electronics

MDPI

Article

Event-Based Pedestrian Detection Using Dynamic Vision Sensors

Jixiang Wan [1,2], Ming Xia [1], Zunkai Huang [1], Li Tian [1], Xiaoying Zheng [1], Victor Chang [3], Yongxin Zhu [1,*] and Hui Wang [1,*]

1 Shanghai Advanced Research Institute, Chinese Academy of Sciences, Shanghai 201210, China; wanjx@sari.ac.cn (J.W.); xiam@sari.ac.cn (M.X.); huangzk@sari.ac.cn (Z.H.); tianl@sari.ac.cn (L.T.); zhengxy@sari.ac.cn (X.Z.)
2 School of Microelectronics, University of Chinese Academy of Sciences, Beijing 100049, China
3 School of Computing & Digital Technologies, Teesside University, Middlesbrough TS1 3JN, UK; V.Chang@tees.ac.uk
* Correspondence: zhuyongxin@sari.ac.cn (Y.Z.); wanghui@sari.ac.cn (H.W.)

Abstract: Pedestrian detection has attracted great research attention in video surveillance, traffic statistics, and especially in autonomous driving. To date, almost all pedestrian detection solutions are derived from conventional framed-based image sensors with limited reaction speed and high data redundancy. Dynamic vision sensor (DVS), which is inspired by biological retinas, efficiently captures the visual information with sparse, asynchronous events rather than dense, synchronous frames. It can eliminate redundant data transmission and avoid motion blur or data leakage in high-speed imaging applications. However, it is usually impractical to directly apply the event streams to conventional object detection algorithms. For this issue, we first propose a novel event-to-frame conversion method by integrating the inherent characteristics of events more efficiently. Moreover, we design an improved feature extraction network that can reuse intermediate features to further reduce the computational effort. We evaluate the performance of our proposed method on a custom dataset containing multiple real-world pedestrian scenes. The results indicate that our proposed method raised its pedestrian detection accuracy by about 5.6–10.8%, and its detection speed is nearly 20% faster than previously reported methods. Furthermore, it can achieve a processing speed of about 26 FPS and an AP of 87.43% when implanted on a single CPU so that it fully meets the requirement of real-time detection.

Keywords: dynamic vision sensor; event data; pedestrian detection; autonomous driving

check for updates

Citation: Wan, J.; Xia, M.; Huang, Z.; Tian, L.; Zheng, X.; Chang, V.; Zhu, Y.; Wang, H. Event-Based Pedestrian Detection Using Dynamic Vision Sensors. *Electronics* **2021**, *10*, 888. https://doi.org/10.3390/electronics10080888

Academic Editor: Daniel Morris

Received: 1 March 2021
Accepted: 2 April 2021
Published: 8 April 2021

1. Introduction

As one of the popular research branches in object detection, pedestrian detection has encountered a significant boost with the tremendous development of deep learning algorithms in the last decade. It is mainly applied in the fields of human behavior analysis, gait recognition, and person re-identification [1–3]. Moreover, it provides important contributions to video surveillance, traffic statistics, and especially in autonomous driving. For automatic driving systems, accurate and rapid detection means drivers have more reaction time to avoid collisions. To date, research on pedestrian detection has made great progress. In general, these detection algorithms are based on time-of-flight sensors or frame-based imaging sensors. The cost issue is the main obstacle to the large-scale deployment of time-of-flight sensor-based systems involving LiDAR [4]. Conventional image sensor generally scans the entire scene at a predetermined frame rate and outputs a sequence of static frames with fixed intervals, regardless of any target activity in the scene. Moreover, what happens between adjacent frames is not captured by the camera, leading to undersampling of information, which cannot completely satisfy the requirements of rapid analysis or real-time monitoring in autonomous driving applications [5]. In addition, the response speed of such cameras is usually limited by the frame rate, and the output continuous video frames are usually highly redundant, resulting in a waste of storage space, computing power, and time.

Inspired by the principle of the biological retina, researchers have designed a dynamic vision sensor (DVS) [6,7]. Unlike traditional cameras that continuously measure the absolute luminance of all pixels at a fixed frame rate, the DVS captures each pixel luminance change at an asynchronous rate and generates an event output only when transient changes in the scene are captured. Therefore, DVS can output the sensitive motion information asynchronously with a high temporal resolution, high dynamic range, low latency, and low bandwidth requirements. The sparse event data captured by DVS reduces the redundancy of source data, which can greatly reduce the computational complexity of back-end vision algorithms and improve computing efficiency. Moreover, DVS focuses on the light intensity changes caused by object motion, and the perception of moving pedestrians focuses on contour and shape. Its output has no texture or appearance, making it difficult to determine the pedestrian's specific identity. In this way, the inevitable privacy problem when using conventional cameras can be solved.

Spiking neural networks (SNNs) are the most suitable network architecture for processing asynchronous event data due to their unique impulse coding properties and biological interpretation capabilities. However, SNNs bring the significant drawback of lacking general and efficient learning algorithms, such as back-propagation [8]. SNNs usually require hand-designed feature filters and cannot automatically learn image features as well as convolutional neural networks (CNNs), which greatly limits their promotion and application in real life. The present literature on CNNs focuses on frame-based data that have achieved outstanding achievement in applications such as object detection, classification, and image segmentation. Many researchers have found that combining discrete event data received by DVS with the well-matures CNNs architecture used in traditional vision is a very promising solution [9]. Chen [10] used a self-supervised learning approach where grayscale images are fed into a state-of-the-art CNN model to predict results (called "pseudo-labels") that are used as ground truth for subsequent training of an event-based object detection model. This method achieves high-speed detection at 100 FPS in real outdoor scenes. Li [11] proposes a joint framework combining event-based and frame-based vision for vehicle detection. They use convolutional SNNs to generate visual attention maps from events and synchronize them with frame-based data streams synchronization. Jiang [12] designed a confidence map fusion scheme to integrate the image frames and event streams and obtained more accurate results than a single data channel. Chen [13] proposed a pedestrian detection framework that fuses multiple event stream coding methods to achieve excellent results. Therefore, encoding discrete event streams into frame-like structures and directly applying them to CNNs is a pressing problem. This is also the main motivation for this work.

In this paper, to exploit the potential of event data in the field of object detection, we propose an end-to-end pedestrian detection pipeline that can detect the presence of pedestrians directly from the event stream from DVS. Its main contribution can be summarized as follows:

1. We propose an online pedestrian detector for asynchronous event streams. The approach allowed easy identification of pedestrians directly from the event stream data collected by DVS;
2. We propose a novel event-to-frame encoding method to encode the event stream more effectively. Compared with previous methods, our method could thoroughly integrate the inherent characteristics of the events and improve the performance of pedestrian detection;
3. We construct an asynchronous feature extracting scheme that could reuse the intermediate features to further decrease the calculation amount. This asynchronous encoding mechanism fits well with the inherent characteristic of asynchronous event streams;

4. We autonomously collected and annotated a custom pedestrian detection dataset using the DAVIS346 event sensor and further evaluated the performance of our proposed event-to-frame encoding method and asynchronous pedestrian detection framework based on the dataset.

The rest of this paper is organized as follows: Section 2 provides the details of experimental methods, including event-to-frame construction and target detection network. Section 3 discusses in detail the experimental process and results. This is followed by Section 4 to conclude the paper.

2. Methodology

2.1. Event-Frame Construction

In the following, we briefly describe several common conversion methods from the event stream to the event frame.

2.1.1. Event-Stream Encoding Based on Frequency

According to the imaging principle of DVS, more events occur near the edges of moving objects. Thus, the event frequency can be used to distinguish the contour of a moving target from the background. Reflected on the image frame plane, the frequency of events occurring at that pixel location can thus be characterized using the pixel brightness in the image. Based on this assumption, Chen [10] proposed a representation for frequency-based event-stream. Specifically, the event data are converted to an image by slicing the event data at fixed time intervals. Each pixel in the image takes the value calculated by following normalization Equation (1):

$$\sigma(x) \;=\; 255 \times \frac{1}{1 \,+\, e^{-\,x/2}} \tag{1}$$

where x is the total number of events occurred at one pixel during a fixed time interval and $\sigma(x)$ is the pixel value, taking values in the range of [0, 255]. This encoding method that maps event frequency to pixel values can greatly enhance the contrast of edge contours of the object, which is beneficial to subsequent classification and detection tasks. On the other hand, this encoding method makes the noise with lower event frequency occupy lower pixel weights in the event frame, filtering the noise.

2.1.2. Event-Stream Encoding Based on Surface of Active Events

Another common form of representation for event data is to create a structure named surface of active events (SAE) [14,15]. SAE was proposed to combine spatial and temporal information of events. Specifically, it records the most recent timestamp of each pixel at (x, y) of incoming events as the Function (2):

$$SAE : (x, y) \mapsto t \tag{2}$$

Therefore, the pixel value in the event-frame is directly determined by the latest time t when the corresponding event occurs. Considering that the timestamp in the event stream output from the event camera is monotonically increasing, and the timestamp will become very large over a long period, it is necessary to normalize the SAE. Normalization can be performed by the following Formula (3):

$$SAE : (x, y) \;=\; 255 \times \frac{t_p - t_0}{T} \tag{3}$$

where t_p is the latest timestamp of each event, t_0 is the initial time and T is the time interval between each frame. This encoding method is directly related to the original timestamp of the event data, and it reflects the temporal information well. The pixel value in the event frame indicates the moment of the event, and their gradients indicate the direction and speed of the event stream. However, the disadvantage of this method is that it treats camera noise as a normal event, and the pastime information will be overwritten by the updated timestamp.

2.1.3. Event-Stream Encoding Based on LIF Neuron Model

The leaky integrate-and-fire (LIF) neuron model [16–18] is an integration mechanism that takes inspiration from the functioning of SNN to maintain the memory of past events. According to the LIF model, each pixel can be regarded as an independent neuron. The neuron rose its membrane potential by a fixed increment by receiving input pulses (event data at this pixel), and the membrane potential will dissipate at a certain rate over time. When its membrane potential exceeds a set threshold, the neuron sends a pulse and enters a refractory period (membrane potential returns to zero and fails to respond to any pulses). After a while, the neuron is reactivated and begins the next round of receiving input pulses. Within a specific time interval, the total number of pulses emitted by the neuron at each pixel point can be used as the pixel value of the corresponding frame. This encoding approach describes the continuous nature of temporal information and the ability of events to occur with intensity. The individually occurring, discontinuous noise is difficult to maintain the membrane voltage with the breakthrough threshold, so the noise in the event frame can be filtered to a great extent.

In Figure 1, we have selected a typical scene to compare the output of the different event-frame encoding methods mentioned above and give a simple visual analysis. As shown in Figure 1a, this example includes two pedestrians, the pedestrian with a backpack in the center of the image (denoted as pedestrian A) and the smaller pedestrian on the top left (denoted as pedestrian B). It is not difficult to see that all the event-frame encoding based on frequency (as shown in Figure 1b), SAE (as shown in Figure 1c), and LIF (as shown in Figure 1d) can clearly reflect the relatively complete outline and walking motion of the moving pedestrian A. However, the characterization results for pedestrian B, who in the far position, is not as good as expected. Due to the relatively long distance from the event camera, the events emitted by pedestrian B are less captured by the sensor (we call them "sparse events"). Therefore, the performance of this object in the event frames does not perform as successfully as the previous pedestrian A after the corresponding encoding. It is important to note that in practical application scenarios, pedestrian B is still truly present and meaningful even though the number of events it collects is less. It can be seen that these three encoding methods may be useful in some scenarios that focus only on the dominant objects. However, they are still inadequate for pedestrian detection tasks, which require detecting the location of all pedestrians within the scope of vision. In practice, if these "sparse event" regions cannot be characterized in the event frames, the subsequent object detection results are bound to have some false detections and missed detections.

By analyzing the encoding process in detail, it is easy to find out the reason. During the LIF-based encoding process, due to the sparsity and the discontinuity of the "sparse events", it is difficult to maintain the membrane potential state up to the threshold value and emit the output pulse. The events in these regions behave similarly to camera noise, and they are "ignored" by the LIF conversion process with powerful noise filtering. For both Frequency-based and SAE-based encoding methods, we assume that their shortcomings lie in the normalization method. The normalization operation is performed over the entire image (the whole field of vision), whether based on the number of events or the timestamp. Therefore, pixels with higher frequency or newer timestamps necessarily have higher weights. In contrast, the pixel values of "sparse event" are at a relatively low level. When there is a large difference between the weights of two parts, the "sparse events" are greatly "suppressed" by the main part after the full domain normalization. From the event-frame results, the target in this region behaves very close to the background region, as if it "disappears". The key point to solve this problem is to find a solution that balances the weight of the pixel values encoded in the event frames for event regions of different densities.

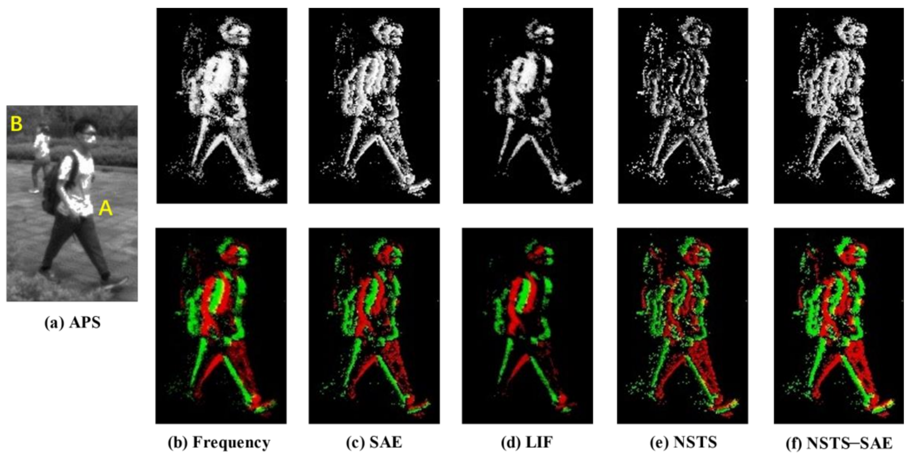

Figure 1. Comparison of the result of different event frame encoding methods. (**a**) The conventional grayscale frames were provided by the active pixel sensor (APS) channel of the DAVIS camera. Event-frame reconstruction from the dynamic vision sensor (DVS) channel by (**b**) frequency [10], (**c**) surface of active events (SAE) [15], (**d**) leaky integrate-and-fire (LIF) [18], (**e**) proposed neighborhood suppression time surface (NSTS), (**f**) proposed NSTS–SAE encoding method. Upper row: the representation by adding positive and negative polarity events (white represents the event, black represents the background). Bottom row: the individual representation of the positive and negative polarity events (red for positive, green for negative, and black for background).

2.1.4. Our Proposed Event-Stream Encoding Method

Inspired by the functioning of the time surface generation approach mentioned in [9] and the HATS method [19], we propose a novel description of the time surface called neighborhood suppression time surface (NSTS). The key point of NSTS is that the intensity of each pixel on the time surface is only suppressed by its local neighborhood and is no longer related to the entire image. In this way, the suppressive effects of the "dense events" area on the weight of the "sparse events" area can be avoided as much as possible.

Our proposed NSTS encoding method can be intuitively understood as follows: assuming a time surface, and the value of each pixel on the time surface maps the corresponding event. When an event arrives, the pixel value of the corresponding location on the time surface is updated with a predetermined value, while the pixel values at the surrounding locations are suppressed. For a given pixel location, the more events occur at its neighborhood locations, the more suppressed that pixel point is and the greater the pixel value reduction. Conversely, the pixel position with fewer events occurring in the neighborhood dominations is less penalty. In the absence of events in the neighborhood, the pixel value is maintained until the time surface's cutoff time. In this way, the same contour is produced on the surface of time for both "dense event" and "sparse event" regions.

To further reduce the impact of dense events, we can also add a time filter before the event integration. Specifically, given a time interval threshold, when an event arrives, by comparing the time interval between the previous event and the current event at the same location, when the interval is less than the given threshold, these two events are considered repeated events, and NSTS will not be updated. However, each event with a larger interval passes through the filter smoothly, makes corresponding changes to the NSTS. Therefore, the "dense events" area's contribution to the NSTS is reduced, and the "sparse events" area is not affected by the filter.

Considering the above two mechanisms together, our proposed NSTS approach is described in detail as follows: given two hyperparameters, R represents the radius of the considered neighborhood locations and T_{thr} symbolizes the time interval threshold. Then, we will initialize two three-dimensional arrays of all zeros, which are event-frame $S(x, y, p)$ and timestamp surface $T(x, y, p)$, respectively. The first two dimensions (x, y) represent the pixel position where the event occurred in the frame, and the third dimension (p) represents the polarity of the event. Whenever each event (x, y, p, t) arrives, we first compare t to the latest timestamp $T(x, y, p)$ recorded at the (x, y) position of the timestamp surface. Only when the difference between the two is greater than T_{thr}, we then consider updating event-frame F. All the pixel values F' located in which neighborhood window $(2R + 1) \times (2R + 1)$ will be subtracted by 1. Meanwhile, the $S(x, y, p)$ at the event's location will be reinitialized to zero. The specific update process can be expressed by formula (4), and the pseudo-code of the proposed algorithm is given in Algorithm 1.

$$S(x + i, y + j, p)\Bigg| = \begin{cases} 0 & if\ i = 0\ and\ j = 0 \\ S(x + i, y + j, p) - 1, & if\ -\mathrm{R} \le i, j \le R \\ S(x + i, y + j, p), & otherwise \end{cases} \tag{4}$$

Algorithm 1 Neighborhood suppression time surface (NSTS)

```
Input: Event $e$ = $(x,y,t,p)$
Output: Time Surface $S(x,y,p)$
Initialization: $S(x,y,p) \leftarrow 0$ for all $(x,y,p)$
Initialization: $T(x,y,p) \leftarrow 0$ for all $(x,y,p)$
For each incoming event $(x,y,t,p)$, extracting $S(x + i,x + j)(-R \le i \le R,-R \le j \le R)$, update $S$:
  if $t - T(x,y) \ge T_{thr}$ do
      $T(x,y,p) \leftarrow t$
      for each $S(x + i,y + i,p)$ do
          $S(x + i,y + j,p) \leftarrow S(x + i,y + j,p) - 1$
      end for
  end if
$S(x,y,p) \leftarrow 0$
```

Figure 1e illustrates the coding performance of the NSTS method in a real pedestrian scene. It can be clearly seen that the contours of the moving target are very concise and clear. Compared with the previous three encoding methods, the feature contour of the over-suppressed pedestrian B is greatly enhanced, showing a relatively complete external contour of the pedestrian. The presence of pedestrians at this location can also be clearly inferred from the event frame images. In addition, our NSTS event-frame obtains a more distinctive result with a high contrast around the edge by asynchronously reducing the pixel weight at the edge of each event area.

However, the NSTS encoding method seems to pay too much attention to the contours of the moving objects so that the representation of the internal details of the object (such as the pedestrian A) is not as complete as SAE and Frequency. Therefore, the target in the NSTS event-frame usually has only contour lines and exhibits less hierarchy. We learned from the popular "attention mechanism" concept in deep learning [20,21] to enhance the internal detail feature information by introducing the SAE encoding method to describe the texture details of the target and fusing it with the NSTS coding method. Comparing the visualization results are shown in Figure 1e,f, it can be found that the texture details, such as the shoulders and backpacks of pedestrian A in the mixed event-frame NSTS–SAE, are much richer and more complete than in NSTS. At the same time, the pixels belonging to pedestrian B still retain sufficient contrast. Visually, NSTS–SAE brings a strong sense of hierarchy. It can be seen that the "attention mechanism" in event-frame reconstruction is conducive to making full use of the information in the event stream.

The process of fusing NSTS and SAE results in the event integrator is represented in Figure 2. This integration is similar to mixing the high-contrast edges extracted by NSTS as weights into the SAE to construct a new event frame. Since we have already recorded the timestamps surface of each pixel location simultaneously during the NSTS calculation, the fusion of two frames does not introduce additional calculation.

2.2. Object Detection Based on CNN

The original event stream is encoded into an event-frame image by an event integrator, and the event stream-based pedestrian detection task is transformed into a conventional frame-based object detection task. Among the object detection approaches in the field of computer vision, the YOLO detector has received much attention from researchers and engineers since its inception [22]. Its latest achievement, YOLOv3, offers significant advantages in detection accuracy and speed in multi-target detection tasks [23].

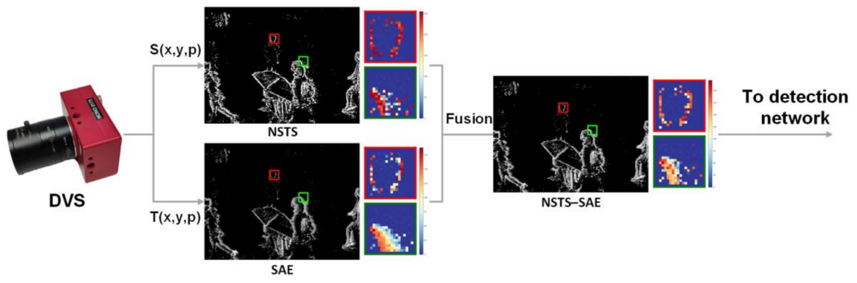

Figure 2. The overview of the event integrator with NSTS–SAE (neighborhood suppression time surface-surface of active events) fusion. Event integrator first encodes the event output in the dynamic vision sensor (DVS) into $S(x,y,p)$ with NSTS method as formulated in Algorithm 1. Then, the NSTS result is fused with the intermediate variable $T(x,y,p)$ in the encoding process regarded as the SAE event-frame. Finally, the fusional NSTS–SAE event-frame is used as the input of the detection network.

In this work, considering the computing power limitation of edge computing, we retained the detection head parts of the YOLOv3 model and replaced the feature extraction backbone net of Darknet53 with MobileNetV3_small [24] (hereafter abbreviated as MobileNetV3) to compose the architecture of our proposed object detector. In this way, the entire network could be applied more smoothly on conventional CPU platforms instead of expensive and energy-intensive GPUs. Table 1 shows the information of Darknet53 and MobileNetV3. We set the input network resolution to 352 × 352 for our datasets. In theory, DrakNet53 with a receptive field of 725 × 725 and MobileNetV3 with a receptive field of 639 × 639 are both sufficient as a backbone network since the receptive field of the network is much larger than the resolution of input features. From the perspective of parameter amount and computational complexity, compared with DarkNet53 with 40.58 M parameters, MobileNetV3 contains only 2.51 M parameters, and floating-point of operations (FLOPs) is nearly two orders of magnitude lower. For the detection speed, Darknet53′s detection frame rate on the GPU is greater than that of MobileNetV3. However, the performance on the CPU is far inferior to MobileNetV3. This is because the depth-wise separable convolution in MobileNetV3 divides a standard convolution into two convolutions (depth-wise and point-wise). GPUs with higher parallel computing capabilities increase the number of computing layers and the amount of data exchanged from memory. On the contrary, for CPUs lacking parallel computing capabilities, the dominant factor in the total computing time is the total amount of computing. The depth-wise separable convolution just reduces the number of parameters and correspondingly reduces the total amount of calculation. This shows that MobileNetV3 is more suitable for computing platforms with limited computing power, such as edge computing applications.

Table 1. Comparison of parameters and performance of different backbone networks.

Backbone Model	Input Network Resolution	Receptive Field Size	Parameters	FLOPs	FPS (GPU Tesla V100)	FPS (CPU Xeon W-2145)
Darknet53	352 × 352	725 × 725	40.58 M	35.3 G	93.5	3.4
MobileNetV3	352 × 352	639 × 639	2.51 M	307.2 M	76.7	27.6

2.2.1. Grids Partition Detection Model

In the conventional video analysis or object detection, each frame of data is independently processed by the entire detector, and recalculating all intermediate feature maps, even if only some pixels have changed between these consecutive frames. Moreover, the feature extraction network is precisely the bottleneck of computations. However, for event cameras, the nature of the output event is to respond asynchronously to change the light intensity at the current pixel location. The sparsity of the event stream determines that the effective pixels in the event frame do not cover every position of the image matrix as conventional image frames. Assuming that the event frames are directly utilized as the input of the CNN network, it will inevitably cause a large amount of wasted power and computations, which is incompatible with the nature of event cameras.

To exploit the potential of event cameras in promoting computational efficiency, we proposed a compromise scheme called grid partition (GP). Specifically, each event-frame is divided into $N \times N$ grids, and the changes of features are calculated independently in each grid. Neither each pixel location is treated individually, nor all grids in the event-frame are treated as a whole. Since only the event features in several regions change between consecutive event frames, only the features corresponding to changed grids need to be recalculated, while the features of other unchanged grids can directly reuse the results extracted from the previous frame without recalculating through the detection network.

Figure 3 showed the framework of the event-based pedestrian detector. The first four convolutional layers in the MobileNetV3 network downsample the input image by a factor of 8 and output an intermediate feature map of size 44×44. This intermediate feature map is the feature map that we need to reuse, called the event reuse feature map (ERFM). The detection process is described as follows: the detection of the first frame is the same as the conventional target detection process, and all grids are input to the detector, but the location of the grid with event activity in the current frame and the intermediate feature map ERFM is additionally recorded. Subsequent event streams are passed through the event integrator, and the grid with event activity in the current frame is calculated before being input to the detector. The event feature patches of these grids are extracted in parallel and replace the feature map patches at the corresponding positions of ERFM of the previous frame. The newly fused ERFM is then fed into the subsequent CNN detection model to obtain detection results for the current frame. As shown in Figure 3, the six small blocks (marked in orange) in the ERFM are the event-active grids that need to be recomputed for the current frame. Compared with recalculating all 16 grids, the computational effort is greatly reduced.

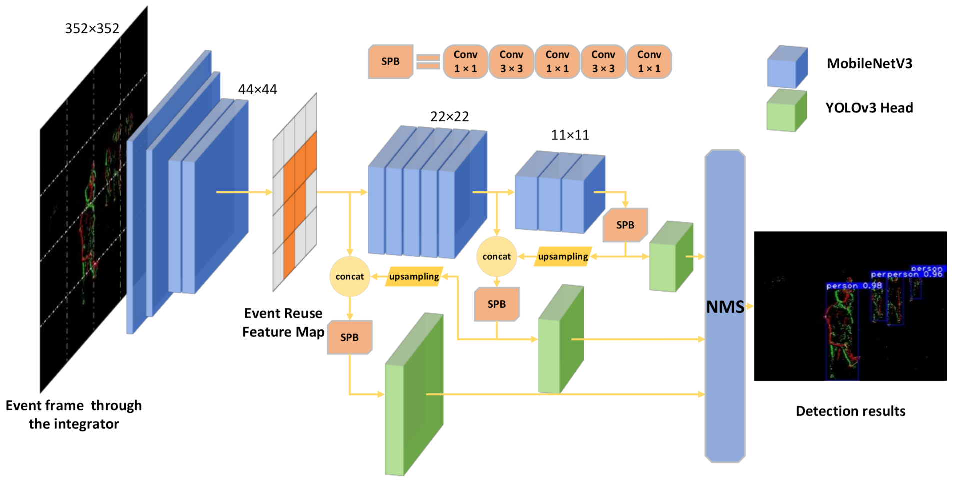

Figure 3. Framework of the pedestrian detector with grids partition of feature maps. The event-frame through the integrator is divided into 4×4 grids, and only the changed grids will be calculated by CNN to get the intermediate feature map, and then the corresponding area of the feature map saved in the previous frame is updated (6 orange marked grids). The fused intermediate feature map is used as the feature of the current frame and is inputted into the latter part of the detector. The detection result is acquired from all the three prediction branches through the NMS (Non-Maximum Suppression).

2.2.2. Asynchronous Event Frame Detection Model

For the entire event frame, we can get a distinct and rich DVS image by accumulating all events in a fixed time interval. This is based on the knowledge that sufficient events occurred within this time window. Now let us think in another way. For regions with relatively high event frequency, only a short time window is needed to get enough events to construct a complete event frame. For regions with fewer events, the corresponding time window needs to be expanded to capture more events. Thanks to the idea of "grid partitioning", we can independently consider the event-frame construction process in each grid. To determine whether the events in each area are adequate to construct a clear and detailed event frame, we count the number of events in each grid and compare it with an empirical threshold E_{thr}. For grids with an event count larger than this threshold, we consider that the grid's events are sufficient to reflect the object motion in the region, and directly input this part of the DVS image into the pedestrian detector, and then reinitialize the event frame in the grid. On the contrary, for grids less than the threshold, we continue to keep this portion of the time surface, and the construction of the next event-frame in the grid is based on this saved time surface instead of the initial zero value until the accumulation of events in this grid reaches our requirements. In this way, each grid region can build the corresponding event-frame image independently, and the moment of input into the CNN depends only on the number of events in the grid, rather than completing the detection of the whole event frame at the same time. Therefore, we call it the asynchronous event-frame detection method.

This asynchronous event-frame detection scheme fits well with the idea of asynchronous output event streams from DVS. Since in the extreme case, each pixel is considered as a grid, and each event updates the event-frame at that pixel location, while that pixel is fed into CNN. It is the most primitive single event detection model. However, in terms of grid statistics, parallel computation and merging of subsequent feature maps, processing each pixel as a grid will result in many tedious computations. At the same time, the merging of feature maps is directly related to the object detection effect. Therefore, the choice of suitable grid size is especially critical to balance the calculation. After experimental comparison and analysis, we divide the input feature with a resolution of 352×352 into 4×4 grids.

3. Experiments and Discussion

In this section, we present the implementation process and related parameters of our proposed pedestrian detection architecture in detail and conduct extensive discussions and analyses of the experimental results

3.1. Datasets

As far as we know, the datasets based on DVS are currently much scarcer than conventional cameras in the field of object detection. Especially, there is no fully public labeled dataset for pedestrian detection. Although several event-based pedestrian detection studies have been conducted [12,13], the datasets they used are not publicly available. Hence, we collected about 6.5 h of event streams with the help of the DAVIS346 event sensor (iniVation, Zurich, Switzerland). DAVIS is a composite sensor that combines event-driven asynchronous readout of temporal contrast with synchronous frame-based APS readout of intensity [25]. The resolution of this event camera is 346×260. The dataset contains multiple real-world pedestrian scenarios, including campus roads, traffic intersections, subway station exits and pedestrian crossings. We collected 141 short event streams with a total of 488 s, including 84 M events, to construct an event data-based pedestrian detection dataset named Pedestrian-SARI (Shanghai Advanced Research Institute). Among them, 107 event streams are used as the train set, and the rest are used as the test set. Each event stream clip is converted into corresponding event frames and annotated manually to record locations of pedestrians as well as the label. In total, 6061 event frames reconstructed with a temporal window of 40 ms are annotated. Figure 4 shows some examples from the pedestrian-SARI dataset.

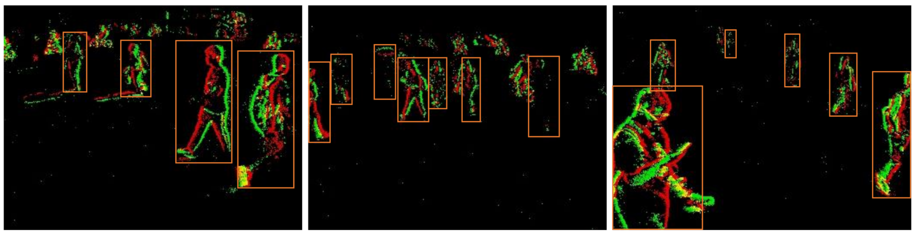

Figure 4. Some examples and pedestrian annotations in the Pedestrian-SARI (Shanghai Advanced Research Institute) dataset. All event frames are constructed by the NSTS–SAE method.

3.2. Event-Frame Construction

We compare our proposed NSTS against the previously published event-frame accumulation method: the event-stream encoding method based on frequency, SAE, and LIF. All these methods are based on the author's public implementation. The specific implementation method is as described in Section 2.1. It is worth mentioning that we have retained event frames with both positive and negative polarities for each construction result. Therefore, the input of the pedestrian detector is equivalent to these two integrated channels. Following, we introduce in detail the parameters that our proposed method relies on.

To be concise, we use the name of the event-to-frame construction method to name the test case. For example, frequency, SAE, and LIF, respectively, represent the event frames constructed by the corresponding event stream encoding method. The event frames drove from the NSTS construction approach with the event filter are abbreviated to NSTS. Analogously, NSTS–SAE denotes the event frames obtained by fusing the two results of NSTS and SAE achieved previously.

3.3. Pedestrian Detection Performance

3.3.1. Comparison of Different Event Frame Encoding Methods

In this work, we designed a pedestrian detection system based on standard MobileNetV3 and YOLO head as our baseline for the event-based detector (EBD). The baseline method was trained from scratch with five different datasets as described in Section 3.2, respectively. In this way, we compare the impact of different event-to-frame construction methods on pedestrian detection accuracy. All the models are trained using the Adam optimizer with a total of 50 epochs. The initial learning rate is 10^{-4}, betas are (0.9, 0.999), eps is $10^{-8,}$ and the learning rate is dynamically adjusted with the trained batches by cosine annealing scheduler [26]. Early stopping was adopted to prevent overfitting of the model. The batch size was chosen as 32 depending on the memory of the GPU. For data augmentation, we only used horizontal flip by a probability of 0.5. Considering the cost of training time, all the training processes are completed on the NVIDIA Tesla V100 GPU (Nvidia, Santa Clara, CA, USA), and it takes about 3 h to train a model from scratch. The detection processes are on the Intel Xeon W-2145 CPU (Intel, Santa Clara, CA, USA). Our methods are implemented based on the open-source Pytorch framework.

We use average precision (AP) as a metric to evaluate the detection performance of different models. The APs of the detectors with different encoding methods are provided in Table 2. Compared with previously reported methods (e.g., frequency, SAE, LIF), our proposed NSTS method achieves better detection performance. This shows that increasing the pixel weight of "sparse events" in DVS images is beneficial to improve detection accuracy. Moreover, the NSTS–SAE frame, which is fused by both NSTS and SAE, has the best performance. The highest AP of 86.37% was obtained, which indicates that NSTS and SAE event frames can be reconsidered as complementary to each other.

Table 2. The performance comparison of different event-to-frame encoding methods evaluated by the event-based detector (EBD) (IOU0.5 AP).

Encoding Methods	AP (%)	Detect Time (ms)
Frequency [10]	81.48	
SAE [14,15]	82.81	
LIF [18]	78.92	47.58
NSTS (proposed)	84.08	
NSTS–SAE (proposed)	86.37	

It is worth stating that our proposed event-based detection pipeline includes two parts: event frame encoding and event frame detection. Event frame encoding is performed in real time as the events arrive, and the consumption time depends on the timestamp of the initial event and the cutoff event, independent of the encoding method or the detector. Therefore, the detection time in Table 2 refers to the time between the event frame input to the detector and the output detection result. This metric is compared to illustrate the detection speed of the different detectors.

The visualized pedestrian detection results from the detectors are presented in Figure 5. As shown in Figure 5a, this is a typical traffic road crossing scene. Interestingly, comparing Figure 5c,e can clearly that NSTS and SAE focus on the different locations, so the fusion of NSTS and SAE event frames can complement each other. Therefore, the results from NSTS–SAE were superior to others.

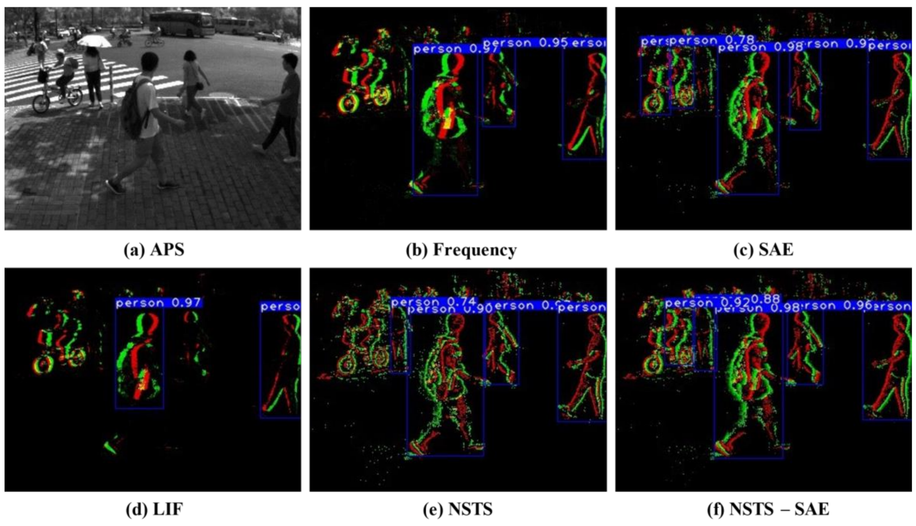

Figure 5. Detection results of different event frame encoding methods in EBD (event-based detector). The blue boxes in each figure mark the corresponding predicted object position. (**a**) APS (active pixel sensor) channel from DAVIS camera (used to display real-world scenario); (**b**) frequency [10]; (**c**) SAE (surface of active events) [15]; (**d**) LIF (leaky integrate-and-fire) [18]; (**e**) NSTS (neighborhood suppression time surface); (**f**) NSTS–SAE.

3.3.2. Comparison of Different CNN Detection Schemes

(1) Grids partition detection model

Compared with the images captured by conventional cameras, the data in DVS images are sparser and simpler. Reusing part of the feature map through GP methods is an effective method to avoid spending much computation resources in the repeated feature calculation process. Based on our proposed GP method, we improved the feature extraction network MobileNetV3 in the EBD, as described in detail in Section 2.2.1. In this work, we take $N = 4$, that is, the event frame and the corresponding feature map is divided into a total of 16 sub-grids. Then we used the same training strategy as the baseline to re-train the GP detection model (EBD–GP) on different event-frame datasets separately and compared the detection results in detail. As illustrated in Table 3, in terms of AP, the detection performance of the EBD–GP network is generally slightly inferior to the baseline. Compared with the five encoding methods, the AP is reduced by −0.05–0.34%. However, the detection speed is reduced from 47.58 ms of baseline to 41.89 ms of EBD–GP, saving about 12% of the detection time. Here we do a brief theoretical analysis. With an input resolution of 352 × 352, the FLOPs of the first four layers of MobileNetV3 are about 105.0 M, accounting for nearly 34% of the entire backbone network. Assuming that we can reuse half of the feature maps between two frames, the total calculation amount will be saved by nearly 17%. This is why our EBD–GP method can significantly reduce the detection time.

Table 3. The performance comparison of pedestrian detection accuracy and detection speed of event frames constructed by different encoding methods evaluated by EBD (event-based detector) and EBD–grid partition (GP), respectively (IOU0.5 AP).

	EBD		**EBD–GP (Proposed)**	
	AP (%)	**Detect Time (ms)**	**AP (%)**	**Detect Time (ms)**
Frequency [10]	81.48	47.58	81.52	41.89
SAE [14,15]	82.81		82.44	
LIF [18]	78.92		78.97	
NSTS (proposed)	84.08		83.96	
NSTS–SAE (proposed)	86.37		86.03	

(2) Asynchronous event frame detection model

Based on the EBD–GP, we further proposed an asynchronous event frame detection model (EBD–AGP). We evaluated the performance of pedestrian detection results by independently accumulating events and updating event frames in each grid. In the experiment, we set the event count threshold $E_{thr} = 200$ to determine whether the time surfaces within grids need to be reinitialized. For convenience, we will use this method of asynchronously initializing the time surface with the mark "A" (asynchronous) in front of it. As presented in Table 4, the detection performance of different encoding methods with the EBD–AGP method has been improved to different degrees. Regarding NSTS and NSTS_SAE, AP increased by 1.43% and 1.06%, respectively. As forecasted, by asynchronously reconstructing event frames, DVS images integrate richer event features, which making pedestrians are easier to be recognized by CNN. This is mainly because, in EBD–AGP, every event frame input to the detector is valid and has enough event information to accurately describe the motion state of the object, which ensures that the object in the region will not be missed or misidentified.

Table 4. The performance comparison of pedestrian detection accuracy and detection speed of event frames constructed by different encoding methods evaluated by EBD (event-based detector) and EBD–AGP (asynchronous event frame detection model), respectively (IOU0.5 AP).

	EBD		**EBD–AGP (Proposed)**	
	AP (%)	**Detect Time (ms)**	**AP (%)**	**Detect Time (ms)**
Frequency [10]	81.48	47.58	82.05	38.26
SAE [14,15]	82.81		84.08	
LIF [18]	78.92		80.84	
NSTS (proposed)	84.08		85.51	
NSTS–SAE (proposed)	86.37		87.43	

It is important to note that there is no notion of conventional frames in EBD–AGP. The detector is triggered once when the number of events in each independent grid region reaches the event count threshold. Therefore, the time interval between two times of detections is not fixed, which makes it difficult to directly determine the average detection time per image frame. Therefore, the average detection time per frame of EBD_AGP is calculated by using the ratio of the detection time of the whole test set and the number of event frames encoded in the baseline model. The calculation shows that the average detection time per frame for the asynchronous event-frame detection scheme is 38.26 ms, which is nearly 20% shorter than the 47.58 ms in EBD. Theoretically, the object detection time relates to the number of detection model operations. In EBD–AGP, the "sparse event" region is not detected as frequently as in EBD but only be detected once when enough events are accumulated.

As shown in Figure 6, the results from EBD–AGP were superior to those from EBD on the same reconstruction methods. It is noted that the event frame in the lower row accumulates richer details asynchronously, while the random white noise generated by the event camera also accumulates. However, the subsequent CNN detector is not sensitive to this noise, so it will not affect the detection accuracy.

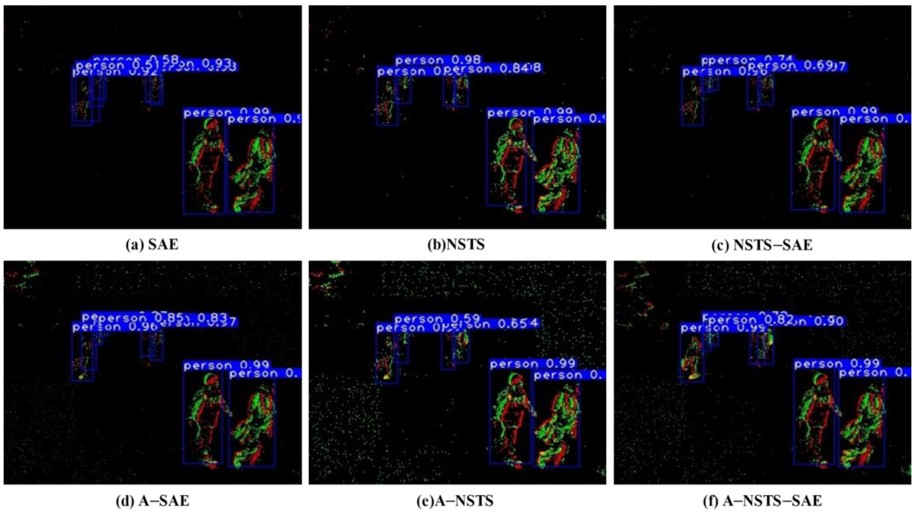

Figure 6. Some examples of visualization results of pedestrian detection. The outputs from EBD (event-based detector) are shown in the upper row, while outputs from EBD–AGP(asynchronous event frame detection model) are shown in the lower row. (**a**) SAE [14,15]; (**b**) NSTS; (**c**) NSTS–SAE; (**d**) A–SAE; (**e**) A–NSTS; (**f**) A–NSTS–SAE.

3.4. Discussion

As mentioned in the previous section, in the baseline architecture of EBD, the performance of our proposed NSTS reconstruction method is about 1.27–5.16% higher than previous work. Moreover, the NSTS–SAE method of fusing the two event frames of NSTS and SAE further significantly improved the AP of the detector by 86.37%, which is about 4.2–9.4% higher than previous reports. Although in EBD–GP, the AP of all cases is slightly reduced by -0.05%–0.34%, as shown in Table 3. But, as shown in Table 4, with the aid of the method of asynchronously constructing time surface in each grid, the AP of all cases increased by 0.57–1.92% in EBD–AGP. Compared with previous results, our final performance is about 5.6–10.8% higher. In terms of detection speed, benefiting from the idea of reusing feature maps with Grids Partition, our detector can reach about 26 FPS, which is nearly 20% higher than baseline. Most importantly, our proposed detection methods EBD–GP and EBD–AGP, have significant improvements in detection speed. Whose innovation is to exploit the sparsity and asynchrony of the event data by making a grid division of the input event frames to optimize the computational complexity of the detector. This enhancement is independent of the specific feature extraction network or detector and is a universal optimization method applicable to event-based object detection networks based on CNNs.

4. Conclusions

To improve pedestrian detection speed and accuracy, we presented a novel event-to-frame conversion method to integrate the inherent characteristics of the events more effectively, and an improved feature extracting network was designed that can reuse intermediate features to further reduce the amount of calculation. Based on these two approaches, we introduced an efficient pedestrian detection system with event data from DVS. After the validation on the custom dataset contains multiple real-world pedestrian scenarios, our proposed method is about 5.6–10.8% higher, and the detection speed is nearly 20% faster than the previous method reported. Finally, the pedestrian detector can run about 26 FPS at an AP of 87.43% on a single CPU, meeting the quasi-real-time requirements. We hope that our attempts will promote further research and application of DVS.

Author Contributions: Conceptualization, J.W., M.X.; Data curation, M.X., Z.H.; Formal analysis, Y.Z.; Investigation, J.W.; Methodology, L.T.; Project administration, H.W.; Writing—original draft, J.W.; Writing—review & editing, Z.H., L.T., X.Z., V.C., Y.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Key Research and Development Project (No. 2019YFC0117302), NSFC Youth Fund (No. 62004201), NSFC Youth Fund (No. 61704179), Shanghai Municipal Science and Technology Commission project (No. 19511131202), Pudong Economic and Technological Commission project (No. PKX2019-D02), Strategic Priority Research Program of Chinese Academy of Sciences (No. XDC02070700), Talents Project of Shanghai Advanced Research Institute Chinese Academy of Science (No. E052891ZZ1).

Data Availability Statement: The data presented in this study are available on request from the corresponding author.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Mao, J.; Xiao, T.; Jiang, Y.; Cao, Z. What can help pedestrian detection? In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 3127–3136.
2. Ye, M.; Shen, J.; Lin, G.; Xiang, T.; Shao, L.; Hoi, S.C. Deep learning for person re-identification: A survey and outlook. *IEEE Trans. Pattern Anal. Mach. Intell.* **2021**. [CrossRef] [PubMed]
3. Zhu, M.; Wu, Y. A Parallel Convolutional Neural Network for Pedestrian Detection. *Electronics* **2020**, *9*, 1478. [CrossRef]
4. Jung, J.; Bae, S.-H. Real-time road lane detection in urban areas using LiDAR data. *Electronics* **2018**, *7*, 276. [CrossRef]
5. Guo, Z.; Huang, Y.; Hu, X.; Wei, H.; Zhao, B. A Survey on Deep Learning Based Approaches for Scene Understanding in Autonomous Driving. *Electronics* **2021**, *10*, 471. [CrossRef]
6. Gallego, G.; Delbruck, T.; Orchard, G.; Bartolozzi, C.; Scaramuzza, D. Event-based Vision: A Survey. *arXiv* **2019**, arXiv:1904.08405. [CrossRef] [PubMed]
7. Leñero-Bardallo, J.A.; Serrano-Gotarredona, T.; Linares-Barranco, B. A 3.6$\mu $ s Latency Asynchronous Frame-Free Event-Driven Dynamic-Vision-Sensor. *IEEE J. Solid-State Circuits* **2011**, *46*, 1443–1455. [CrossRef]
8. Lakshmi, A.; Chakraborty, A.; Thakur, C.S. Neuromorphic vision: From sensors to event-based algorithms. *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.* **2019**, *9*, e1310. [CrossRef]
9. Haessig, G.; Benosman, R. A sparse coding multi-scale precise-timing machine learning algorithm for neuromorphic event-based sensors. In Proceedings of the Micro-and Nanotechnology Sensors Systems, and Applications X, Orlando, FL, USA, 15–19 April 2018; p. 106391U.
10. Chen, N.F. Pseudo-labels for supervised learning on dynamic vision sensor data, applied to object detection under ego-motion. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, Salt Lake City, UT, USA, 18–22 June 2018; pp. 644–653.
11. Li, J.; Dong, S.; Yu, Z.; Tian, Y.; Huang, T. Event-based vision enhanced: A joint detection framework in autonomous driving. In Proceedings of the IEEE International Conference on Multimedia and Expo (ICME), Shanghai, China, 8–12 July 2019; pp. 1396–1401.
12. Jiang, Z.; Xia, P.; Huang, K.; Stechele, W.; Chen, G.; Bing, Z.; Knoll, A. Mixed frame-/event-driven fast pedestrian detection. In Proceedings of the International Conference on Robotics and Automation (ICRA), Montreal, QC, Canada, 20–24 May 2019; pp. 8332–8338.
13. Chen, G.; Cao, H.; Ye, C.; Zhang, Z.; Liu, X.; Mo, X.; Qu, Z.; Conradt, J.; Röhrbein, F.; Knoll, A. Multi-cue event information fusion for pedestrian detection with neuromorphic vision sensors. *Front. Neurorobotics* **2019**, *13*, 10. [CrossRef] [PubMed]

14. Mueggler, E.; Bartolozzi, C.; Scaramuzza, D. Fast event-based corner detection. In Proceedings of the British Machine Vision Conference (BMVC), London, UK, 4–7 September 2017.
15. Mohamed, S.A.; Haghbayan, M.-H.; Heikkonen, J.; Tenhunen, H.; Plosila, J. Towards real-time edge detection for event cameras based on lifetime and dynamic slicing. In Proceedings of the Joint European-US Workshop on Applications of Invariance in Computer Vision, Ponta Delgada, Portugal, 9–14 October 1993; pp. 584–593.
16. Miao, S.; Chen, G.; Ning, X.; Zi, Y.; Ren, K.; Bing, Z.; Knoll, A. Neuromorphic Vision Datasets for Pedestrian Detection, Action Recognition, and Fall Detection. *Front. Neurorobotics* **2019**, *13*. [CrossRef] [PubMed]
17. Li, H.; Li, G.; Ji, X.; Shi, L. Deep representation via convolutional neural network for classification of spatiotemporal event streams. *Neurocomputing* **2018**, *299*, 1–9. [CrossRef]
18. Fang, W. Leaky Integrate-and-Fire Spiking Neuron with Learnable Membrane Time Parameter. *arXiv* **2020**, arXiv:abs/2007.05785.
19. Sironi, A.; Brambilla, M.; Bourdis, N.; Lagorce, X.; Benosman, R. HATS: Histograms of averaged time surfaces for robust event-based object classification. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 June 2018; pp. 1731–1740.
20. Yan, C.; Tu, Y.; Wang, X.; Zhang, Y.; Hao, X.; Zhang, Y.; Dai, Q. Stat: Spatial-temporal attention mechanism for video captioning. *IEEE Trans. Multimed.* **2019**, *22*, 229–241. [CrossRef]
21. Choi, E.; Bahadori, M.T.; Sun, J.; Kulas, J.; Schuetz, A.; Stewart, W. Retain: An interpretable predictive model for healthcare using reverse time attention mechanism. In Proceedings of the Advances in Neural Information Processing Systems, Barcelona, Spain, 5–10 December 2016; pp. 3504–3512.
22. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You Only Look Once: Unified, Real-Time Object Detection. In Proceedings of the Proceedings of the IEEE conference on computer vision and pattern recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 779–788.
23. Redmon, J.; Farhadi, A. Yolov3: An incremental improvement. *arXiv* **2018**, arXiv:abs/1804.02767.
24. Howard, A.; Sandler, M.; Chu, G.; Chen, L.-C.; Chen, B.; Tan, M.; Wang, W.; Zhu, Y.; Pang, R.; Vasudevan, V. Searching for mobilenetv3. In Proceedings of the IEEE International Conference on Computer Vision, Seoul, Korea, 27 October–2 November 2019; pp. 1314–1324.
25. Berner, R.; Brandli, C.; Yang, M.; Liu, S.-C.; Delbruck, T. A 240 $\times$ 180 120 db 10 mw 12us-latency sparse output vision sensor for mobile applications. In Proceedings of the International Image Sensors Workshop, Snowbird, UT, USA, 12–16 June 2013; pp. 41–44.
26. Loshchilov, I.; Hutter, F. Sgdr: Stochastic gradient descent with warm restarts. *arXiv* **2016**, arXiv:abs/1608.03983.