Journal of Real-Time Image Processing (2022) 19:363–374
https://doi.org/10.1007/s11554-021-01191-y

ORIGINAL RESEARCH PAPER

Event camera simulator design for modeling attention-based inference architectures

Md Jubaer Hossain Pantho[1] · Joel Mandebi Mbongue[1] · Pankaj Bhowmik[1] · Christophe Bobda[1]

Received: 18 May 2021 / Accepted: 14 December 2021 / Published online: 5 January 2022

Abstract
In recent years, there has been a growing interest in realizing methodologies to integrate more and more computation at the level of the image sensor. The rising trend has seen an increased research interest in developing novel event cameras that can facilitate CNN computation directly in the sensor. However, event-based cameras ca be expensive, limiting performance exploration on high-level models and algorithms. This paper presents an event camera simulator that can be a potent tool for hardware design prototyping, parameter optimization, attention-based innovative algorithm development, and benchmarking. The proposed simulator implements a distributed computation model to identify relevant regions in an image frame. Our simulator's relevance computation model is realized as a collection of modules and performs computations in parallel. The distributed computation model is configurable, making it highly useful for design space exploration. The Rendering engine of the simulator samples frame-regions only when there is a new event. The simulator closely emulates an image processing pipeline similar to that of physical cameras. Our experimental results show that the simulator can effectively emulate event vision with low overheads

Keywords Simulator · Convolutional neural network · Embedded vision · Pixel processing

1 Introduction

Event cameras are bio-inspired vision sensors designed to generate image frames asynchronously based on scenic events [1]. In contrast to conventional camera sensors where raw frame pixels are streamed to a backend processor at a fixed rate, event-based cameras generate output only when there is a new event(s). Recently, researchers are seeking novel methodologies to incorporate machine learning models (in particular CNNs) in the image sensor [2, 3]. This has revived interest in event cameras to facilitate efficient dataflow between the sensor and the near-sensor processing system. However, novel algorithms and methods are required to process the unorthodox data streams from these vision sensors to unlock their full potential [4]. However, researchers working on this domain face two major challenges. First, there are not sufficient event-cameras in the market, limiting the research to a few applications. Second, the commercially available event cameras suffer from different setbacks such as low resolution, lack of reconfiguration, etc.

Several camera simulators have been proposed in the literature to accommodate the research demands [5, 6]. For instance, authors in [5] presented ESIM, a camera simulator that resembles an event camera's behavior. The simulator integrates an adaptive rendering scheme that only samples frames when necessary. In addition to generating events, the simulator can produce a depth map, motion field, and camera trajectory. However, the simulator was developed for robotics applications and not specifically designed to explore inference architectures near the sensors. Therefore, any in-sensor high-level processing engine that aims to leverage the event sensor in the processing pipeline will fail to utilize the full potential of the events generated from this camera simulator. At best, the simulator would allow the inference engine only to activate whenever a new event is

✉ Md Jubaer Hossain Pantho
mpantho@ufl.edu

Joel Mandebi Mbongue
jmandebimbongue@ufl.edu

Pankaj Bhowmik
pankajbhowmik@ufl.edu

Christophe Bobda
cbobda@ece.ufl.edu

[1] University of Florida, Gainesville, USA

detected on the sensor interface. However, at each iteration, the full image will get processed in the inference engine regardless of the size of the ROI (Region-Of-Interest). The newest developments in imaging technology have brought forth parallel processing image sensors that can be combined with an inference engine to provide high-performance computation models near the sensor [1, 7–9]. By tightly coupling computation on the inference layer to specific image regions, it is possible to improve the computational capabilities of these systems and reduce data communications. Nevertheless, a suitable platform is required to explore the design space of these architectures.

In this paper, we present a novel event camera simulator that simulates a per-pixel image sensor's behavior aiming to accommodate CNN inference in the sensor interface. The events captured in the simulator are identified on a region level. Therefore, only specific regions can be forwarded to the following computation layer to activate the inference engine minimally (Fig. 1). Similar to the work mentioned above, our rendering-module samples image frames whenever there is a new event. However, instead of sampling the complete image, respective event regions are only sampled. The simulator can generate valid event data from a video stream that can be used to model and train event-based learning models. We have prototyped the simulator's computation module on an FPGA to estimate the hardware cost. Our evaluation results suggest that we can significantly reduce computation with our event-based camera approach with decent hardware overhead.

The main contributions of this paper are:

- A novel camera simulator design that identifies regional events and facilitates a suitable interface for inference architectures.
- A thorough evaluation of our region-level relevance computation model to highlight the significance.
- An FPGA prototype of the relevance computation model to indicate hardware overheads related to our approach.

The main motivation of this work is to design a novel simulator that identifies relevant data on a regional level. The work aims to generate custom attention-based datasets that can be used to jointly consider algorithm–hardware co-design frameworks to address computational overheads. The proposed simulator can be used to explore the architecture design space of inference engines that uses tiling-based operations on image data. We have released our design open source [10]. The remaining sections of this paper are organized as follows. Section 2 discusses the related works in the literature. Section 3 provides a detailed explanation of our design. We evaluate the performance of our model in Sect. 4.

2 Related work

In recent years, vision-based algorithms are increasingly being used in different application domains to solve different complex problems [11, 12]. The primary task in these applications includes identifying objects and recognizing them. In many cases, it is always beneficial to narrow down the computation to a specific region by utilizing an attention-based model. Several camera simulators can be found in the literature emulating the behavior of an event camera [13–15]. And, in recent years, various approaches have been proposed to bring inference computation close to the sensor. We start by studying the state-of-the-art camera simulator and highlight the advantages of our proposed toolchain. Next, we will discuss the in-sensor processing architectures that leverage event-based camera designs.

In [16], authors present an event sensor simulator that can render events from a 3D scene. The simulator was designed to facilitate research in robotic vision. However, it is not tailored for in-sensor processing exploration. The virtual camera proposed in [17] offers an interactive interface with a custom rendering engine that can be used for benchmarking different SLAM algorithms. Similar to previous work, here, the authors did not illustrate the use cases with inference architectures but focused on generating photo-realistic indoor scenes datasets.

We found ESIM as one of the thorough works on event camera simulation [5]. It provides an open-source design and illustrates use cases on learning optical flow. However, ESIM (including all the other works described above) identifies events at a pixel level. These fine-grained events captured in the sensor interface can reduce the rendering engine's workload. However, with the current setup, any subsequent inference engine in the processing pipeline

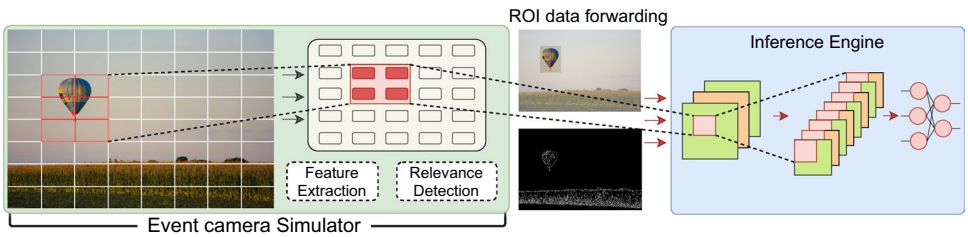

Fig. 1 Region-based event camera simulator designed to accommodate inference processing near the sensor

will not be able to leverage many benefit from these fine-grained events due to the available dataflow mechanisms.

There are other recent simulators that operate on DVS events [18, 19]. These works present simulators to generate realistic DVS events that can be useful for training networks. Our work differs from these works by considering both spatial and temporal features to identify relevant regions.

The ReImagine program launched by DARPA aims to integrate revolutionary capabilities in the imaging system [20]. They demonstrated that a single, reconfigurable ROIC (ReadOut Integrated Circuit) architecture could accommodate multiple modes of imaging operations that may be defined after a chip has been designed. The program seeks ROI-based efficient computation models to enable real-time analysis. Even though preliminary works have shown promising results, the landscape of the high-level computation part is still in progress. Further development in this research direction faces setbacks due to the lack of appropriate physical cameras that can accommodate these operations.

Other works in accommodating CNNs in an image sensor involve coupling an array of pixel processors to a parallel processing camera [21, 22]. Authors in [21] proposed a region-aware processing model to reduce high-level computation to relevant regions. However, the authors mainly discussed the hardware aspects of the architecture. Whereas, it is essential to thoroughly assess the behavior of region-aware processing models for different applications. For instance, the methodologies and threshold values used to identify relevant image regions can differ for different scenarios.

Our simulator design differs from the works mentioned above by considering the CNN computation models that operate on the sensor's collected data. The approaches found in the literature provide solutions at best for generic use cases. Our proposed simulator emulates event cameras that capture changes at a regional level as opposed to pixel-level sampling. This allows the subsequent computation layers to minimize computation on irrelevant regions. We believe our simulators will enable researchers to develop optimized attention-based hardware architectures by accurately analyzing the relevance model. Besides, the configurability of the simulator allows exploration of the design space for event cameras.

3 Proposed design flow

In this section, we first describe the concept and the principles of operation of the event camera that we simulate. Then, we illustrate the design flow and architecture of the simulator.

3.1 Camera model

Our virtual camera's baseline design considers a parallel imager, where each sensing unit in the photodiode array has an analog to digital converter (ADC) and a local memory [8]. At the sensor interface, the incoming image frame is logically divided into M image regions where $N \times N$ pixels reside in each image patch (shown in Fig. 1). There is a regional processing unit (RPU) for each image patch for the local handling of computation. Each RPU has one streaming channel to transfer pixel/event data from the corresponding region to the next buffer (or computation module). All RPUs operate independently and generate output in parallel. Within the RPU, the saliency data for the corresponding region are computed. A saliency score is calculated to reflect the spatial and temporal relevance of that region. Based on the saliency score, only specific image regions are forwarded to the next plane to enable attention-based near-sensor computation. Depending on the application, the high-level processing module can initiate computation for separate regional events or extract information from the accumulated events in an image frame. The proposed camera model generates attention-based image data by computing the events captured at the sensor interface. Here, the actual image pixels are forwarded to the higher processing plane to allow the inference processing modules to operate on a reduced set of pixel data. Interested readers can look at [23, 24], to learn more about similar camera models.

3.2 Simulator architecture

The difference between a conventional camera and an event camera is the latter does not capture intensity information from the scene synchronously. Instead, it samples visual signals asynchronously and independently for each pixel/region. In our design, we simulate this behavior with a regular vision system. The simulator's input is a stream of image frames from a camera or video clip captured at discrete time intervals. Whereas the output of the simulator includes localized pixel and event information generated at irregular intervals. The simulator comprises a capture module, a relevance computation module(RCM), and a rendering module. The high-level simulator architecture is shown in Fig. 2. The capture module collects image frames at a regular interval, divides the image frame into equal-sized image patches, and forwards them to the RCM. The RCM comprises an array of RPUs operating in parallel. Within the RPU, saliency scores are computed. The saliency scores are calculated based on spatial and temporal information. Visual attention can be drawn from different details embedded within the image pixels (i.e., edges, corners, motion, error surface, optical flow, data distribution). If the saliency score is greater than some threshold, then that region is identified as relevant.

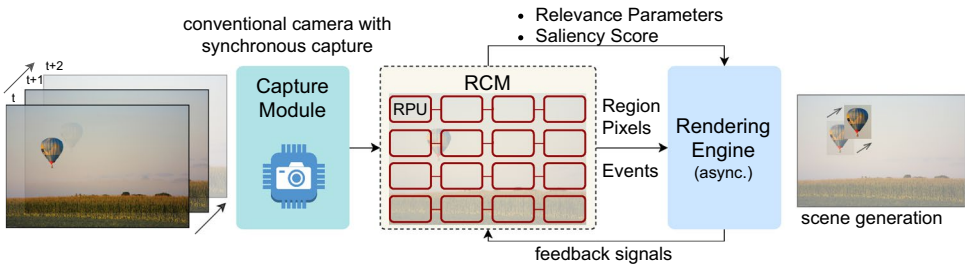

Fig. 2 Proposed Simulator model. The capture module propagates image regions to the RCM. The RCM computes the relevance and feeds an asynchronous rendering engine to generate event-based frames

The renderer collects data from the RCM and constructs the image frame for the high-level processing units in the image processing pipeline. This includes raw pixel data, saliency score, and other feature information calculated to identify the region of interest (ROI). The rendering engine renders an image at time t based on the events captured at time t interval and the renderer's previous state at time $t-1$. Therefore, if we denote the renderer output as R, it can be written as:

$$R(t) = R_{ROI_{spatial}}(t-1) + R_{ROI_{temporal}}(t). \quad (1)$$

The next section details the relevance computation model utilized in our simulator.

3.3 Relevance computation module (RCM)

An image processing pipeline with a vision sensor and a high-level back-end processor imitates the eye and brain's combined functionality. Except, a human eye has around 130 million pixels, with only 1.3 million synaptic connections to the brain, indicating a 1% sparsity [20]. It is believed that the massive sparsity is essential for power and latency trade space and helps avoid sending repetitive information to the latter parts of the brain. The RCM of our simulator is designed to emulate the behavior of the biological vision system. This means that the RCM will receive a large number of incoming pixels from the sensor interface and forward a limited number of pixels from specific image regions to the higher processing module. The RPUs in this module operate on a region-parallel basis. The RPU performs the relevance function on image pixels and accumulates the relevance score for all pixels in a region. The spatial relevance score can be calculated from a set of indexes based on the user-defined environment (i.e., edge, corners, variance, segmentation, etc.). For instance, if we consider edge points as a spatial relevance index, we count the number of edge points found in an image region. Then, we use this value to rank the image regions based on a predefined threshold. Likewise, to check the spatial data distribution, the RPU can calculate the mean absolute deviation and classify the image regions based on data variation in a similar manner. Our proposed simulator implements a number of spatial relevance detection functions, from where the user can select the

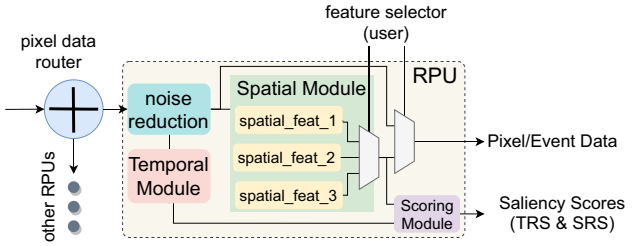

Fig. 3 RPU block diagram. Here, spatial_feat_i indicates feature indexes used to identify spatially relevant regions (i.e., edges, corners, optical flow, etc.)

Table 1 Computation based on the Relevance score

TRS	SRS	RPU	Rendering engine output
1	1	Active	Driven by current state
0	1	Inactive	Driven by previous state
(0/1)	0	Inactive	Forced to Zero/previous state

appropriate method that best suits a given scenario/dataset. The functionality of the RPU is shown in Fig. 3. Here, the noise reduction module is used to remove noise and interference from the incoming image region. It helps to reduce the miss-detection of events.

For temporal saliency, RPUs compare the incoming pixel to its temporal neighbors. This means the temporal relevance is computed by comparing the incoming pixel to the existing pixel. The number of temporal mismatches within a region is compared against a temporal threshold value to determine temporal relevance. If this value exceeds a specific value, then we mark that region as temporally relevant. The threshold value can be adjusted on the simulator to find the optimum computation point. The image patches are categorized using two-bit information, each for spatial and temporal saliency. This information is forwarded to the rendering engine that requests data from the RCM module based on the relevance score. The operation of the rendering engine based on the relevance score is shown in Table 1.

In Table 1, the TRS value indicated temporal relevance score, whereas the SRS value refers to the spatial relevance score. The active notion in RPU implies that for a given

input frame, new image data are forwarded to the rendering engine from that RPU.

As it can be inferred, the attributes of ROI depend on the threshold value used in the relevance computation algorithm. Here, the threshold values are achieved empirically. As mentioned earlier, spatial relevance can be drawn from different parameters. And it is understood that for different application/feature combinations, the threshold value will be different. It can be tuned by observing the histogram analysis of the feature points. The formula to measure spatial relevance score (SRS) is shown in Eq. 2. When we select edges as the spatial feature index, in Eq. 2, $feat_{value}$ will be the number of edge points found in an RPU region.

$$SRS = \begin{cases} 1, & feat_{value} > Threshold \\ 0, & feat_{value} \leq Threshold. \end{cases} \tag{2}$$

Please note that, for different datasets, the size of the object can vary widely. For example, in small object-detection applications, using a smaller region size for computation can be beneficial. Whereas, for larger objects, the opposite is true. Each RPU in the computational plane can be designed to operate in parallel. Therefore, the latency of the computation plane will depend only on a single RPUs task completion time. A larger region size indicates that there will be a greater number of pixels and will have higher latency. However, the total number of RPUs in the plane will decrease, contributing to lesser resources. It is a trade-off that needs to be resolved at the design stage. Therefore, the size of the RPU in this method is a design choice, and our simulator can assist in finding the optimum region size for different datasets.

The rendering engine in our proposed simulator model refreshes the image frame asynchronously. What this means is, it does not update the rendered image frame on each incoming frame. Instead, it waits for the RCM layer to send in the saliency score and relevance values. Depending on the saliency score, the new pixel regions are requested from the RCM. Now, for different applications that operate on the output rendering engine, it is possible that the current relevance parameters (threshold) are not yielding ideal results. In that case, the simulator allows a feedback signal to be sent back to the RCM layer to adjust the threshold value.

3.4 Pixel-level relevance vs region-level relevance

As discussed above, we identify important events in our simulator on a regional level. This indicates that we label image patches with a relevance score and not individual pixels. The approach is in contrast with popular methods where events are detected on a pixel basis. For instance, the ESIM simulator detects events on a pixel basis and estimates based on motion, optical flow, depth, and other indexes [5].

We opted for a different approach because we found that a single isolated pixel-event propagated to the subsequent processing units does not provide any high-level knowledge inference. Here, we would like to highlight that high-level knowledge is inferred with machine learning algorithms in almost all image processing pipelines. And CNNs are the most popular among them. In CNNs, identical window-based operations are performed on each input feature point at each convolutional layer. The common approaches to carry out convolution on CNN accelerators include systolic array operations or vectored window operations. In both cases, even if we narrow down our calculation to each new eventful pixel, the dataflow mechanism will limit the accelerator's ability to maximize the performance based on the fine-grained events. The limitation for the dataflow mechanism indicates that, for every isolated pixel event, a CNN accelerator engine will have to load all the neighboring pixels corresponding to the convolution window to operate. The dataflow mechanism in contemporary acceleration engines cannot minimize the redundant data loading to the inference engine from pixel events. Besides, not every isolated pixel event corresponds to a relevant event. It is not possible to tell from a pixel's perspective. This can lead to redundant operations in the engine. In other words, the inference module will not be able to leverage the fine-grained events generated at the pixel level. Whereas with our region-level saliency detection approach, a carefully designed inference engine can localize the computation, and any new events will initiate computation only in a specific region using a vectored window operation. Besides, it is possible to opt out calculation on isolated pixel events residing in low-scoring image patches by adequately calibrating the event camera. We found that the pruning of redundant regions has a minimum to no impact on the accuracy of the inference model. Moreover, the approach can improve the performance of sparsity-aware models by eliminating computational redundancies from the processing pipeline. For instance, authors in [25] presented a CNN-based tiny object detection mechanism that schedules image patches to a classifier and a detector to identify objects. Here, our simulator can reduce the computation by eliminating redundant image patches early at the sensor interface. Besides, in [24], authors schedule image tiles in their accelerator architecture to perform CNN operations. The output of our simulator tags each image region with its saliency score. Therefore, by adequately eliminating low-scoring regions, our event camera model can be utilized in tile-based accelerators to improve computational efficiency.

3.5 Configurability

The benefit of our simulator is that it allows camera parameter reconfigurations for different applications. We understand that the size of the regions, the spatial relevance

index, and the threshold values dictating the saliency may differ for different application scenarios. Therefore, the simulator enables users to set up these environment parameters to generate custom event-based datasets that can be later used to develop and train region-aware inference models.

The design flow of our simulator is shown in Fig. 4. The simulator takes in a conventional stream of images or image datasets as input and generates region-based events based on user-specified region size and relevant functions. By analyzing the generated events and observing the data distribution, it is possible to calibrate the user-defined parameters to fine-tune the captured events. The simulator's output will be a custom image dataset of contiguous events that can facilitate the training of inference models for high-level computation.

4 Results

In this section, we detail our evaluation infrastructure and provide experimental results to indicate the efficacy of our design.

4.1 Evaluation infrastructure

Our proposed simulator computes Spatio-temporal relevance to detect regions with events. However, to better evaluate the impact of the relevance function, we test the spatial and temporal modules separately for different datasets. The goal of this evaluation is to quantify the influence of our region-based relevance model. Next, we assess the effect of the region size and threshold values in our approach. Then, we try to evaluate the change in accuracy for different CNN models when trained on our event-driven datasets. Finally, we prototype the RCM module on an FPGA to estimate the resource overhead of our model to evaluate the viability of realizing it at the edge. We end our evaluation by comparing our simulator with other event-based simulators found in the literature.

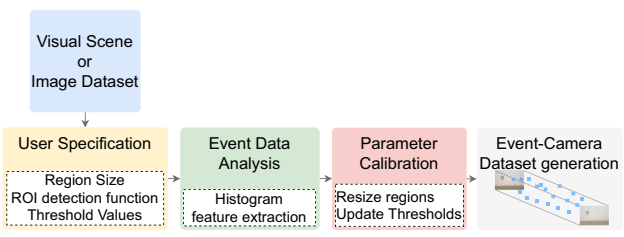

Fig. 4 Simulator design flow. The output of the simulator will be a custom dataset that is similar to the images generated by the event camera described in Sect. 3.1

4.2 Evaluation details

In our evaluation, we aim to show that the relevance computation model can successfully identify the relevant regions. We demonstrate this by showing the performance of the classifier/detector models operating on the generated custom dataset. The high accuracy of the ROI-extracted model indicates that the simulator is well equipped to identify relevant regions. Besides, we want to show how our simulator can be utilized to resolve design choices related to embedded hardware. Our result suggests that an ideal RPU size can be crucial to identify the optimum tradeoff point.

The proposed simulator is written in Python scripting language. For the inference modeling performed in this evaluation, we used the PyTorch framework. For this evaluation, we used different image datasets as the simulator's input and generated custom event-driven datasets with a reduced amount of data. For noise reduction, we used median filtering on incoming images. However, other noise reduction mechanisms can also be used. For spatial relevance detection, we implemented three feature indices within the RPU: edge, corner, and mean absolute deviation (MAD). While edges and corners provide the locality of early feature points within an image frame, the MAD value gives an insight into the statistical distribution of the region data. The edge and corner points are common feature indexes used to draw ROI in an image. Therefore, we will emphasize our evaluation of the spatial distribution of the data. Here, we chose 'mean absolute deviation' over variance due to their implementation's hardware cost. The equation for calculating variance is shown in Eq. 3.

$$\sigma^2 = \frac{\sum_{i=1}^{n}(x_i - \mu)^2}{n}. \tag{3}$$

Here, μ represents the mean value. Here, the square operation consumes considerable hardware resources. In contrast, MAD computation does not require square operation and has minimum hardware overhead. MAD is shown in Eq. 4.

$$MAD = \frac{\sum_{i=1}^{n}|x_i - \mu|}{n}. \tag{4}$$

To evaluate the effectiveness of mean absolute deviation, we first analyze the data distribution of different datasets. For this experiment, we selected four different datasets: MNIST, FashionMNIST, CIFAR10, and MOT17-08. For the first three datasets, the image size is 32×32, and the region size is selected to be 4×4. Whereas, for MOT-17 dataset, image

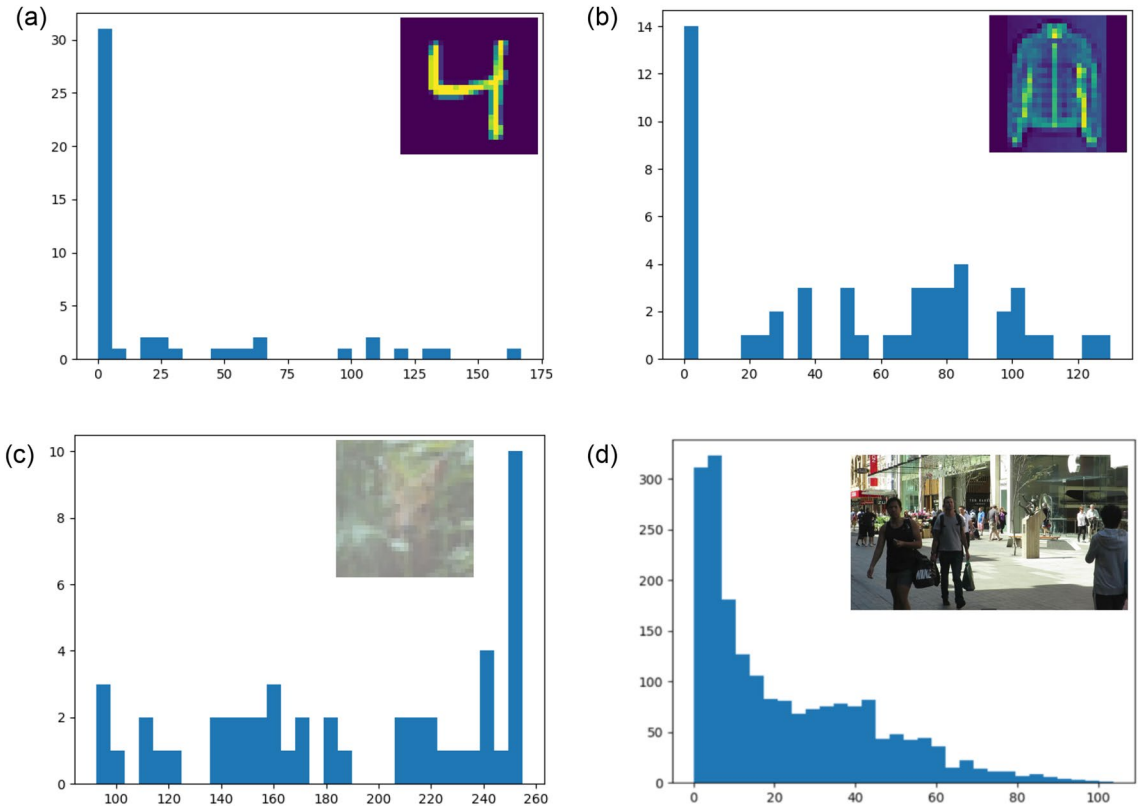

Fig. 5 Distribution of mean absolute deviation. For **a**, **b**, and **c**, images are divided into 4×4 patches. In image **d**, region size of 32×32 is used. **a** MNIST, **b** FashionMNIST, **c** CIFAR10 datasets, **d** MOT17-08

Table 2 Region-level temporal relevance analysis on MOT17 datasets

Dataset	Description	Avg. ROI (%)
MOT17-08	Pedestrian street (static cam)	41.60
MOT17-03	Sidewalk at night (static)	25
MOT17-01	Busy square (static)	28.29
MOT17-12	Shopping mall (moving cam)	69.43

resolution is 1920×1080 and we opted for a region size of 32×32. Figure 5 illustrates some sample results. As we can see, for datasets (a), (b), and (d), there are a large number of regions with a MAD value close to 0. However, for image (d), this is not the case. Because in CIFAR10, the foreground to background pixel ratio is very high, and the chosen region size is comparable to the actual image size.

4.3 Temporal relevance analysis

Next, we seek to estimate the typical size of the ROI detected by the temporal module of the simulator. For this evaluation, we used the MOT17 datasets for a real-world scenario [26]. The dataset contains different video clips of people moving in public places. The video clips are captured with a 30fps camera with an image resolution of 1920×1080. We tested our simulator on four different MOT17 datasets. Table 2 indicates the mean percentage of non-relevant regions for each dataset. The table indicates that more than 50% of the regions contains repetitive regions over time for static camera positions. For region-level relevance detection, it is possible to reduce a more significant amount of redundancies by carefully selecting the threshold value. Here, regions with insignificant temporal changes can be discarded from the computation. However, we notice that, for the 4th entry in the table, we have a comparatively lower number of irrelevant regions due to the moving camera position. Therefore, for moving camera systems, spatial redundancy reduction techniques can be used for further improvement. The results in Table 2 further confirm the spatiotemporal redundancy reduction technique used in our simulator.

Figure 6 provides a pictorial view of the event-based outputs generated by our simulator for the MOT17 dataset. The graphs in Fig. 6 indicate the average percentage of ROI regions over time.

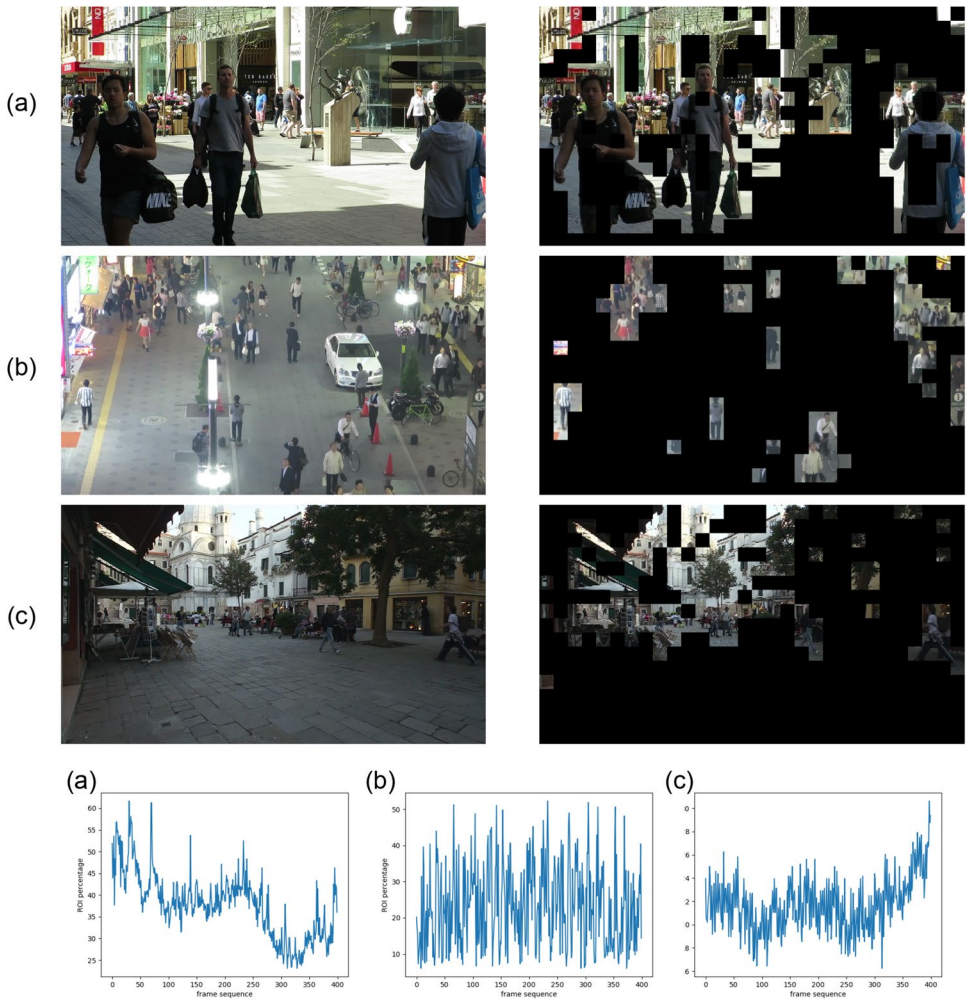

Fig. 6 Region-level temporal relevance. Left column indicating original image. The second column illustrates temporal ROIs. The right column shows percentage of ROI region size over time. **a** MOT17-08, **b** MOT17-03, **c** MOT17-01

Table 3 Region-level spatial relevance analysis

Dataset	Image size	Avg. redundancy (%)
MNIST	28×28	50
FashionMNIST	28×28	29
OpenImages (airplane)	224×224	31
Mosquito	224×224	40

4.4 Spatial relevance analysis

We perform a similar study for spatial relevance detection on different datasets. We selected four datasets for this study: MNIST, FashionMNIST, OpenImages [27] and mosquito species [28]. For the OpenImages dataset, we tested our simulator only on the airplane class due to the low foreground to background ratio on airplane images. It is easily understood that a high background ratio indicates the existence of a higher amount of redundancy. The average rates of spatially redundant regions in these datasets are shown in Table 3. Here, the images are resized before passing them through the simulator. As the table indicates, all four datasets contain spatial redundancies that can be removed using our event-camera simulator.

Figure 7 provides a pictorial view of the ROI detected images shown in Table 3. For different datasets, region sizes are adjusted for optimal results. Here, we would like to emphasize again that the average redundancy found in these datasets is dependent on the threshold values and the region size selected for them. We have used databases of different image sizes to demonstrate the adaptive capabilities of our proposed simulator. It is possible for our simulator to automatically select the region size based on the default granularity parameters and the permissible ROI percentage of the image. The granularity parameters include the maximum and minimum ratio of the patch size to the original image. With an iterative approach, the simulator can automatically choose the optimum region size that meets the allowed ROI percentage set by the user. Once the simulator computes a range of region sizes,

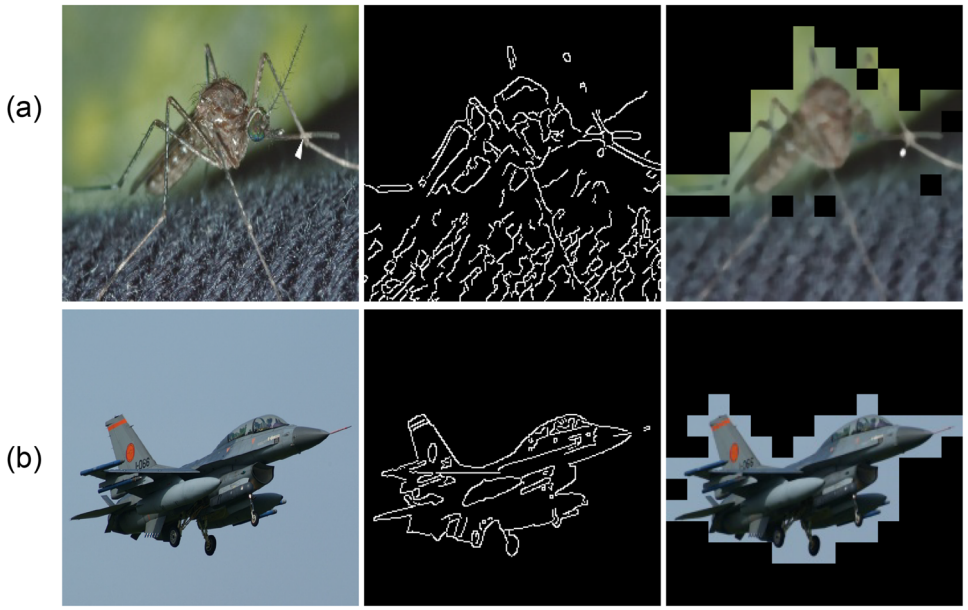

Fig. 7 Region-level spatial relevance. Left column indicating original image. The second column indicates edge points as possible feature points. The right column shows output image from our simulator. **a** Mosquito [28], **b** OpenImages [27]

the value can be manually tuned to improve redundancy reduction.

The threshold values for the temporal and spatial relevance analysis are obtained empirically. The value differs for different datasets as the foreground/background information changes. To illustrate the impact, we tested different threshold values and region sizes on the Mosquito data used in Table 3. Figure 8 illustrates the results. Here, we calculated edge points to identify spatial redundancy. For a given threshold value, Fig. 8a was generated. As we can see, the number of redundant regions decreases as we increase the size of the RPU region. This is because as we increase the region size, fine-grained regions get excluded from redundancy calculation. We observe a similar scenario as we decrease the threshold value. In Fig. 8b we use a region size of 16×16 for calculation. However, increasing the threshold value very high may cause improper ROI detection with key regions excluded. Therefore, it is necessary to identify the optimal point for the threshold and RPU region size.

The spatially redundant regions are labeled with an SRS value of 0, and temporally redundant regions are tagged with a TRS value of 0. Therefore, while using these datasets in CNN inference hardware such as [24], it is possible to skip computation for low SRS tiles and avoid repetitive computation for low TRS tiles.

4.5 Impact on CNN inference

The goal of the simulator is to generate custom event datasets for inference modeling. Here, we evaluate the impact on the accuracy of different CNN models when trained on these custom region-based ROI-extracted datasets. We tested on three different models with three different datasets. The results are listed in Table 4. As the table suggests, we see little to no drop in accuracy when trained on our

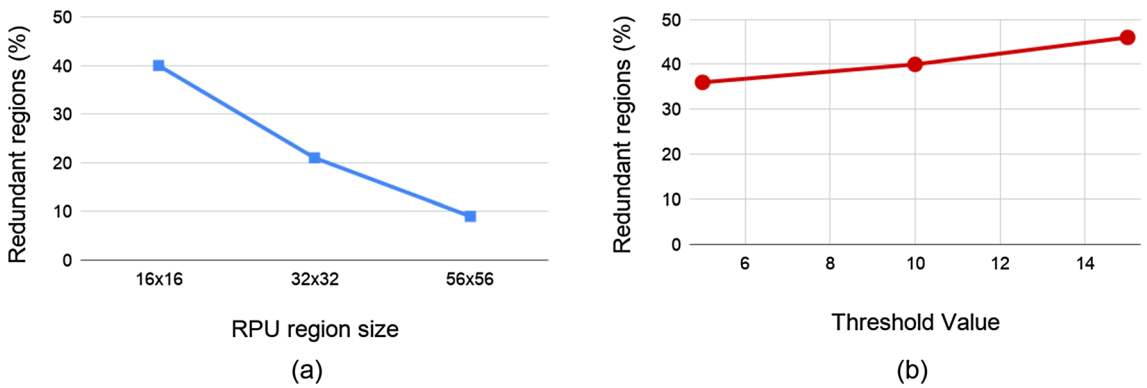

Fig. 8 Change in ROI size with **a** RPU region size and **b** threshold value (tested on Mosquito dataset [28])

Table 4 Impact on CNN model accuracy

Dataset	Models	Accuracy (Original) (%)	Accuracy (roi-based) (%)
MNIST	LeNet-5	98.93	98.8
FashionMNIST	LeNet-5	88	87.8
Mosquito [28]	ResNet-50	99	99
Mosquito	VGG-16	99	99

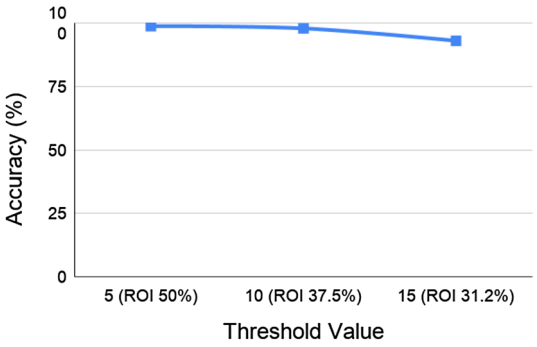

Fig. 9 Change in accuracy with threshold value for MNIST data

simulator-generated datasets for all the cases. However, we believe further studies can bring about even better results for event-detected datasets in the future. And our designed simulator can play a vital role in assisting these works.

The accuracy listed in Table 4 was achieved on the simulator-generated datasets with spatially redundant regions discarded (listed in Table 3). As we mentioned before, by increasing the threshold value, it is possible to decrease the relevant region size in images. However, it will impact the accuracy of the following CNN model. We tested it on the MNIST dataset for different thresholds. We see that the accuracy of the LeNet-5 model starts decreasing as we start increasing the threshold value beyond a certain point. This is shown in Fig. 9. Here, we select the RPU region size of 8×8 and edge points as spatial feature index. The threshold value of 5 indicates that the number of edge points in an 8×8 region has to be greater than equal to 5 to be considered a relevant region.

While designing the inference pipeline, it is also possible to allocate a reduced number of bit width for irrelevant pixels to reduce data transportation between the edge sensor and the computation unit. We tested this on the MOT17 dataset with the YOLOv3 model and found that the precision value remains the same for the YOLOv3 CNN model while operating on the region-detected dataset. Precision is defined as the ratio of correct detections to the sum of correct detections and false detections. This is shown in Fig. 10a, b. Here, we used a bit width ratio of 0.25 (non-relevant bit width/relevant bit width) for the generated custom dataset.

4.5.1 Hardware design evaluation

The end goal of this research is to develop suitable inference architectures that can be integrated with a region-aware camera sensor facilitating an event-based processing pipeline at the edge. Therefore, while designing the simulator, it is necessary to adopt ROI-detection functions with minimum hardware overhead. We prototyped the RPU and the RCM module of our simulator in a Virtex UltrScale plus FPGA (VCU118) to estimate the hardware cost associated with it. We opted to realize the RCM module because this is the module that draws visual attention in our simulator. The resource utilization is shown in Table 5. Here, the RPU

Fig. 10 **a** Object detected on original image with uniform bit-width (MOT17). **b** Object detected for image ROI extracted images (bit-width ratio of non-ROI/ROI =0.25)

Table 5 FPGA resource utilization of the RCM

Module name	LUT	FF	LUTRAM
RPU	183	90	16
RCM (784 RPUs)	143,472	70,562	12,544

is designed for an 8×8 region size, and the RCM data are estimated for 224×224 incoming image frames. The table indicates that the RCM module only consumes 12% combinational logics available in the Virtex FPGA. This confirms the viability of its realization with available CNN acceleration engines.

Next, we perform a qualitative comparison to our work with existing camera simulators found in the literature. This is shown in Table 6. The work in [5] and [29] captures events at the sensor interface and transmits events along with the frame. The simulators model an event camera where each pixel operates independently and asynchronously, reporting changes in brightness as they occur and staying inactive otherwise. However, both of the works primarily focus on temporal changes. The spatial redundancy is not thoroughly explored in these works. In contrast, our proposed design flow operates hierarchically. The region-level events are detected in two stages. First, we identify the temporal regions, and then spatial computations are performed on the extracted temporally relevant regions to reduce spatial redundancy. Since CNNs trained on our generated custom datasets can maintain the same level of accuracy (Table 4) without performing computation on the spatially redundant regions, our proposed simulator can serve as a development tool for modeling inference engines.

Finally, we compared our work with a tiled-based computation architecture that reduces computational complexity by efficiently encoding pixels in each region. Here, we considered the above-mentioned inference architecture that operates on the region-aware data generated by the proposed simulator architecture. We show that a similar CNN model trained on our simulator-generated dataset will yield better performance than other existing works aiming to reduce computational complexity. The detailed performance comparison is shown in Table 7. The table indicates that our ROI-based model can yield better accuracy while reducing computation on each frame.

Table 6 Simulator design comparison

	[5]	[29]	Ours
Transmit events along with frame	✓	✓	✓
Adaptive rendering	✓	✗	✓
Events detected	Pixel	Pixel	Region
Configurability	✓	N/A	✓

Table 7 Performance Comparison

	Convolution	Accuracy
CNN-PPA [30]	160 μs	93%
Our Design	31.2 μs	98.8%

5 Conclusion

This paper presents an event-camera simulator that emulates the behavior of an attention-based parallel camera sensor. The simulator computes the relevant score for each region and performs rendering operations for only relevant regions. The region-based ROI detection model adopted in this work can provide high-performance computing for high-level reasoning models. Our proposed simulator will serve as an analyzing tool to develop machine learning models that can best explore the event camera in the processing chain. The ROI detecting functions used in the simulator have low hardware cost. This makes it viable to implement in a distributed architecture. Our experimental results show that the attention-based approach used in this work can significantly reduce operation execution for inference engines.

Acknowledgements This work was supported by the National Science Foundation (NSF) under Grant-1946088.

References

1. Gallego, G., Delbruck, T., Orchard, G.M., Bartolozzi, C., Taba, B., Censi, A., Leutenegger, S., Davison, A., Conradt, J., Daniilidis, K., Scaramuzza, D.: Event-based vision: a survey. IEEE Trans. Pattern Anal. Mach. Intell. **1**, 1 (2020)
2. Cannici, M., Ciccone, M., Romanoni, A., Matteucci, M.: Asynchronous convolutional networks for object detection in neuromorphic cameras. in 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 1656-1665 (2019)
3. Pantho, M.J.H., Bhowmik, P., Bobda, C.: Towards an efficient cnn inference architecture enabling in-sensor processing. Sensors **21**, 6 (2021)
4. Gehrig, D., Rebecq, H., Gallego, G., Scaramuzza, D.: Asynchronous, photometric feature tracking using events and frames. CoRR, vol. abs/1807.09713, (2018). [Online]. Available: arXiv:1807.09713
5. Rebecq, H., Gehrig, D., Scaramuzza, D.: Esim: an open event camera simulator.in Proceedings of The 2nd Conference on Robot Learning, ser. Proceedings of Machine Learning Research, Billard, A., Dragan, A., Peters, J., Morimoto, J. Eds., vol. 87. PMLR, 29-31 Oct (2018), pp.969-982
6. Reichel, P., Hoppe, C., Döge, J., Peter, N.: Simulation environment for a vision-system-on-chip with integrated processing. in Proceedings of the 9th International Conference on Distributed

Smart Cameras, ser. ICDSC '15. New York, NY, USA: Association for Computing Machinery, p. 20-25 (2015)
7. Chen, G., Cao, H., Conradt, J., Tang, H., Rohrbein, F., Knoll, A.: Event-based neuromorphic vision for autonomous driving: A paradigm shift for bio-inspired visual sensing and perception. IEEE Signal Process. Mag. **37**(4), 34–49 (2020)
8. Sakakibara, M., et al.: A back-illuminated global-shutter cmos image sensor with pixel-parallel 14b subthreshold adc," in 2018 ISSCC. IEEE, 80-82 (2018)
9. Bobda, C., Velipasalar, S.: Distributed Embedded Smart Cameras: Architectures, Design and Applications. Springer Publishing Company, Incorporated, New York (2014)
10. Lab, S.S.: Smart image sensor. [Online]. Available: https://smartsystems.ece.ufl.edu/research/projects/smart-image-sensor (2021). Accessed 18 Oct 2021
11. Leng, L., Zhang, J., Khan, M., Chen, X., Alghathbar, K.: Dynamic weighted discrimination power analysis: a novel approach for face and palmprint recognition in dct domain. Int. J. Phys. Sci. **5**, 2543–2554 (2010)
12. Leng, L., Li, M., Kim, C., Bi, X.: Dual-source discrimination power analysis for multi-instance contactless palmprint recognition. Multimed. Tools Appl. **76**, 333–354 (2017)
13. Bi, Y., Andreopoulos, Y.: Pix2nvs: Parameterized conversion of pixel-domain video frames to neuromorphic vision streams. in 2017 IEEE International Conference on Image Processing (ICIP), 1990-1994 (2017)
14. García, G. P., Camilleri, P., Liu, Qian., Furber,S.: pydvs: An extensible, real-time dynamic vision sensor emulator using off-the-shelf hardware. in 2016 IEEE Symposium Series on Computational Intelligence (SSCI), 1-7 (2016)
15. Katz, M. L., Nikolic, K., Delbruck, T.: Live demonstration: Behavioural emulation of event-based vision sensors. in 2012 IEEE International Symposium on Circuits and Systems (ISCAS), 736-740 (2012)
16. Mueggler, E., Rebecq, H., Gallego, G., Delbruck, T., Scaramuzza, D.: The event-camera dataset and simulator: event-based data for pose estimation, visual odometry, and slam. Int. J. Robot. Res. **36**(2), 142–149 (2017). https://doi.org/10.1177/0278364917691115
17. Li, W., Saeedi, S., McCormac, J., Clark, R., Tzoumanikas, D., Ye, Q., Huang, Y., Tang, R., Leutenegger, S.: Interiornet: Mega-scale multi-sensor photo-realistic indoor scenes dataset. in British Machine Vision Conference (BMVC), (2018)
18. Hu, Y., Liu, S.-C., Delbruck, T.: v2e: From video frames to realistic dvs events. (2021)
19. Gehrig, D., Gehrig, M., Hidalgo-Carrio, J., Scaramuzza, D.: Video to events: Recycling video datasets for event cameras. in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), June (2020)
20. Mason, W.: New frontiers in imaging at DARPA MTO (Conference Presentation). in Infrared Technology and Applications XLVI, Andresen, B.F., Fulop, G.F., Miller, J.L., Zheng, L. Eds., vol. 11407, International Society for Optics and Photonics. SPIE, (2020)
21. Hossain Pantho, M.J., Bhowmik, P., Bobda, C.: Near-sensor inference architecture with region aware processing. in 2020 IEEE 38th International Conference on Computer Design (ICCD), 271-278 (2020)
22. Chen , J., et al.: Scamp5d vision system and development framework. in Proceedings of the 12th International Conference on Distributed Smart Cameras, ser. ICDSC '18. New York, NY, USA: Association for Computing Machinery, (2018)
23. Bhowmik, P., Pantho, M.J.H., Bobda, C.: Bio-inspired smart sision sensor: toward a reconfigurable hardware modeling of the hierarchical processing in the brain. J. Real-Time Image Process. **18**, 157–174 (2021)
24. Chen, Y., Krishna, T., Emer, J., Sze, V.: 14.5 eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks. in 2016 IEEE International Solid-State Circuits Conference (ISSCC), (2016)
25. Pang, J., Li, C., Shi, J., Xu, Z., Feng, H.: R^2 - cnn: fast tiny object detection in large-scale remote sensing images. IEEE Trans. Geosci. Remote Sens. **57**(8), 5512–5524 (2019). https://doi.org/10.1109/TGRS.2019.2899955
26. Milan, A., Leal-Taixé, L., Reid, I., Roth, S., Schindler, K.: MOT16: A benchmark for multi-object tracking. arXiv :1603.00831 [cs], (2016), arXiv:1603.00831. [Online]. Available: http://arxiv.org/abs/1603.00831
27. Kuznetsova, A., Rom, H., Alldrin, N., Uijlings, J., Krasin, I., Pont-Tuset, J., Kamali, S., Kamali, Popov, S., Malloci, M., Kolesnikov, A., Duerig, T., Ferrari, V.: The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale. IJCV, (2020)
28. Chumchu, R.P.M.A.K.P.P.: Image dataset of aedes and culex mosquito species. IEEE Dataport (2020). https://doi.org/10.21227/m05g-mq78
29. Kaiser, J., Vasquez Tieck, J.C., Hubschneider, C., Wolf, P., Weber, M., Hoff, M., Friedrich, A., Wojtasik, K., Roennau, A., Kohlhaas, R., Dill-mann, R., Zöllner, J.M.: Towards a framework for end-to-end control of a simulated vehicle with spiking neural networks. in 2016 IEEE International Conference on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR), (2016), pp. 127-134
30. Bose, L., et al.: A camera that cnns: Towards embedded neural networks on pixel processor arrays. in 2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019. IEEE, (2019), pp. 13350-1344