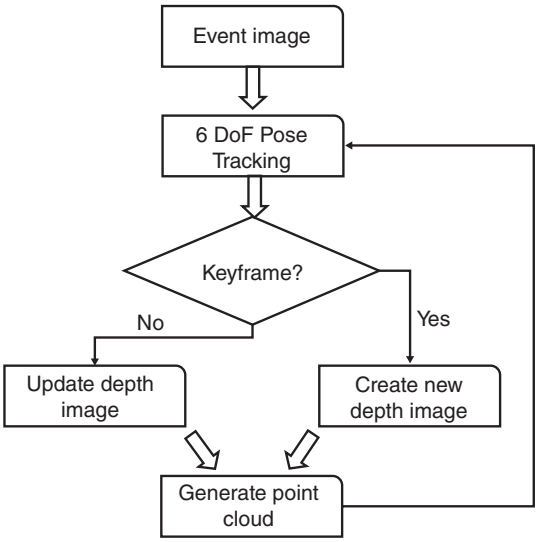

FIGURE 17 The architecture has a pose tracking module and disparity space image (DSI) generation module. DSI is generated every key-frame, which in turn gets created based on the movement of the camera

an IMU and a conventional camera, the study of complementing event sensors with the IMU is a developing research subject. In this section, we furnish a brief outline of the papers that have proposed novelties in visual-inertial fusion.

The major breakthrough in visual inertial event based SLAM came with the work of Zhu et al. which proposed an asynchronous filter-based algorithm to fuse event-based pose estimation with the IMU estimate (Zhu et al., 2017). It used a sensor package consisting of the IMU and an event-based camera, with the assumption of the IMU and camera frames being aligned with each other. The joint state of the IMU and event camera includes position, velocity and orientation of the global frame in sensor frame, accelerometer and gyroscope biases. The sensor was assumed to operate in an environment with landmarks $L := \{L_j \in R_3\}_{j=1}^{m}$ with a known initial state s_i. When an event-based camera generated events at discrete times t_1, t_2, …, the visual tracker used sensor state information to track the positions of landmarks (of the events generated) within the image plane. The mentioned tracking algorithm makes use of the fact that the events triggered by a particular landmark will fall on a spatio-temporal curve. The gradient along this curve is an indication of the optical flow of that particular landmark. Hence, tracking proceeded as estimating the projection of landmarks on the image plane, finding the spatio-temporal curve, and estimating the gradient along the curve. Having estimated the visual tracks, an EKF with a structureless vision model has been used to fuse the same with the available IMU readings. The structureless form of the EKF was used as it facilitates us to impose constraints between the poses that observe the landmarks as against the structured EKF, which tries to optimize over feature points as well. This proves to be a major achievement in limiting the size of the state vector to a greater extent.

The major hurdles of EKF are that it requires very robust features and it becomes computationally expensive with increasing map size. In contrast to the EKF-based fusion algorithms, Rebecq et al. (2017) proposed a smoothing approach based on nonlinear optimization on selected keyframes. It has two modules: front end, which estimated the features, tracked them and determined their 3D position via triangulation; and back end, which fused the IMU readings and triangulated feature tracks in a nonlinear fashion to update the camera pose and IMU parameters. The front end comprised the following steps: (i) Events spatiotemporal window: The given set of events $\{e_1, \ \dots, e_n\}$ were divided into a set of overlapping spatiotemporal windows $\{W_k\}$, where $W_k = \{e_{ks}, \ \dots, e_{ks+N-1}\}$ with N and s being the window size and step size, respectively, (ii) Event Frame Generation: A well-known approach to generate event frames is to sum the events that fired a response at pixel location $\mathbf{x}_j$. This does not convey good information to detect features, as a small window size will result in noisy images and larger window size will result in motion blur. Hence, the authors have motion-compensated each event, based on the timing of the event. The event frames generated from these motion-compensated events were further used for feature tracking, (iii) Feature detection and Tracking: New features were detected in keyframes using the FAST corner detector. The detected features were tracked using pyramidal Lukas-Kanade tracking. As soon as the landmark becomes eligible for triangulation, the corresponding feature was triangulated and added to the map. The proposed back end was based on the back end implementation of Leutenegger et al. (2015). The visual inertial localization problem was formulated as a joint optimization of a cost function that includes visual landmark reprojection error and inertial error terms.

The next leap in visual IMU fusion came with the introduction of tight coupling of the frame-based camera, event-based camera, and the IMU (Vidal et al., 2018) for pose estimation to leverage the complementary advantages of frame-based cameras and event-based cameras. This extended the work of Rebecq et al. (2017) to include optical cameras as well for fusion. In

TABLE 9 Analyses of localization and mapping algorithms

Methods	Merits	De-merits	Dataset	Trajectory error
Weikersdorfer et al. (2014a)	Processes each and every event and hence produces trajectory estimation with low latency. Computational speed is increased by performing resampling only after every k_{th} event	Needs a depth sensor for pose estimation	Not publicly available dataset	Root mean square error (RMSE) of position is 3.1 cm
Kueng et al. (2016)	Event-based. Hence preserves the low latency advantage of event sensor. Utilizes geometric information conveyed by edge features which are more prominent in event data. Event noise is filtered by supplementing tracking with local histogram	Initial step of feature detection requires frame data	Not publicly available datasets (checkerboard scene and natural scene)	For checkerboard experiment, the average position error is 16 mm and for a natural scene, the average position error is 30 mm
Kim et al. (2016)	Provides estimation of 3D reconstruction and pose estimation	Involves intensity image reconstruction, which is computationally complex	Not publicly available dataset	Only qualitative results are provided
Rebecq et al. (2017)	Parallel tracking and mapping methodology carry out costly 3D reconstruction only in keyframes. This reduces the computational complexity and latency. Noise in 3D reconstruction is reduced with a median filter.	Bootstrapping assumes planar map, which may not lead to convergence always. DSI is discretized to allow for depth estimation of far away objects, which might lead to loss of depth resolution.	Not publicly available dataset	Average drift over 30 m trajectory is 6 cm
Reinbacher et al. (2017a)	First event based parallel tracking and mapping approach, which gives map as well as pose only from event data. As it does not require image reconstruction, it is computationally effective. Works for natural scenes as well. Costly 3D reconstruction is carried out only on keyframes.	Does not estimate pose for each and every event. Works on event image, obtained by collecting events over a period of time	Not publicly available dataset	For multi key-frame trajectory, 6 cm translational error and 3° rotational drift

Note. The performance measure column gives the best performance among the variants of that particular algorithm/different databases used for that algorithm.

addition to feature tracks from the event-based camera, the algorithm estimated feature tracks from the frame-based camera as well. The feature tracks from the event camera, frame-based camera, and the IMU readings were fed into the back end module for joint optimization to estimate the new state.

The above methods estimated visual odometry from a set of events and fed the same into the standard frame-based Visual Inertial fusion techniques. This method may not be appropriate as events are asynchronous and the estimated pose may not correspond to any single time stamp. Moreover, the temporal resolution of events is very high. Hence, Mueggler et al. (2018) have adopted a continuous time framework. A similar technique of continuous time trajectory estimation has been published by the same authors for vision only system on line based maps. A similar system has been extended to the fusion of vision and IMU for natural scenes. The camera trajectory has been approximated as a smooth curve in rigid body motion space using splines. The trajectory estimation problem was put forth as a probabilistic approach that maximizes the likelihood of the trajectory over a period of time T for the given event measurements and IMU measurements (angular velocities and linear acceleration) (Table 10).

Vision with loop closure based error correction

In the conventional camera SLAM, the techniques such as bundle adjustment and pose graph optimization are used to minimize the drifts in visual odometry estimation. The application of bundle adjustment or pose graph optimization involves loop closure detection via place recognition. As place recognition with event data is a very cumbersome task, there has not been much progress on loop closure detection on event data and hence not many works published on error reduction using these techniques. The one recent work for pose estimation using place recognition is proposed in (Milford et al., 2015). This could form a baseline to integrate place recognition, loop detection and error correction into event-based localization and mapping. They have built a place recognition module on top of the conventional SeqSLAM algorithm (Milford & Wyeth, 2012). Event images were generated by binning the events which were then fed into the place recognition and loop closure modules of SeqSLAM. These modules were integrated into the system as proposed in (Milford & George, 2014).

4 | DATASETS AND SIMULATORS

Benchmark datasets assume an indispensable role in the development of research in various fields of computer vision. These datasets help to identify and conquer the gap in the research in these fields and enable quantitative comparison of algorithms

TABLE 10 Analyses of algorithms that fuse vision and IMU to improve the localization estimation

Methods	Merits	De-merits	Dataset	Trajectory error
Zhu et al. (2017)	Optimization over feature points is avoided which reduces the dimension of the state to be updated in EKF	It is assumed that optical flow within a small spatio-temporal window is constant	Mueggler et al. (2017)	Minimum mean position error of 1.23% and rotation error of 0.02°/m
(Rebecq et al., 2017)	Keyframe based tightly coupled visual inertial fusion. Outlier feature tracks are rejected by RANSAC	Works on event images, which is generated by accumulating events over a period of time. Window size becomes a critical parameter	Mueggler et al. (2017)	Minimum mean position error of 0.54% and yaw error of 0.03°/m
(Vidal et al., 2018)	Improved performance by combining standard frames with IMU and event data	As it includes frame data as well, it inherits the disadvantages faced with conventional camera	Mueggler et al. (2017)	Minimum mean position error of 0.10% and orientation error of 0.02°/m
(Mueggler et al., 2018)	Continuous temporal model assumed to cater for high temporal resolution of event data and IMU data. Continuous models allow to derive pose and linear velocity from a unique trajectory representation	Assumed that scene map is available	Not publicly available dataset	Mean position error of 1.48 cm and mean orientation error of 0.19°

Note. The performance measure column gives the best performance among the variants of that particular algorithm/different databases used for that algorithm.

developed across the globe. The availability of benchmark datasets for event-based vision sensors is limited as compared to their counterparts, the frame-based cameras. This can be chiefly ascribed to the commercial nonavailability of these sensors. Nevertheless, the researchers working on event-based vision sensors have driven the generation of event data dataset for various vision algorithms such as optical flow estimation, visual odometry, SLAM, object recognition, and action recognition.

Despite the fact that the event-based vision sensors have been commercialized, the nonavailability of a wide variety of event-based data still holds on. This has prompted the use of the existing frame-based vision datasets to simulate event data. This spares the effort and time spent in collecting data and furthermore allows for direct comparison between algorithms in the two-sensor domain.

This section gives a short layout of the datasets available for event-based vision sensors under two distinct categories, real dataset and simulated dataset.

4.1 | Real dataset

This section furnishes the details of publicly available event datasets captured on 3D scenes to evaluate the performance of optical flow, object recognition and SLAM.

4.1.1 | Optical flow

The dataset that accompanies (Rueckauer & Delbruck, 2015) is an optical flow dataset. The samples were recorded from a 240 × 180 pixel DAVIS240 sensor (Brandli et al., 2014a, 2014b). The dataset contains asynchronous events from the DVS, as well as synchronous frames. The dataset consists of two synthesized sequences, two simple real sequences, and a more complex real sequence. The synthesized samples were created in MATLAB. The authors have provided the script for generating synthetic sequences. The two synthetic sequences comprise a translating square and a rotating bar. The real sequences were captured by mounting the camera on a rotor that is allowed for any one of the motions, pan or tilt or roll. In order to generate ground truth, the underlying geometric principles were used in the case of the synthetic sequence and IMU, gyro sensors were used in the case of the real sequence.

4.1.2 | SLAM

Following the paper (Weikersdorfer et al., 2014b), Dynamic Vision/RGBD dataset (Dynamic Vision RGBD Dataset, 2014) was released. The data contains three data streams, RGB data, depth data, and event data captured from D-eDVS, a depth-augmented dynamic vision sensor proposed by the authors. The sensor consists of an embedded DVS with a resolution of 128 × 128, a depth sensor like PrimeSense RGB-D sensor with a resolution of 320 × 240 at 60 Hz. In order to compensate for the information missing in the temporal space of the RGBD sensor, the smallest depth value from the latest frame is chosen for each event. In addition to the temporal resolution difference, the spatial resolution difference should also be addressed. The mapping of world point $x \in R^3$ to RGBD pixel points $u_d \in R^3$ is defined as $u_d = \mathbf{K}_d x$, where $\mathbf{K}_d$ is the calibration matrix of the RGBD sensor. The corresponding point in the DVS is estimated as $\mathbf{K}_e \mathbf{T} \mathbf{K}_d^{-1} u_d$, where $\mathbf{K}_e$ and $\mathbf{T}$ are the calibration matrix of the DVS and the transformation matrix between the RGBD and event sensor, respectively. The dataset consists of five different scenarios. Each scenario comprises clips of duration 30–60 s each.

Mueggler et al. (2017) provide dataset captured from the Dynamic and Active Pixel Vision Sensor (DAVIS), which transmits frames in addition to events. Events are pixel-level relative brightness change. The DAVIS also carries an IMU that includes a gyroscope and an accelerometer, thus helping in the evaluation of visual-inertial algorithms. The axis of the IMU is coincident with that of the event sensor. The dataset comprises events, frames at 24 Hz, inertial measurements at 1 KHz, and ground truth pose from motion capture system at 200 Hz. The dataset consists of indoor and outdoor scenarios. The indoor dataset depicts the motion of objects of different shapes, poster and boxes. The outdoor dataset was acquired in an urban environment with walking and running people. The ground truth is available only for indoor data. The calibrations provided are intrinsic camera calibration and camera-IMU calibration.

The DDD17 dataset (Dynamic Vision Dataset, 2016; Neil et al., 2017) was recorded using the Ford Mondeo MK3 European Model using the DAVIS346B sensor, which contains an Active Pixel Vision (AVS) sensor and a DVS camera in order to record traditional frame-based data and event-based data simultaneously. The camera has a resolution of 346 × 240 pixels, with its architecture similar to that presented in Brandli et al. (2014a, 2014b). Since the AVS data and the DVS data are time-stamped with their own clocks, both data streams were supplemented with the system time of the recording computer for possible synchronization in the future. The recorded data includes 12 hr of data recorded under different weathers, driving and lighting conditions on different roads of Switzerland and Germany. The dynamic driving condition is achieved by varying the speed. Low light condition is simulated by driving the vehicle with the headlights off.

Zhu et al. (2018) presented a large dataset with a synchronized stereo pair of an event-based camera, DAVIS m346B, a 360 × 240 pixel camera with DVS, APS and IMU. The dataset contains APS gray scale images at 50 Hz, IMU measurements, left and right DVS camera events, stereo frame-based images from VI sensor, and ground truth data. The ground truth pose is recorded using a GPS and motion capture systems. The ground truth depth images for left and right cameras were recorded using the Velodyne VLP-16 LIDAR, which is mounted such that there is enough overlap between the field of view of the LIDAR and DVS. The data is captured by three different modes: (i) by mounting it on a hexacopter, (ii) by carrying it around, and (iii) by driving in a sedan. Hexacopter data contains indoor and outdoor scenarios. The motion capture system provides the ground truth pose. High-dynamic data are collected by carrying it in hand. The LIDAR SLAM provides the ground truth pose and depth. For slow to medium motion, the sedan is used for data capture. It includes day and evening situations. The ground truth pose is obtained from the GPS and LIDAR SLAM, whereas ground truth depth is provided by the LIDAR. The GPS gives additional ground truth reference in outdoor in all the three modes.

4.1.3 | Object recognition

Moeys et al. (2016) have made available the PRED18 dataset (Dynamic Vision Object Recog Dataset, 2016) to demonstrate predator/prey scenario, where the predator has to recognize the prey. The data was captured using a DAVIS camera, which has a 240 × 180 AVS that captures images at the rate of 15 Hz and a DVS that records events at variable rates ranging from 15 to 500 Hz depending on the speed of the robot. The camera was mounted on a Robotnik Summit XL mobile robot, running at a maximum speed of 2 *m*/*s*. The dataset consists of around 20 recordings that extend to a duration of 1.25 hr. The scenario recorded comprised prey robot and humans as well. Different lighting conditions were simulated by turning the lights on and off, by shutting the blinds, and by the ambient light that entered the room through unshaded windows. The ground truth labeling of prey robot was achieved by annotating the scene with the jAER software framework.

Sironi et al. (2018a, 2018b) introduced an event dataset called N-CARS (Dynamic Vision Object Recog Dataset, 2018) recorded from real-world scenes using the ATIS camera for a duration of 80 min. The camera was mounted behind the windshield of a car. The N-CARS is a two class dataset with 12,336 car samples and 11,693 noncar samples. Conventional gray scale images were created from the measurements of the ATIS camera. In order to generate the ground truth, gray scale images were processed with a state-of-the-art object detector, followed by the automatic extraction of the bounding box and the corresponding labels.

4.2 | Simulated dataset

Serrano et al. (Orchard et al., 2015; Gotarredona & Linares-Barranco, 2015; Hu et al., 2016) have converted the famous frame-based vision benchmark datasets into event-based data. As the event-based vision sensors are designed to respond to changes in illumination, the static images can be converted into events by simulating changes in intensity with respect to time I_t. Generally, intensity change over time is achieved either by flashing the images in front of an event-based sensor (flashing input method) or by moving the images over a computer monitor and seeing them through an event-based vision sensor (moving input method). In flashing input, the image display is alternated with that of a blank image. This methodology targets the evaluation of recognition problem alone, whereas the moving input method targets evaluation of algorithms in a real-world scenario.

Event datasets of the MNIST were generated via four different protocols, two of which are flashing-input-based and two are moving-input-based. Gotarredona and Linares-Barranco (2015) and Hu et al. (2016) followed the moving input method to simulate the MNIST-DVS dataset (Dynamic Vision Object Recog Simulated Dataset, 2015) and single object tracking dataset, action recognition dataset, Caltech-256 object recognition dataset (Dynamic Vision Object Recog and Action Recog and Tracking Simulated Dataset, 2016), respectively. However, the response thus obtained from the event-based vision sensor will differ from the sensor viewing the real 3D scene, as I_t depends on the depth of the scene, which is lost during perspective projection of conventional image capture. Hence, the authors of Orchard et al. (2015) have proposed rotation as the relative motion between the camera and the static image, as the rotational motion is independent of depth.

5 | OPEN SOURCE CODES

Open source codes are available for certain publications, thus enabling the researchers to quickly evaluate their algorithms against the literature. Open source codes can be found for the following computer vision technologies

1. Optical flow: Source Code for Optical Flow (2017a) provides code for the algorithm proposed in Benosman et al. (2013). Several algorithms compared in the paper (Rueckauer & Delbruck, 2016) are implemented in Source Code for Optical Flow (2017b). The code provided in Source Code for Event Lifetime (2017) augments each event with its lifetime, which is nothing but the time taken by moving brightness that caused the event to move one pixel. The code is based on the event lifetime estimation algorithm proposed in Mueggler et al. (2015).
2. Intensity image reconstruction: Codes developed in Source Code for Image Reconstruction (2017a) and Source Code for Image Reconstruction (2017b) reconstruct intensity images from the DVS as per the algorithms proposed in Kim et al. (2014) and Reinbacher et al. (2016), respectively.
3. Localization: Localization, also known as ego motion estimation algorithm is implemented in Source Code for Localization (2016) inspired by the algorithm proposed in Reinbacher et al. (2017b).
4. Object Recognition: Implementation of a spiking neural network to recognize objects from the DVS data (Orchard et al., 2015) is provided at Source Code for Object Recognition (2016).

6 | CONCLUSION

In this review paper, we have tried to provide a summary of vision algorithms developed for event-based vision sensors. We have not restricted ourselves to algorithms, but additionally explored the silicon retina in brief. The paper also provides an insight into numerous databases available to evaluate event-based vision algorithms.

The beginning of the paper gives a short introduction of cutting-edge vision sensors, referred to as the silicon retina. The goal of this section was to produce a helpful insight into the immensely improved capabilities of the silicon retina over the conventional camera. Further down, this section puts forward the varied types of silicon retinae and elaborates on a particular well-known type, the temporal difference silicon retina commonly known as activity-driven event-based vision sensor. The paper explains the image generation, transmission, and processing mechanisms of the event-based vision sensors, focussing on its benefits such as low latency and power efficiency. This is followed by a brief history of event-based vision sensors, with emphasis on the architecture of the DVS and ATIS. Before venturing into event-based vision algorithms, the paper analyses the dimensions that paved way for these sensors to gain their position in vision applications. For the sake of completeness, we have also provided a short note on the software frameworks, which are essential for the easy use of sensors and enhancement of interoperability.

The body of the paper involves an elaborated presentation of three major vision applications proposed in the literature to deal with event-based data. The commercialization of these sensors has accelerated the research in event-based vision algorithms. The three major vision applications discussed in this paper are those that have seen good development in event data domain in the past few years (object detection/recognition, object tracking and localization). The vision algorithms such as anomaly detection or activity analyses are in the early stages of development as far as event data is concerned. Although there are quite a few research papers in the field of event data processing, most of them are conventional vision algorithms adapted to event data. The vision algorithms for event data should stem from computational aspects of biological vision. As event cameras are a result of sensing from a biological perspective, the algorithms for event data should be bio-inspired, which will extract information from the data as biological vision does. Bio-inspired vision algorithms are a developing field, which demands the merger of biological vision and computer vision.

Availability of datasets is mandatory for any field of research to enable researchers to compare the algorithms developed across the globe. Though the event-based dataset is not as exhaustive as the frame-based dataset, lot of work has gone into acquiring and annotating the event-based dataset, both real and simulated. We have tried to give comprehensive information on the real dataset, which was captured with event-based vision sensors on real 3D scenes and as well on the simulated dataset, which was built on the conventional frame-based dataset through varied conversion mechanisms.

CONFLICT OF INTEREST

The authors have declared no conflicts of interest for this article.

ENDNOTES

[1]https://github.com/uzh–rpg/event– *based_ vision_*resources.

[2]https://github.com/rasmusbergpalm/DeepLearnToolbox.

[3]http://ilab.usc.edu/neo2/dataset/.

[4]http://www.prophesee.ai/dataset-n-cars/.

RELATED WIREs ARTICLE

Local computation of optical flow using a silicon retina

REFERENCES

Afshar, S., Cohen, G., Hamilton, T., Tapson, J., & van Schaik, A. (2016). Investigation of event-based memory surfaces for high-speed tracking, unsupervised feature extraction and object recognition. *arXiv preprint arXiv:1603.04223*.

Alzugaray, I., & Chli, M. (2018). Asynchronous corner detection and tracking for event cameras in real-time. *IEEE Robotics and Automation Letters*.

Ballard, D., & Jehee, J. (2012). Dynamic coding of signed quantities in cortical feedback circuits. *Frontiers in Psychology*, *3*(254).

Barbaro, M., Burgi, P. Y., Mortara, A., Nussbaum, P., & Heitger, F. (2002). A 100 x 100 pixel silicon retina for gradient extraction with steering filter capabilities and temporal output coding. *IEEE Journal of Solid-State Circuits*, *37*, 160–172.

Bardallo, J., Gotarredona, T., & Barranco, B. (2009). A mismatch calibrated bipolar spatial contrast AER retina with adjustable contrast threshold. In *IEEE international symposium on circuits and systems* (pp. 1493–1496).

Bardallo, J., Gotarredona, T., & Barranco, B. (2011, June). A 3.6 s latency asynchronous frame-free even-driven dynamic-vision-sensor. *The IEEE Journal of Solid-State Circuits*, *46*, 1443–1455.

Barranco, F., Fermuller, C., & Ros, E. (2018). Real-time clustering and multi-target tracking using event-based sensors. *arXiv preprint arXiv:1807.02851*.

Benosman, R., Clercq, C., Lagorce, X., Ieng, S., & Bartolozzi, C. (2014). Event-based visual flow. *IEEE Transactions on Neural Networks and Learning Systems*, *25* (2), 407–417.

Benosman, R., Clercq, C., Lagorce, X., Ieng, S. H., & Bartolozzi, C. (2013). A event based visual flow. In *IEEE transactions on neural networks and learning systems*.

Binas, J., Neil, D., Liu, S., & Delbruck, T. (2017a). Ddd17: End-to-end Davis driving dataset. *ICML workshop on machine learning for autonomous vehicles.*

Binas, J., Neil, D., Liu, S. C., & Delbruck, T. (2017b). Ddd17: End-to-end Davis driving dataset. In *The international conference on machine learning*.

Brandli, C., Berner, R., Yang, M., Liu, S. C., & Delbruck, T. (2014a). A 240 180 130 db 3 us latency global shutter spatiotemporal vision sensor. *The IEEE Journal of Solid-State Circ*, *49*, 2333–2341.

Brandli, C., Berner, R., Yang, M., Liu, S. C., & Delbruck, T. (2014b). A 240x180 130 db 3 us latency global shutter spatiotemporal vision sensor. *The IEEE Journal of Solid-State Circuits*, *49*, 2333–2341.

CamunasMesa, L., SerranoGotarredona, T., Ieng, S., Benosman, R., & LinaresBarranco, B. (2018). Event-driven stereo visual tracking algorithm to solve object occlusion. *IEEE Transactions on Neural Networks and Learning Systems*, *29*(9), 4223–4237.

CamunasMesa, L., SerranoGotarredona, T., Ieng, S., Benosman, R., & Linares-Barranco, B. (2014). On the use of orientation filters for 3d reconstruction in event-driven stereo vision. *Frontiers in Neuroscience*, *8*, 48.

Cannici, M., Ciccone, M., Romanoni, A., & Matteucci, M. (2018). Event-based convolutional networks for object detection in neuromorphic cameras. *CoRR*, *abs/1805.07931*. Retrieved from http://arxiv.org/abs/1805.07931

Cao, Y., Chen, Y., & Khosla, D. (2014). Spiking deep convolutional neural networks for energy-efficient object recognition 1–13.

Censi, A., & Scaramuzza, D. (2014). Low-latency event-based visual odometry. In *IEEE international conference on robotics and automation (ICRA)*.

Clady, X., Ieng, S. H., & Benosman, R. (2015). Asynchronous event-based corner detection and matching. *Neural Networks*, *65*, 91–106.

Connor, O., Neil, D., Liu, S. C., Delbruck, T., & Pfeiffer, M. (2016). Real-time classification and sensor fusion with a spiking deep belief network. *Frontiers in Neuroscience*.

Conradt, J., Berner, R., Cook, M., & Delbruck, T. (2009). An embedded AER dynamic vision sensor for low-latency pole balancing. In *IEEE 12th international conference on computer vision workshops (ICCV workshops)* (pp. 780–785).

Conradt, J., Cook, M., Berner, R., Lichtsteiner, P., Douglas, R., & Delbruck, T. (2009). A pencil balancing robot using a pair of aer dynamic vision sensors. In *IEEE international symposium on circuits and systems (ISCAS)* (pp. 781–784).

Cook, M., Gugelmann, L., Jug, F., Krautz, C., & Steger, A. (2011). Interacting maps for fast visual interpretation. In *International joint conference on neural networks (IJCNN)* (pp. 770–776).

Dalal, N., & Triggs, B. (2005). *Histograms of oriented gradients for human detection*. CVPR.

Delbruck, J. L. T., & Pfeiffer, M. (2016). Training deep spiking neural networks using backpropagation. *Frontiers in Neuroscience*.

Delbruck, T. (2008). Frame-free dynamic digital vision. In *Proceedings of international symposium on secure-life electronics* (Vol. 1, pp. 21–26).

Delbruck, T., & Berner, R. (2010). Temporal contrast aer pixel with 0.3 threshold. In *Proceedings of 2010 I.E. international symposium on circuits and systems (ISCAS)* (pp. 2442–2445).

Delbruck, T., & Lichtsteiner, P. (2007). Fast sensory motor control based on event-based hybrid neuromorphic-procedural system. In *IEEE international symposium on circuits and systems* (pp. 845–848).

Delbruck, T., & Lang, M. (2013). Robotic goalie with 3 ms reaction time at 4 cpu load using event-based dynamic vision sensor. *Frontiers in Neuroscience*.

Diehl P, Neil D, Binas J, Cook M, Liu SC, & Pfeiffer M. (2015). Fast classifying, high-accuracy spiking deep networks through weight and threshold balancing.

Dynamic Vision Dataset. (2016). http://sensors.ini.uzh.ch/databases.html

Dynamic Vision Object Recog and Action Recog and Tracking Simulated Dataset. (2016). http://sensors.ini.uzh.ch/databases.html

Dynamic Vision Object Recog Dataset. (2016). http://sensors.ini.uzh.ch/databases.html

Dynamic Vision Object Recog Dataset. (2018). http://www.prophesee.ai/dataset-n-cars/

Dynamic Vision Object Recog Simulated Dataset. (2015). http://www2.imse-cnm.csic.es/caviar/MNISTDVS.html

Dynamic Vision RGBD Dataset. (2014). http://ci.nst.ei.tum.de/EBSLAM3D/dataset

Dynamic Vision Software. (2017). https://github.com/SensorsINI/

Dynamic Vision Software. (2018a). https://github.com/inilabs/caer

Dynamic Vision Software. (2018b). https://github.com/inilabs/libcaer

Dynamic Vision Software. (2018c). https://github.com/uzh-rpg/

Dynamic Vision Software. (2018d). https://github.com/robotology/event-driven

Farabet, C., Martini, B., Akselrod, P., Talay, S., LeCun, Y., & Culurciello, E. (2010). Hardware accelerated convolutional neural networks for synthetic vision systems. In *The IEEE international symposium on circuits and systems* (pp. 257–260).

Folowosele, F., Vogelstein, R., & EtienneCummings, R. (2011). Towards a cortical prosthesis: Implementing a spike-based hmax model of visual object recognition in silico. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, *1*(4), 516–525.

Forster, C., Pizzoli, M., & Scaramuzza, D. (2014). *Svo: Fast semi-direct monocular visual odometry* (pp. 15–22). ICRA.

Fukushima, K., Yamaguchi, Y., Yasuda, M., & Nagata, S. (1970). An electronic model of the retina. In *Proceedings of the IEEE* (Vol. 11, pp. 1950–1952). https://doi.org/10.1109/PROC.1970.8066

Gallego, G., Lund, J., Mueggler, E., Rebecq, H., Delbruck, T., & Scaramuzza, D. (2016). Event-based, 6-dof camera tracking for high-speed applications. *arXiv preprint arXiv:1607.03468.*

Gallego, G., & Scaramuzza, D. (2017). Accurate angular velocity estimation with an event camera. *IEEE Robotics and Automation Letters*, *2*(2), 632–639.

Ghosh, R., Mishra, A., Orchard, G., & Thakor, V. (2014). Real-time object recognition and orientation estimation using an event-based camera and cnn. In *IEEE biomedical circuits and systems conference* (pp. 544–547).

Glover, A., & Bartolozzi, C. (2016). Event-driven ball detection and gaze fixation in clutter. In *IEEE/RSJ international conference on intelligent robots and systems (IROS)* (pp. 9–14).

Glover, A., & Bartolozzi, C. (2017). Robust visual tracking with a freely-moving event camera. In *IEEE/RSJ international conference on intelligent robots and systems (IROS)*.

Gotarredona, S., & Barranco, L. (2015). Poker-dvs and mnist-dvs. Their history, how they were made, and other details. *Frontiers in Neuroscience*.

Gotarredona, T. S., Andreou, A., & Barranco, B. L. (1999). Aer image filtering architecture for vision processing systems. *IEEE Transactions on Circuits and Systems*, *46*(9), 1064–1071.

Gotarredona, T. S., Andreou, A., & Linarese, B. (1999). Aer image filtering architecture for vision-processing systems. *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, *46*(9), 1064–1071.

Gotarredona, T. S., & Barranco, B. L. (2013). A 128x128 1.5 asynchronous frame-free dynamic vision sensor using transimpedance amplifiers. *IEEE Journal of Solid-State Circuits*, *48*(3), 827–838.

Haessig, G. & Benosman, R. (2018). A sparse coding multi-scale precise-timing machine learning algorithm for neuromorphic event-based sensors. *CoRR*, *abs/1804.09236.* Retrieved from http://arxiv.org/abs/1804.09236

Harris, C., & Stephens, M. (1988). A combined corner and edge detector. In *Alvey vision conference*.

Hu, Y., Liu, H., Pfeiffer, M., & Delbruck, T. (2016). Dvs benchmark datasets for object tracking, action recognition, and object recognition. *Frontiers in Neuroscience*.

Illingworth, J., & Kittler, J. (1987). The adaptive hough transform. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *9*(5), 690–698.

Izadi, S., Kim, D., Hilliges, O., Molyneaux, D., Newcombe, R., Kohli, P., … Davison, A. (2011). Kinectfusion: Real-time 3d reconstruction and interaction using a moving depth camera. In *Proceedings of the 24th annual ACM symposium on User interface software and technology* (pp. 559–568).

Izhikevich, E. M. (2004). Which model to use for cortical spiking neurons? *IEEE Transactions on Neural Networks*, *15*(5), 1063–1070.

Kim, H., Handa, A., Benosman, R., Ieng, S., & Davison, A. (2014). Simultaneous mosaicing and tracking with an event camera. In *British machine vision conference*.

Kim, H., Leutenegger, S., & Davison, A. (2016). *Real-time 3d reconstruction and 6-dof tracking with an event camera*. ECCV.

Kramer, J. (2002). An on/off transient imager with event-driven, asynchronous read-out. In *IEEE international symposium on circuits and systems* (pp. 165–168).

Krizhevsky, A. (2009). *Learning multiple layers of features from tiny images.* (*MSc thesis).* University of Toronto, Department of Computer Science.

Kueng, B., Mueggler, E., Gallego, G., & Scaramuzza, D. (2016). Low-latency visual odometry using event-based feature tracks. In *IEEE/RSJ International Conference Intelligent Robots and Systems (IROS)* (pp. 16–23).

Lagorce, X., Meyer, C., Ieng, S., Filliat, D., & Benosman, R. (2015). Asynchronous event-based multikernel algorithm for high-speed visual features tracking. *IEEE Transactions on Neural Networks and Learning Systems*, *26*(8), 1710–1720.

Lagorce, X., Orchard, G., Galluppi, F., Shi, B., & Benosman, R. (2017). Hots: A hierarchy of event-based time-surfaces for pattern recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *39*(7), 1346–1359.

Lecun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient based learning applied to document recognition. In *Proceedings of the IEEE* (pp. 1178–2324).

Leutenegger, S., Lynen, S., Bosse, M., Siegwart, R., & Furgale, P. (2015). Visual-inertial slam using nonlinear optimization. *The International Journal of Robotics Research*.

Li, C., Brandli, C., Berner, R., Liu, H., Yang, M., Liu, S. C., & Delbruck, T. (2015). Design of an rgbw color vga rolling and global shutter dynamic and active-pixel vision sensor. In *IEEE international symposium on circuits and systems (ISCAS)* (pp. 718–721).

Li, J., Shi, F., Liu, W., Zou, D., Wang, Q., Lee, H., Park, P., & Ryu, H. (2017). Adaptive temporal pooling for object detection using dynamic vision sensor. British machine vision conference (BMVC), 1.

Lichtsteiner, P., Posch, C., & Delbruck, T. (2006). A 128 x 128 120db 30mw asynchronous vision sensor that responds to relative intensity change. In *IEEE international solid state circuits conference* (pp. 2004–2006).

Lichtsteiner, P., Posch, C., & Delbruck, T. (2008). A 128x128 120db 15 micro seconds latency asynchronous temporal contrast vision sensor. *IEEE Journal of Solid-State Circuits*, *566–576*.

LinaresBarranco, A., GomezRodriguez, F., Villanueva, V., Longinotti, L., & Delbruck, T. (2015). A usb3. 0 fpga event-based filtering and tracking framework for dynamic vision sensors. *IEEE International Symposium on Circuits and Systems (ISCAS)*, *26*(12), 2417–2420.

Litzenberger, M., Posch, C., Bauer, D., Belbachir, A., Schon, P., Kohn, B., & Garn, H. (2006). Embedded vision system for real-time object tracking using an asynchronous transient vision sensors. In *IEEE workshop on digital signal processing and signal processing education DSP/SPE* (pp. 173–178).

Liu, H., Moeys, D., Das, G., Neil, D., Liu, S., & Delbruck, T. (2016). Combined frame and event-based detection and tracking. In *IEEE international symposium on circuits and systems (ISCAS)* (pp. 2511–2514).

Mahowald, M. (1994). An analog vlsi system for stereoscopic vision. In *International series in engineering and computer science*.

Mallik, U., Clapp, M., Choi, E., Cauwenberghs, G., & Cummings, R. E. (2005). Temporal change threshold detection imager. In *IEEE international digest of technical papers, solid-state circuits conference* (pp. 362–364).

Maqueda, A., Loquercio, A., Gallego, G., Garcia, N., & Scaramuzza, D. (2018). *Event-based vision meets deep learning on steering prediction for self-driving cars*. CVPR.

Milford, M., & George, A. (2014). *Featureless visual processing for slam in changing outdoor environments* (pp. 569–583). Berlin, Heidelberg: Field and Service Robotics, Springer.

Milford, M., Kim, H., Leutenegger, S., & Davison, A. (2015). Towards visual slam with event-based cameras. In *The problem of mobile sensors workshop in conjunction with RSS*.

Milford, M., & Wyeth, G. (2012). Visual route-based navigation for sunny summer days and stormy winter nights. In *IEEE international conference on robotics and automation*.

Mitrokhin, A., Fermuller, C., Parameshwara, C., & Aloimonos, Y. (2018). Event-based moving object detection and tracking. *arXiv preprint arXiv:1803.04523*.

Moeys, D. P., Neil, D., Corradi, F., Kerr, E., Vance, P., et al. (2016). *Pred18: Dataset and further experiments with Davis event camera in predator-prey robot chasing*. EBCCSP.

Mueggler, E., Bartolozzi, C., & Scaramuzza, D. (2017). Fast event-based corner detection. In *Proceedings of the British machine vision conference*.

Mueggler, E., Forster, C., Baumli, N., Gallego, G., & Scaramuzza, D. (2015). Lifetime estimation of events from dynamic vision sensors. In *IEEE international conference on robotics and automation (ICRA)*.

Mueggler, E., Gallego, G., Rebecq, H., & Scaramuzza, D. (2018). Continuous-time visual-inertial odometry for event cameras. *IEEE Transactions on Robotics*, 1–16.

Mueggler, E., Gallego, G., & Scaramuzza, D. (2015). Continuous-time trajectory estimation for event-based vision sensors. In *Robotics: Science and systems XI (RSS)*.

Mueggler, E., Huber, B., & Scaramuzza, D. (2014). Event-based, 6-dof pose tracking for high-speed maneuvers. In *IEEE/RSJ international conference on intelligent robots and systems (IROS)* (pp. 2761–2768).

Mueggler, E., Rebecq, H., Gallego, G., Delbruck, T., & Scaramuzza, D. (2017). The event-camera dataset and simulator: Event-based data for pose estimation, visual odometry, and slam. *International Journal of Robotics Research*, *36*, 142–149.

Neil, D., Pfeiffer, M., & Liu, S. (2016). Learning to be efficient: Algorithms for training low-latency, low-compute deep spiking neural networks. In *Proceedings of the 31st annual ACM symposium on applied computing*.

Newcombe, R., Lovegrove, S., & Davison, A. (2011). Dtam: Dense tracking and mapping in real-time. In *International conference on computer vision (ICCV)* (pp. 2320–2327).

Nguyen, A., ThanhToan, D., Darwin, G., & Nikos, G. (2017). Real-time pose estimation for event cameras with stacked spatial lstm networks. *arXiv preprint arXiv: 1708.09011*.

Ni, Z., Ieng, S., Posch, C., Regnier, S., & Benosman, R. (2015). Visual tracking using neuromorphic asynchronous event-based cameras. *Neural Computation*, *27*(4), 925–953.

Ni, Z., Pacoret, C., Benosman, R., Leng, S., & Regnier, S. (2012). Asynchronous event-based high speed vision for microparticle tracking. *Journal of Microscopy*, *245* (3), 236–244.

Orchard, G. (2015). Hfirst: A temporal approach to object recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *37*(10), 2028–2040.

Orchard, G., Jayawant, A., Cohen, G. K., & Thakor, N. (2015). Converting static image datasets to spiking neuromorphic datasets using saccades. *Frontiers in Neuroscience*.

Perez-Carrasco, J., Zhao, B., Serrano, C., Acha, B., SerranoGotarredona, T., Chen, S., & Linares-Barranco, B. (2013). Mapping from frame- driven to frame-free event-driven vision systems by low-rate rate coding and coincidence processing-application to feedforward convnets. *Pattern Analysis and Machine Intelligence*, *35*(11), 2706–2719.

Perez-Carrasco, J. A., Zhao, B., Serrano, C., & Chen, S. (2013, Nov). Mapping from frame-driven to frame-free event-driven vision systems by low-rate rate coding and coincidence processing application to feedforward convnets. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, *35*(11), 2706–2719.

Piatkowska, E., Belbachir, A., Schraml, S., & Gelautz, M. (2012). Spatiotemporal multiple persons tracking using dynamic vision sensor. In *IEEE conference on computer vision and pattern recognition* (pp. 35–40).

Pizzoli, M., Forster, C., & Scaramuzza, D. (2014). Remode: Probabilistic monocular dense reconstruction in real time. In *IEEE international conference on robotics and automation (ICRA)*.

Posch, C., Hoffstaetter, M., & Schoen, P. (2010). A sparc-compatible general purpose address-event processor with 20-bit 10ns-resolution asynchronous sensor data interface in 0.18um cmos. In *IEEE international symposium on circuits and systems (ISCAS)*.

Posch, C., Matolin, D., & Wohlgenannt, R. (2011). 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*, *46*(1), 259–275.

Posch, C., Matolin, D., Wohlgenannt, R., Maier, T., & Litzenberger, M. (2009). A microbolometer asynchronous dynamic vision sensor for lwir. *IEEE Sensors Journal*, *9*.

Rebecq, H., Gallego, G., & Scaramuzza, D. (2016). *Emvs: Event-based multi-view stereo*. BMVC.

Rebecq, H., Horstschaefer, T., Gallego, G., & Scaramuzza, D. (2017). Evo: A geometric approach to event-based 6-dof parallel tracking and mapping in real-time. *IEEE Robotics and Automation Letters (RA-L)*, 593–600.

Rebecq, H., Horstschaefer, T., & Scaramuzza, D. (2017). Real-time visual-inertial odometry for event cameras using keyframe-based nonlinear optimization. In *British machine vision conference (BMVC)*.

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 779–788).

Reinbacher, C., Graber, G., & Pock, T. (2016). Real time intensity-image reconstruction for event cameras using manifold regularisation. In *British machine vision conference (BMVC)*.

Reinbacher, C., Munda, G., & Pock, T. (2017a). Real-time panoramic tracking for event cameras. *arXiv preprint arXiv:1703.05161*.

Reinbacher, C., Munda, G., & Pock, T. (2017b). Real-time panoramic tracking for event cameras. In *International conference on computational photography*.

Rosten, E., & Drummond, T. (2006). Machine learning for high-speed corner detection. In *European Conference on Computer Vision (ECCV)* (pp. 430–443).

Rueckauer, B., & Delbruck, T. (2015). Evaluation of algorithms for normal optical flow from dynamic vision sensors. *Frontiers in Neuroscience*.

Rueckauer, B., & Delbruck, T. (2016). *Evaluation of event-based algorithms for optical flow with ground-truth from inertial measurement sensor*. FNINS.

Ruedi, P. F., Heim, P., Gyger, S., Kaess, F., Arm, C., et al. (2009). An soc combining a 132db qvga pixel array and a 32b dsp/mcu processor for vision applications. In *IEEE international solid-state circuits conference* (pp. 47–48).

Schraml, S., & Belbachir, A. (2010). A spatio-temporal clustering method using real-time motion analysis on event-based 3D vision. In *IEEE conference on computer vision and pattern recognition*.

SerranoGotarredona, T., & Linares-Barranco, B. (2015). Poker-dvs and mnist-dvs. Their history, how they were made, and other details. *Frontiers Neuroscience*, *9*.

Simard, P., Steinkraus, D., & Platt, J. C. (2003). Best practices for convolutional neural networks applied to visual document analysis. In *Seventh international conference on document Analysis and recognition, 2003. Proceedings* (pp. 958–963).

Sironi, A., Brambilla, M., Bourdis, N., Lagorce, X., & Benosman, R. (2018a). HATS: Histograms of averaged time surfaces for robust event-based object classification. *CoRR*, *abs/1803.07913*. Retrieved from http://arxiv.org/abs/1803.07913

Sironi, A., Brambilla, M., Bourdis, N., Lagorce, X., & Benosman, R. (2018b). Hats: Histograms of averaged time surfaces for robust event-based object classification. In *IEEE conference computer vision and pattern recognition (CVPR)*.

Son, B., Suh, Y., et al. (2017). A 640 480 dynamic vision sensor with a 9m pixel and 300meps address-event representation. In *IEEE international solid-state circuits conference (ISSCC)* (pp. 66–67).

Source Code for Event Lifetime. (2017). https://github.com/uzh-rpg/rpg_event_lifetime

Source Code for Image Reconstruction. (2017a). https://github.com/uzh-rpg/rpg_image_reconstruction_from_events

Source Code for Image Reconstruction. (2017b). https://github.com/VLOGroup/dvs-reconstruction

Source Code for Localization. (2016). https://github.com/VLOGroup/dvs-panotracking

Source Code for Object Recognition. (2016). http://www.garrickorchard.com/code

Source Code for Optical Flow. (2017a). https://sourceforge.net/p/jaer/code/HEAD/tree/jAER/trunk/src/ch/unizh/ini/jaer/projects/rbodo/opticalflow/LocalPlanesFlow.java

Source Code for Optical Flow. (2017b). https://sourceforge.net/p/jaer/code/HEAD/tree/jAER/trunk/src/ch/unizh/ini/jaer/projects/rbodo/opticalflow/

Stromatias, E., Soto, M., SerranoGotarredona, T., & LinaresBarranco, B. (2017). An event-driven classifier for spiking neural networks fed with synthetic or dynamic vision sensor data. *Frontiers in Neuroscience*.

Tedaldi, D., Gallego, G., Mueggler, E., & Scaramuzza, D. (2016). Feature detection and tracking with the dynamic and active-pixel vision sensor (Davis). In *IEEE international conference on event-based control, communication, and signal processing (EBCCSP)*.

Tumblin, J., Agrawal, A., & Raskar, R. (2005). Why I want a gradient camera. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*.

Valeiras, D., Lagorce, X., Clady, X., Bartolozzi, C., Ieng, S., & Benosman, R. (2015). An asynchronous neuromorphic event-driven visual part-based shape tracking. *IEEE Transactions on Neural Networks and Learning Systems*, *26*(12), 3045–3059.

Vasco, V., Glover, A., & Bartolozzi, C. (2016). Fast event-based Harris corner detection exploiting the advantages of event-driven cameras. *Intelligent Robots and Systems (IROS)*.

Vasco, V., Glover, A., Mueggler, E., Scaramuzza, D., Natale, L., & Bartolozzi, C. (2017). Independent motion detection with event-driven cameras. In *International conference on advanced robotics (ICAR)* (pp. 530–536).

Vidal, A., Rebecq, H., Horstschaefer, T., & Scaramuzza, D. (2018). Ultimate slam? Combining events, images, and imu for robust visual slam in hdr and high speed scenarios. *IEEE Robotics and Automation Letters*, *3*(2), 994–1001.

Weikersdorfer, D., Adrian, D., Cremers, D., & Conradt, J. (2014a). Event-based 3D slam with a depth-augmented dynamic vision sensor. In *IEEE international conference on robotics and automation (ICRA)*.

Weikersdorfer, D., Adrian, D. B., Cremers, D., & Conradt, J. (2014b). Event-based 3d slam with a depth-augmented dynamic vision sensor. In *International conference on robotics and automation*.

Weikersdorfer, D., & Conradt, J. (2012). Event-based particle filtering for robot self-localization. In *IEEE international conference on robotics biomimetics* (pp. 866–870).

Weikersdorfer, D., Hoffmann, R., & Conradt, J. (2013). Simultaneous localization and mapping for event-based vision systems. In *International conference computer vision systems (ICVS)* (pp. 133–142).

Yang, M., Liu, S. C., & Delbruck, T. (2015). A dynamic vision sensor with 1asynchronous delta modulator for event encoding. *IEEE Journal of Solid-State Circuits*, *50*, 2149–2160.

Zhu, A., Daniilidis, K., & Atanasov, N. (2017). Event-based visual inertial odometry. In *IEEE conference computer vision and pattern recognition (CVPR)* (pp. 5816–5824).

Zhu, A., Thakur, D., Ozaslan, T., Pfrommer, B., Kumar, V., & Daniilidis, K. (2018). The multi vehicle stereo event camera dataset: An event camera dataset for 3d perception. *IEEE Robotics and Automation Letters*.

How to cite this article: Lakshmi A, Chakraborty A, Thakur CS. Neuromorphic vision: Sensors to event-based algorithms. *WIREs Data Mining Knowl Discov*. 2019;e1310. https://doi.org/10.1002/widm.1310