Adaptive Temporal Pooling for Object Detection using Dynamic Vision Sensor

Jia Li[1]
j0402.li@samsung.com
Feng Shi[1]
f0815.shi@samsung.com
Weiheng Liu[1]
weiheng.liu@samsung.com
Dongqing Zou[1]
dongqing.zou@samsung.com
Qiang Wang[1]
qiang.w@samsung.com
Hyunku Lee[2]
hyunku.lee@samsung.com
Paul-K.J. Park[2]
k.j.park@samsung.com
Hyunsurk Eric Ryu[2]
eric_ryu@samsung.com

[1] SAIT China Lab
SRC-Beijing
Samsung Electronics
Beijing, China

[2] LSI Product and Technology
System LSI Business
Samsung Electronics
South Korea

Abstract

Dynamic Vision Sensor (DVS) is a kind of asynchronous sensor which can capture visual information with low power consumption. Encouraged by the superior properties of DVS, we propose a novel object detection method for DVS. The imaging of DVS is highly related to the object motion speed. For sparse events induced by slow motions, directly applying existing object detection methods cannot obtain reliable features, leading to loss in performance. For this issue, we introduce a new model to extract motion invariant features for object detection using DVS, namely recursive adaptive temporal pooling. The model works in a recursive manner and adaptively integrates features in varied time interval into a holistic representation for current time-step with respect to the motion speed. The pooled features are then fed into a detection sub-network to produce the final results. Our framework is end-to-end trainable and is highly efficient benefitting from the recursive manner and the characteristics of DVS data. Our method is evaluated on a large DVS dataset with about 100k ground-truth images and compared with the state-of-the-art methods. Experiments demonstrate the effectiveness and efficiency of our method.

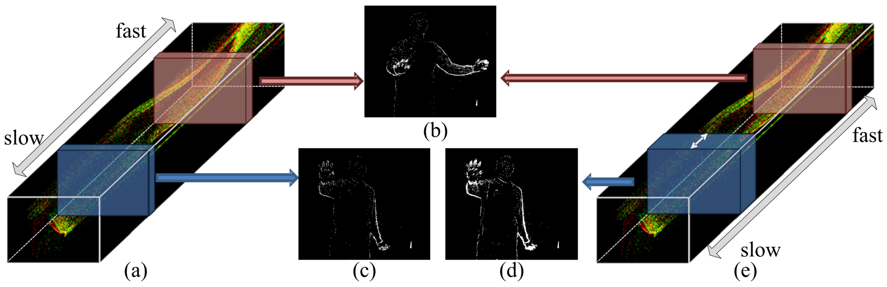

Figure 1: (a) (e) are the spatial-temporal volume of the same event stream, with each dot representing an event. The motion in the stream is slow at the first and gets fast at the end. (b)(c)(d) are DVS images with certain integration time interval. With the same integration time interval, the contours of the hands show clearly for fast motion (b) while very sparse events are generated for slow motion (c). However, within a longer time interval, the contours of the hands in slow motion become clear as shown in (d).

1 Introduction

Object detection is a fundamental topic in computer vision and plays an important role for a lot of applications such as auto driving, drone, smart home assistant and gesture based natural UI systems. For practical applications, power-efficient camera, algorithms with real time processing capability on power-efficient embedded processor and robust performance are crucial for widespread deployment.

Recently, Dynamic Vision Sensor (DVS) [2, 17] has gained more attention because of their potentials for computer vision tasks [1, 16, 23, 29, 30]. Inspired by human vision, DVS efficiently encodes visual information by recording pixel-level changes caused by movements in a scene instead of intensity images, and generates a stream of asynchronous events in microsecond precision. An image-like representation of DVS can be generated by accumulating events over a fixed time interval (Fig.1). Compared with conventional RGB sensor, DVS has much higher temporal resolution, providing a feasible way to track object in rapid motions. Moreover, its power-efficiency and sparse representation make it possible to develop highly efficient computer vision solution. All the above form its nature advantages for object detection.

Along with the superior properties of DVS, the event distribution of same object dramatically varies by different motion speed, as demonstrated by Fig.1. Within a fixed time interval, while the contour of hand in fast motion shows clearly, the hand in slow motion generates very sparse events. The information carried in the latter case is rather limited and it is hard for existing object detection methods [6, 7, 9, 18, 21, 22] to extract reliable feature for detection, leading to a loss in performance. However, the information of object for detection is adequate in DVS stream, only it spreads in a varied interval of temporal dimension with respect to the motion speed, as shown in Fig.1(b)(d). This suggests that it is crucial to exploit information in both spatial and temporal dimensions. Several methods have been proposed to incorporate temporal information for object detection [8, 12, 13]. However, they are based on a multi-stage framework. Within the multi-stage framework, it is hard to recover the errors in first stage especially when the detection bounding boxes output by first stage is not reliable.

In this paper, we aim to improve the detection performance for DVS by extracting motion invariant features. More specifically, for detection at each time-step, we extract features from a spatial-temporal volume using temporal pooling, as a holistic representation of its information. The temporal interval of the spatial-temporal volume should be dynamically determined with respect to motion speed. Meanwhile, to prevent importing ambiguity when aggregating information, the spatial misalignment cross frames caused by object motions should be considered.

Based on the above insights, we propose a novel end-to-end neural network model for object detection using DVS, named as Recursive Adaptive Temporal Pooling (RATP). It is designed to recursively pool the feature maps of successive images with a predicted weight map which indicates the consistency of the features being pooled at each spatial location. There are three components in our model. First, a feature map is extracted from each image using a feature extraction sub-network. Then, an adaptive temporal pooling module (Fig.2) is used to recursively pool the extracted feature maps according to a weight map predicted by a Multilayer Perceptron (MLP) network. At last, the pooled feature map is fed into a detection sub-network that produces final detection results. All the above three components are integrated in one deep neural network, so it can be optimized in an end-to-end manner. The major contributions of this paper are as follows:

1) We propose a novel adaptive temporal pooling module that (i) adaptively pools feature maps from successive images together, and produces motion invariant feature from DVS stream, and (ii) works in a recursive manner, i.e. requires less memory and computation cost at test time.

2) Based on the adaptive temporal pooling module, our object detection solution proposed for DVS is (i) end-to-end trainable along with DVS streams, the final objective of classification and localization, and (ii) efficient benefitting from the recursive manner of our module and the superior properties of DVS.

2 Related Work

Object Detection Object detection has obtained a significant boost benefiting from the rapid development of deep CNN in recent years [3]. Girshick et al. proposed R-CNN [7] for object detection with region proposal generation [25], CNN feature extraction, and region classification. For acceleration, Fast R-CNN [6] directly extracts the features of image patches from feature map of input image using ROI pooling, instead of feeding each image patch into CNN independently. For further speedup region proposal generation, Faster R-CNN [22] uses Region Proposal Network (RPN) to generate region proposals with convolutional features shared with Fast R-CNN.

Although great success has been achieved in object detection for still images, object detection in video remains a challenging problem. In this field, most methods rely on the off-the-shelf still-image object detector, thus are multi-stage pipelines. Han et al. [8] improve the frame-level detection along sequence via a re-scoring strategy. Kang et al. [13] generate tubelet proposals by propagating predicted bounding boxes along the time axis using tracking algorithm [26], and use a temporal convolutional network to improve detection results along the tubelets. In [14], Kang et al. propose a tubelet proposal network for generating tubelet proposals with better quality and efficiency. The tubelet features are then fed into LSTM for classification. In contrast, Our adaptive temporal pooling module exploits temporal information in feature-level and is end-to-end trainable.

Object Recognition with DVS Several methods are proposed to utilize temporal information for object recognition with DVS. Orchard et al. [20] propose a spiking hierarchical model which uses spike timing to encode the strength of neuron activation. Lagorce et al. [19] use a hierarchical time-surface feature extraction technique to build features using both spatial and temporal information. However, the problem that object appearance is dependent on motion speed is not addressed. The method proposed in [6] uses a dynamic time window with fixed number of events to eliminate the effect of motion speed on object appearance. But it cannot work when multiple objects move in different speed.

Temporal Pooling In computer vision, the idea of temporal pooling usually involves a layer in CNN used to aggregate frame-level features into a video-level feature. It is mainly used to convert videos with varying frame length into a fixed-length CNN feature representation which is convenient for further processing [15, 27]. In [4], temporal rank pooling is proposed to capture time varying information in videos for better classification. Our method is the first to utilize temporal pooling to handle the issue in object detection for DVS.

In addition, our framework has similar structure with the LSTM network [10]. The difference is that LSTM is designed to encode the temporal dynamics in video sequences, while our network is for learning the motion invariant feature for object detection with DVS.

3 Proposed Method

We now describe the proposed method in detail. The whole network consists of three major components: first, a convolutional sub-network is applied to extract feature map of each frame independently; then, an adaptive temporal pooling module is used to extract a representation robust to motion speed by aggregating feature maps of previous frames adaptively in a recursive manner; finally, classification labels and object bounding boxes are predicted by a detection sub-network based on the pooled feature map at each time-step.

3.1 Feature Extraction Network

To fully take advantage of the compact representation of DVS data, we design a computationally efficient network for feature extraction.

Compared to conventional dense RGB images, DVS images are sparse and binary. In DVS, all the regular, unchanging values are eliminated, leaving only the data with saliency changes, such as edges and salient movements. It is approximately as a regular image going through a high-pass filter then being binarized, leaving only mid and high frequency information need to be modeled. That means fewer kernels are required to represent DVS images than regular images. Additionally, since there is a large portion of image regions that are zeros in DVS image, applying kernels on those empty regions is non-informative. Large stride at frontal layers helps to get a more compact and meaningful description of sparse DVS image at early stage of the network, instead of spending much computation resource on the noisy details. Based on above observations, we design our feature extraction network as shown in Table 1. The number of convolution kernels of each layer is smaller than those in networks for RGB images. In first three layers, large strides greatly shrink the feature map, reducing the computation complexity of following layers.

Layers	Conv1	Pool	Conv2	Conv3	Conv4	Conv5
Param	8x8x8/4	4x4/2	4x4x12/2	4x4x12	4x4x16	4x4x64

Table 1: Network structure of feature extraction network.

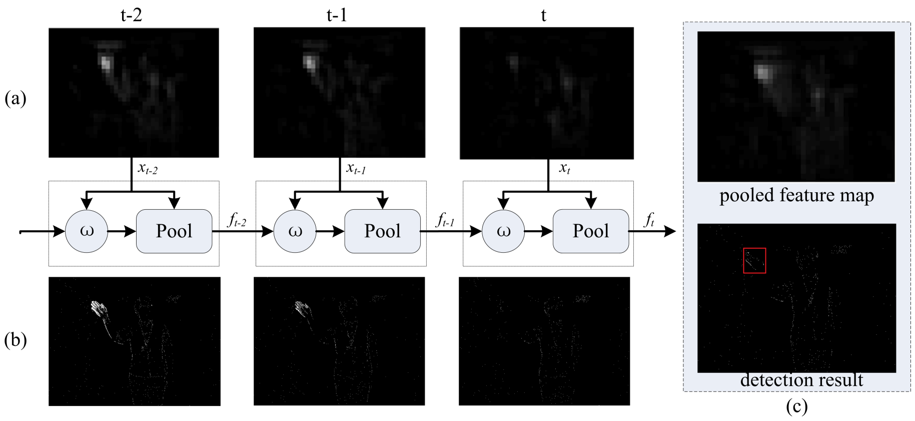

Figure 2: Row(a): feature map before adaptive temporal pooling. Row(b): original DVS images. (c): feature map after adaptive temporal pooling and final detection result. Since the hand in slow motion produces sparse events which are not discriminative from background noises, the activations at t are low (a) resulting detection failure. After adaptive temporal pooling, the activations of hand region get enhanced (c) and detection succeeds.

3.2 Adaptive Temporal Pooling

Adaptive temporal pooling is the key module of our model. The structure of the module is shown in Fig.2. The idea behind adaptive temporal pooling is as follows. DVS generates events in terms of contrast changes. It produces continuous response to rapid motions but very sparse response for slow motions. To mitigate this problem, adaptive temporal pooling module dynamically aggregates information from multiple frames within a varied time interval with respect to motion speed using pooling operation, and produces a holistic feature that is invariant to motion speed. A weight map is used to indicate the contribution from previous frames at each spatial location, to prevent inducing ambiguity from spatial misalignment. More specifically, at each location, the weight should be a small value if the feature of current frame is discriminative from the previously pooled feature; otherwise a large weight is used. We note that the weighted pooling behaves similarly to these models that based on attention mechanism [28]. It focuses on the relevant temporal range within DVS stream to detect target objects. The difference is that our model works in temporal dimension instead of spatial dimension, and is in a recursive manner instead of putting features in memory buffer and predicting weights for all the features together.

Let us denote the feature maps input to adaptive temporal pooling module as $\{x_1, x_2, ..., x_T\}$, $x_t \in \Re^{CxHxW}$, where t represents the index of frame and each x_t is obtained by the feature extraction sub-network $x_t = \mathbf{N}_{feat}(I_t)$. The aim is to compute the aggregated feature map $f_t \in \Re^{CxHxW}$ for each time-step. First, the weight map ω_t is computed using a weight prediction net based on the adaptive pooled feature f_{t-1} at previous time-step and feature map of current frame x_t:

$$\omega_{t,i} = \mathbf{N}_{mlp}(f_{t-1,i}, x_{t,i}). \tag{1}$$

Here, $\omega_{t,i} \in (0,1)$ and i is the location indices of vectors in feature map. $\mathbf{N}_{mlp}$ represents a MLP used to predict the each weight in ω_t with feature vectors at corresponding location in f_t and x_t. The MLP can be efficiently implemented as a fully convolutional model consists of 1x1 convolutional layers sequentially connected to each other with non-linear activation layers alternating in between. The kernel number of convolution layer corresponds to the number of hidden nodes of the MLP. A sigmoid layer is used at the end of the MLP to make sure the value of weights is between 0 and 1. As input of the weight prediction net, we use the residual of f_t and x_t, since residual gives better performance compared to simple concatenation (Sec. 4.3).

Then, with the predicted weight map, a new pooled feature map f_t is computed by pooling the previously pooled feature map f_{t-1} and the feature map corresponding to current frame x_t. The formulation is

$$f_t = \rho[\mathbf{s}(f_{t-1}, \omega_t), x_t]. \tag{2}$$

Here, $\mathbf{s}(\cdot)$ is a function that scales the values in pooled feature map f_{t-1} with predicted weight map ω_t. The weight at a specific spatial location is shared for all the channels of the feature map f_{t-1}. And, $\rho(\cdot)$ represents the pooling function.

3.3 Training and Inference

With the feature map extracted from feature extraction net and adaptive temporal pooling, we follow the state-of-the-art Faster-RCNN detection framework and train our model in the approximate joint training scheme. For both RPN and Fast-RCNN, the loss function is defined as summation of the cross-entropy loss and the box regression loss $\mathbf{L}(s, t_{x,y,w,h}) = \mathbf{L}_{cls}(s, c^*) + \lambda[c^* > 0]\mathbf{L}_{reg}(t, t^*)$. Here c^* is the ground-truth label, $\mathbf{L}_{cls}(s, c^*)$ is the cross-entropy loss for classification, $\mathbf{L}_{reg}(t, t^*)$ is the bounding box regression loss as defined in [22], and t^* represents the ground truth box. $[c^* > 0]$ is an indicator which equals to 1 if the argument is true and 0 otherwise.

In inference phase, our RATP model loops over all the frames of DVS stream in an online manner and predicts detection labels and bounding boxes at every time-step. At each time-step, the incoming frame first goes through the feature extraction net. Then the extracted feature map, together with the pooled feature map from previous time-step, is input into the temporal pooling module and pooled together with respect to the predicted weight map. At last, the pooled feature map is fed into the detection sub-network producing final detection results and saved in memory for the processing of next time-step. As for runtime complexity, our method is computationally efficient. Compared to baseline detection model, the only additional computational cost of our method is the MLP for weight map prediction and the pooling operation in adaptive temporal pooling module.

4 Experimental Results

4.1 Dataset

Our DVS dataset for training contains 264 DVS sequences total 259,681 VGA size images. There are one to seven persons in each sequence, making various gestures such as waving hand, circling hand, waving finger, etc. in various speed and under different illuminations. The distance between target objects and DVS camera ranges from 1.5 to 3 meters with hand size varying from 20×20 to 100×100. All of the sequences are with annotations of human hand bounding box. In these DVS sequences, the hands represent dramatic deformation while gesturing and the event distribution differs with hand motion speed and illumination conditions. In addition, the recording scenario includes flicking monitors, light switching and walking around people etc., which make background cluttered. All the above factors make the dataset particularly challenging for detection task. The DVS dataset for testing contains 93 DVS sequences which are shot in the same settings as training dataset.

Our dataset is created with the aid of well-developed Kinect sensor. The Kinect and DVS are calibrated spatially and temporally, and an IR filter is attached in front of DVS lens to filter out the IR light generated by Kinect. The hand segmentation on depth image of Kinect is projected on DVS with its generation timestamp. The hand segmentation mask forms a volume in spatial-temporal space, and its temporal length is set as the frame interval of Kinect. The events within the volume are labelled as hand, and morphology transformation techniques are used to recover the miss-labelled events if there is any. Based on the labelled event stream, the dataset can generate images of DVS with ground-truth annotation within any time interval, which is set to 20ms in this paper.

Although our dataset is created for human hand detection which is a special case for general object detection, it is reasonable to infer that for deep network based algorithms, methods effective for one specific object class possibly work for other object classes. Dataset creation and method evaluation for general object detection will be our future work.

4.2 Experiment Settings

Evaluation Metrics For each predicted bounding box, if the maximum IoU is larger than an IOU threshold, it is regarded as a true positive, otherwise, it is a false alarm. We use average precision and recall (P&R) and average IOU score of true positive (AvgIOU) as the video object detection metric, and evaluate all detection models in an end-to-end manner.

Implementation Details We take the full images in dataset in a resolution of 640×480 pixels as input. All the models in our experiments are trained from scratch with same initialization method, since DVS images are totally different from conventional RGB images and there is no proper pre-training model for initialization. The training process starts with a learning rate of 1e-4 using a stair case decay of 0.9 applied after 19 epochs. We use a batch size of 32. For Adam, we use $\beta 1 = 0.9$ and $\varepsilon = 0.1$. For data augmentation, we mirror each training image horizontally.

We use Faster-RCNN detection framework in our experiments. For feature extraction sub-network, we use the structure shown in Table 1. For the computation-efficiency sake, the MLP for weight map prediction is simplified and implemented as a one 1x1 convolutional layer with sigmoid activation. The initial state of the pooled feature map is set as all zero. The detection net is trained using the hard example mining technique [24] with additional heuristic that ROIs without any event are removed in training stage since they are non-informative. The fg-and-bg ratio is set as 1/3. Our method is implemented based on the Caffe framework [11], and trained on NVIDIA K80 GPU.

IOU threshold	0.3	0.5	0.7	Average
(a) single-frame based FRCNN	73.69%	73.37%	66.55%	71.20%
(b) Pooling with constant decay factor	73.23%	67.62%	43.02%	61.29%
(c) Adaptive pooling + concatenation	79.49%	78.30%	68.01%	75.27%
(d) Adaptive pooling + residual	79.73%	79.11%	69.26%	76.03%

Table 2: Performance of baseline model and variants of our method.

Method	P&R$^{IOU=.3}$	AvgIOU
RATP	79.73%	0.663
LSTM	79.78%	0.687
RATP+LSTM	84.05%	0.682

Table 3: Comparison with LSTM[10].

Method	P&R$^{IOU=.3}$	AvgIOU
RATP	79.73%	0.663
Seqnms	80.06%	0.810
RATP+Seqnms	86.53%	0.811

Table 4: Comparison with SeqNMS[8].

4.3 Results

Ablation Study We explicitly investigate different settings of the proposed method and compare it with baseline model in this section. We perform a test on single-frame baseline model first. For a fair comparison, we implement the baseline method using the same deep network as our method except for the adaptive temporal pooling module. Then, we compare the performance of the baseline with our method and its variants, as listed in Table 2. More specifically, Model (a) is the baseline Faster-RCNN model trained and tested on single DVS image. Model (b) is a degenerated variant of our method that uses a naive approach to pool feature maps together. Instead of a predicted weight map, Model (b) imposes a constant decay factor $\omega_c \in (0,1)$ on the pooled feature map (0.9 in our experiment), i.e. the weight map ω_t in Eq.(1) and Eq.(2) is set to all ω_c at each time-step. Model (c) is the proposed recursive temporal pooling model, but takes input of the MLP network $\mathbf{N}_{mlp}$ as the concatenation of pooled feature $f_{t-1,i}$ and feature from current frame $x_{t,i}$. Model (d) is the recursive temporal pooling model with residual of $f_{t-1,i}$ and $x_{t,i}$ as input of $\mathbf{N}_{mlp}$. It is obvious that the proposed recursive temporal pooling model achieves significant improvement over Model (a)(b). For Model (b), the average P&R decreases to 61.29%. This indicates that the spatial misalignment across frames caused by motion can degrade the performance even with a decay factor. Results of Model (c)(d) show that using the residual of feature maps further improves the results. We conjecture that this due to the fact that residual reflects the difference of feature map more explicitly and allows our model to identify the discriminations between previously pooled features and the features of current frame.

We also visualize the effect of temporal pooling model as shown in Fig.3. From Fig.3 we can see that the feature map cannot be activated for the hand with sparse events based on single frame, since the network cannot extract features discriminative enough to background noises for the sparse hand. With temporal pooling module the feature map can get continuous responses even at the sparse hand region by pooling features from previous frame together. More qualitative comparison of baseline and our method is shown in Fig.3.

Comparison with LSTM Our RATP module shares similar structure with LSTM [10]. Here, we explicitly investigate the difference of LSTM with our method. The comparison results are shown in Table 3. For comparison, we replace the RATP module in Model (d) with LSTM. More specifically, we build a LSTM at each spatial location in the feature map,

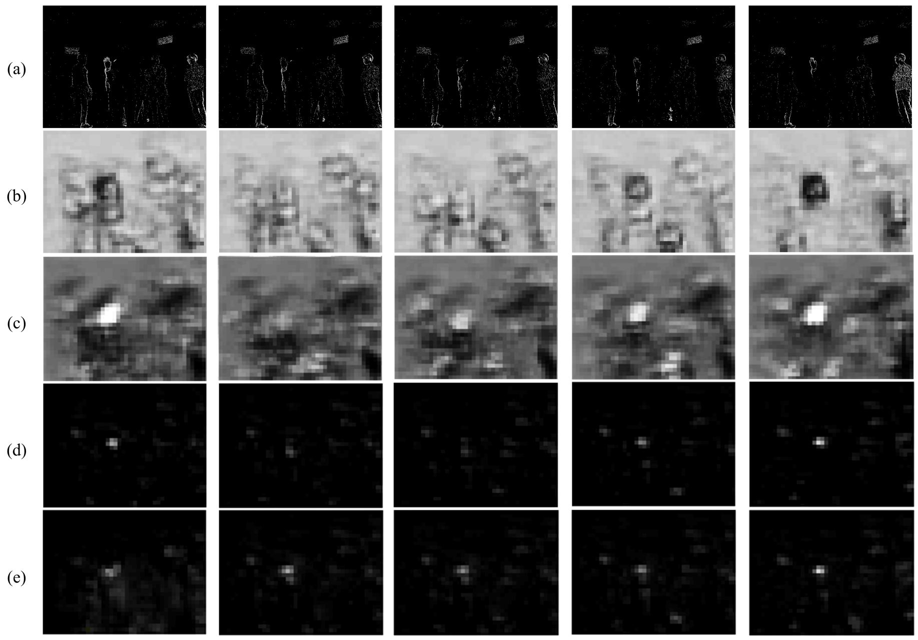

Figure 3: (a) the original DVS images, (b) the weight map of RATP, (c) the visualization of forget gates in LSTM[10], (d) feature maps before RATP, and (e) feature maps after RATP. For all the visualizations, brighter pixel means larger value, otherwise smaller value.

and parameters of all the LSTMs are shared. The input of a LSTM is the feature vector at corresponding spatial location of the feature maps that extracted from multiple successive frames. As shown in Table 3, the performance of LSTM model is slightly better than our RATP. However, they improve the baseline model from different aspects. While our model extracts a motion invariant representation for the target object, LSTM attempts to model the dynamic structure of the motion. In Fig.3, Fig.3(b) visualizes the predicted weight map of our RATP module, and Fig.3(c) visualizes the value of forget gate in the LSTM which works similarly with the weight map of RATP. It is obvious that for the regions that hands in fast motion, the weight map in our model tries to filter the impact from previous frame out to prevent importing ambiguity by aggregating misaligned features. By contrast, LSTM emphasizes those regions, as the feature vectors from the motion region are required for modelling motion dynamics. Also, we train a model that combines our RATP with LSTM by stacking RATP on top of LSTM as two sequential layers. It further improves the performance to 84.05%, showing that the information modelled by LSTM and our model is different and complementary.

Comparison with Seq-NMS We further compare our method with the state-of-the-art method proposed for task of object detection in video. While our method aims to extract more reliable features for target object from DVS stream, Seq-NMS [8] attempts to refine the given object candidate boxes by exploiting temporal constraint. In Table 4, we show the results with our method, Seq-NMS, and their combination. Seq-NMS improves the baseline by large margin, showing its effectiveness. However, the integration of our RATP module with Seq-NMS obtains the largest gain, and the performance is further improved to 86.53%,

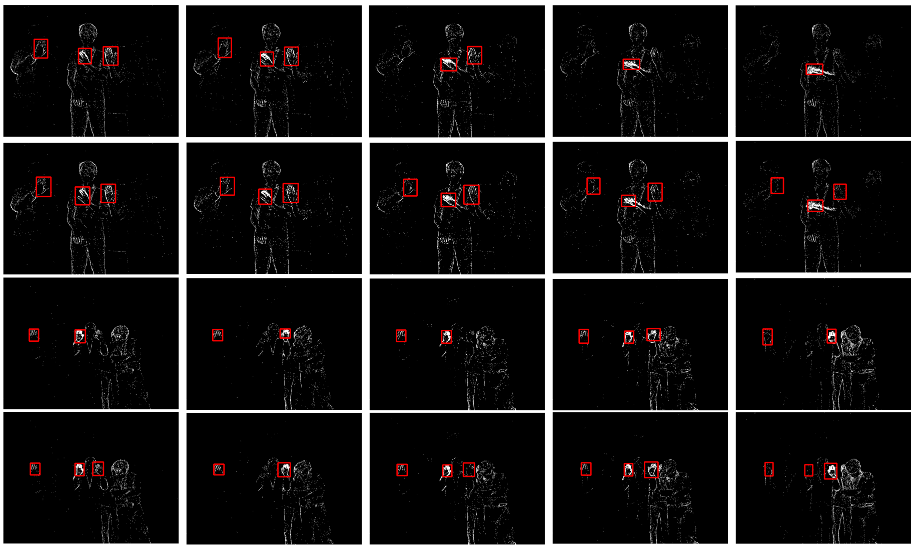

Figure 4: Object detection results from two different test sequences. In each pair of rows, the top row shows the detection results of baseline model and the bottom row shows our method. In baseline model, the hands in slow motion often missed since they produces very sparse events, while our method performs better for these cases.

halving the error rate of baseline.

Runtime Analysis We test the runtime of our method on single Core of Intel i5-4590. By fully taking advantage of the superior properties of DVS, our method can achieve a speed of 11.6ms per frame. That means it is feasible to apply the whole detection system with DVS and our detection method to applications with embedded systems.

5 Conclusion and Future Work

This paper proposes an efficient, end-to-end object detection method for DVS. Our method pools features within varied time interval together and generates features invariant to motion speed. Since its improvement is in feature level, it can be geared to existing deep neural network based detection method as a complementary technique to further improve detection performance. Experiments on a large DVS dataset demonstrate its efficiency and effectiveness. For next work, the DVS features will be further investigated with information carried in polarity and timestamp in DVS events. Also, currently we perform experiments on DVS dataset for hand detection as a special case of general object detection. As the next step we plan to build a more general dataset with objects in more categories for further evaluation.

References

[1] Patrick Bardow, Andrew J Davison, and Stefan Leutenegger. Simultaneous optical flow and intensity estimation from an event camera. *Proc. CVPR*, pages 884–892, 2016.

[2] Kwabena Boahen. Neuromorphic chips. *Scientific American*, 292:38–46, 2005.

[3] Mark Everingham, Luc Van Gool, Christopher K I Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2):303–338, 2010.

[4] Basura Fernando and Stephen Gould. Learning end-to-end video classification with rank-pooling. *Proc. ICML*, pages 1187–1196, 2016.

[5] Rohan Ghosh, Abhishek Mishra, Garrick Orchard, and Nitish V. Thakor. Real-time object recognition and orientation estimation using an event-based camera and cnn. *IEEE Biomedical Circuits and Systems Conference*, pages 544–547, 2014.

[6] Ross Girshick. Fast r-cnn. *Proc. ICCV*, pages 1440–1448, 2015.

[7] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *Proc. CVPR*, pages 580–587, 2014.

[8] W. Han, P. Khorrami, T. L. Paine, P. Ramachandran, M. Babaeizadeh, H. Shi, J. Li, S. Yan, and T. S. Huang. Seq-nms for video object detection. arXiv: 1602.08465, 2016.

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *Proc. ECCV*, pages 346–361, 2014.

[10] Sepp Hochreiter and Jurgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[11] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. *ACM Multimedia*, pages 675–678, 2014.

[12] K. Kang, W. Ouyang, H. Li, and X Wang. T-cnn: Tubelets with convolutional neural networks for object detection from videos. arXiv: 1604.02532, 2016.

[13] Kai Kang, Wanli Ouyang, Hongsheng Li, and Xiaogang Wang. Object detection from video tubelets with convolutional neural networks. *Proc. CVPR*, pages 817–825, 2016.

[14] Kai Kang, Hongsheng Li, Tong Xiao, Wanli Ouyang, Junjie Yan, Xihui Liu, and Xiaogang Wang. Object detection in videos with tubelet proposal networks. *Proc. CVPR*, 2017.

[15] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Feifei. Large-scale video classification with convolutional neural networks. *Proc. CVPR*, pages 1725–1732, 2014.

[16] Hanme Kim, Stefan Leutenegger, and Andrew J Davison. Real-time 3d reconstruction and 6-dof tracking with an event camera. *Proc. ECCV*, pages 349–364, 2016.

[17] Patrick Lichtsteiner, Christoph Posch, and Tobi Delbruck. A 128 $\times$ 128 120 db 15 μ s latency asynchronous temporal contrast vision sensor. *IEEE Journal of Solid-state Circuits*, 43(2):566–576, 2008.

[18] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, S Reed, Chengyang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. *Proc. ECCV*, pages 21–37, 2015.

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

[20] Garrick Orchard, Cedric Meyer, Ralph Etienne-Cummings, Christoph Posch, Nitish Thakor, and Ryad Benosman. Hfirst: A temporal approach to object recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(10):2028–2040, 2015.

[21] Joseph Redmon, Santosh K Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. *Proc. CVPR*, pages 779–788, 2015.

[22] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 91–99, 2015.

[23] Sajjad Seifozzakerini, Wei-Yun Yau, Bo Zhao, and Kezhi Mao. Event-based hough transform in a spiking neural network for multiple line detection and tracking using a dynamic vision sensor. *Proc. BMVC*, 2016.

[24] Abhinav Shrivastava, A K Gupta, and Ross Girshick. Training region-based object detectors with online hard example mining. *Proc. CVPR*, pages 761–769, 2016.

[25] Jasper Uijlings, K E Sande, T Gevers, and Arnold Smeulders. Selective search for object recognition. *International Journal of Computer Vision*, 104(2):154–171, 2013.

[26] Lijun Wang, Wanli Ouyang, Xiaogang Wang, and Huchuan Lu. Visual tracking with fully convolutional networks. *Proc. ICCV*, pages 3119–3127, 2015.

[27] Peng Wang, Yuanzhouhan Cao, Chunhua Shen, Lingqiao Liu, and Heng Tao Shen. Temporal pyramid pooling based convolutional neural networks for action recognition. *arXiv preprint arXiv:1503.01224*, 2015.

[28] Kelvin Xu, Jimmy Ba, Ryan Kiros, Aaron Courville, Ruslan Salakhutdinov, Richard Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. *Proc. ICML*, pages 2048–2057, 2015.

[29] Dongqing Zou, Ping Guo, Qiang Wang, Xiaotao Wang, Guangqi Shao, Feng Shi, Jia Li, and Paul K. J. Park. Context-aware event-driven stereo matching. In *IEEE International Conference on Image Processing*, pages 1076–1080, 2016.

[30] Dongqing Zou, Feng Shi, Weiheng Liu, Jia Li, Qiang Wang, Paul K. J. Park, Chang-Woo Shin, Yohan J. Roh, and Hyunsurk Eric Ryu. Robust dense depth map estimation from sparse dvs stereos. In *BMVC*, 2017.