Combined frame- and event-based detection and tracking

Hongjie Liu[1], Diederik Paul Moeys[1], Gautham Das[2], Daniel Neil[1], Shih-Chii Liu[1], Tobi Delbrück[1]
[1]Institute of Neuroinformatics, University of Zürich and ETH Zürich, Switzerland
[2]Intelligent Systems Research Center, University of Ulster, Ireland

***Abstract*— This paper reports an object tracking algorithm for a moving platform using the dynamic and active-pixel vision sensor (DAVIS). It takes advantage of both the active pixel sensor (APS) frame and dynamic vision sensor (DVS) event outputs from the DAVIS. The tracking is performed in a three step-manner: regions of interest (ROIs) are generated by a cluster-based tracking using the DVS output, likely target locations are detected by using a convolutional neural network (CNN) on the APS output to classify the ROIs as foreground and background, and finally a particle filter infers the target location from the ROIs. Doing convolution only in the ROIs boosts the speed by a factor of 70 compared with full-frame convolutions for the 240x180 frame input from the DAVIS. The tracking accuracy on a predator and prey robot database reaches 90% with a cost of less than 20ms/frame in Matlab on a normal PC without using a GPU.**

Keywords— event-based tracking and detection, DVS, DAVIS, particle filtering, Convolutional Neural Network.

I. INTRODUCTION

The DAVIS [1] is a neuromorphic camera that outputs static active pixel sensor (**APS**) image frames concurrently with dynamic vision sensor (**DVS**) temporal contrast events [2]. DVS address-events (**AE**s) asynchronously signal changes of brightness in the scene. The application of Convolutional Neural Networks (**CNN**s) to tracking have become widespread and there are a large number of tools available for application of this technology. Our aim in this study was to develop an object tracking system that uses DVS events to guide efficient application of CNN technology to DAVIS sensors and to demonstrate benefits from such a system.

Object tracking has been studied for many years. Speed versus accuracy are traded off in conventional frame-based visual tracking. Reported CNN based trackers (**CNT**) (e.g. [3], [4]) are usually either too slow or too expensive for real-time applications. Increasing attention has been on tracking using event-based vision sensors [5]–[13]. These event-based tracking systems can achieve relatively high accuracy and high speed due to the low-latency and sparse data output from the DVS. However, these DVS trackers usually do not deal with the scenario of tracking a moving object on a moving background, because the distinction between the events generated by object and ego movement is a hard task. Citation [14] proposes an algorithm for separating the two kinds of events, however their algorithm has only been tested with simple simulated environments and realistic scenarios are as yet unproven.

This work aimed to develop a tracker that can perform tracking in an ego-motion scenario where both the observer and the object move around a cluttered environment. We used a sliding window CNN which labels the input image with a likelihood score to perform classification-based detection of the object. To reduce the number of convolutions and thus the computational cost of each frame, we use regions of interest (**ROI**s) generated by the DVS event output of the DAVIS camera. The detected ROIs are fed to a particle filter which improves the tracking accuracy due to misclassification by the CNN.

The main contributions of this paper are as follows: combined event- and frame- based detection and an efficient way of combining a CNN and particle filtering to solve a complicated tracking task. The tracking system could be applied to robotics, unmanned aerial vehicles, search and rescue, etc. The rest of this paper is organized as follows. Sec. II describes the tracking method and dataset used; Sec. III presents the experimental results; and Sec. IV presents concluding remarks.

II. DATASET AND TRACKING METHOD

A. Dataset

This work is based on a robot predator and prey dataset which was recorded at University of Ulster's Intelligent Systems Research Centre ("Ulster dataset"). The DAVIS sensor was mounted on the top of a Pioneer four-wheeled robotic platform (the predator robot) and followed a second Pioneer robot (the prey robot) (Fig. 1). The prey robot, which is manually controlled with a joystick, was mounted with visual targets, similar to QR codes, suitable for tracking using available OpenCV toolboxes. By using OpenCV for tracking the tags, the predator robot could autonomously follow the prey (or go into a search rotation when the prey was lost) while recordings with the DAVIS sensor were made with the on-board computer running jAER [15], the software to process DAVIS data. (No QR tag search algorithm was used by our tracking algorithm)

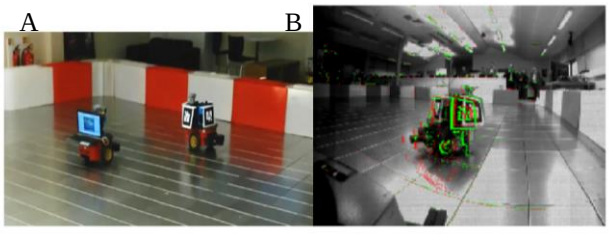

Fig. 1. **A**: The data collection arena in Ulster with the predator on the right and prey on the left. **B**: An example of the recorded data showing frames and AEs. Green and red pixels correspond to ON and OFF DVS events. The exposure time is about 20ms. The DVS event rate varied from 10k events per second (**eps**) to 300keps with a mean of about 100keps. The APS frame rate was about 8.1Hz. The DAVIS sensor had 240x180 18.5um x 18.5um pixels and the lens had a focal length of 2.1mm.

This work is supported by the European Union funded project SEEBETTER (FP7-ICT-2009-6), VISUALISE (FP7-ICT-2011.9.11), the Swiss National Science Foundation through the NCCR Robotics, ETH Zurich, and the University of Zurich.

The Ulster dataset consists of 20 minutes of data with 9k APS frames and 160 million DVS events. From this recording, the ground truth about the position of the prey robot at a time resolution of about 5ms was hand-labelled by capturing the cursor position in jAER while following the prey with the mouse. These locations at the various timestamps, together with the APS frames and DVS events constitute the database on which the following experiments were performed.

B. System architecture

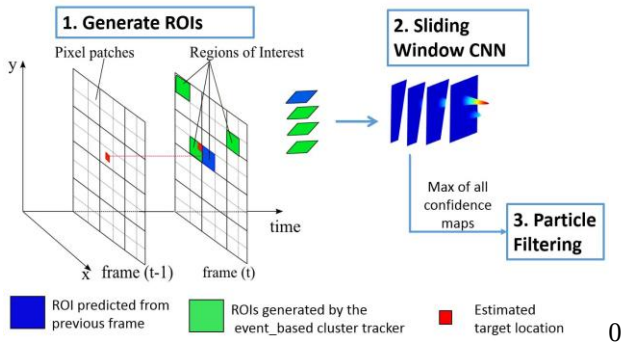

Fig. 2. Tracking architecture.

Fig. 2 shows the architecture of the tracking system. It includes an event-based cluster tracker (RectangularClusterTracker [6]) to generate a maximum of 3 ROIs based on local spatio-temporal coherence of the events, i.e. clusters tend to follow high-contrast compact features that generate correlated event streams. The frame-based sliding window convolution performs classification-based detection in the ROIs. A fourth ROI is generated around the estimated location of the robot in the previous frame. The sliding window convolution generates a confidence map ("heatmap") based on the classification result. The most likely location of the updated confidence map is later fed into the particle filter that gives the final estimation of the location of the prey robot.

C. Generating ROIs based on event clusters

Fig. 3. Regions of interest generated by the cluster tracker for two different frames.

Fig. 3 shows the output of the cluster tracker ([6] and [13]) at two times (green boxes). The cluster tracker is only based on events. It assigns incoming events to the nearest existing cluster or creates a new cluster for this event if no cluster is present in its neighborhood. The clusters die out when there is no new event to support them within a threshold time interval. A cluster is visible only when it receives a certain number of events. The event-based cluster tracker is a good approach for tracking isolated compact objects in a case without ego motion. Because too many events are generated by the cluttered background in our tracking scenario, we only use the tracker to generate ROIs that indicate possible locations of the object. For example, in the Fig. 3 left image the prey robot is captured by one of the regions, however, in the Fig. 3 right image, due to slow relative movement between the predator and prey robots compared to the background movement, too few events are generated by the prey. Therefore, it is not detected by the cluster tracker. Therefore we added another ROI centered at the prey location of the previous frame to decrease the detection failure rate.

D. CNN-based classification and detection

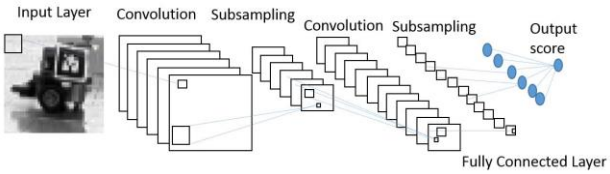

Fig. 4. CNN classification to generate likelihood scores.

Fig. 4 shows the architecture of the CNN used in this work [16]. It has two convolutional layers with 6 and 12 feature maps, both with kernel size 5x5, and two 2x2 subsampling layers that do average pooling. The input image size is 36x36 pixels with examples in Fig. 5. The network is trained with 373 positive and 1500 negative samples from DAVIS APS images (training set). The positive samples are 36x36 pixel cropped pixel patches centered at the ground truth target locations. The negative samples are cropped pixel patches of equal size randomly sampled from the rest of the image frame. The classification error is 2% on test frames (different from training set) in the recorded video while the chance error rate is 20% which is given by the ratio of positive test samples and total test samples.

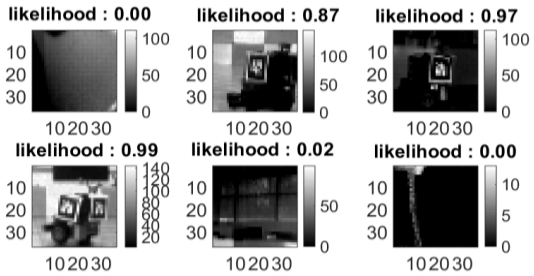

Fig. 5. CNN classification results on example gray level input patches. The likelihood that the image patch contains the robot is given above each frame.

Fig. 5 shows examples of the classification results from the CNN. The target likelihood scores above each frame are used to generate a confidence heatmap over the frame (Fig. 6). In the first frame, the classification is done over the entire frame with a defined stride size to generate the initial map. If the maximum score of the frame is below a certain threshold, it usually means that the robot is out of view, and then we perform a full frame convolution on the next frame until we

get a high peak of the confidence map. We then do convolution only over the ROIs for subsequent frames. Fig. 6 shows the smoothed confidence map generated from a sliding window CNN for a sample frame. The first peak indicates the most likely position of the robot whereas the second peak in this frame is generated by a part of the background which has similar features.

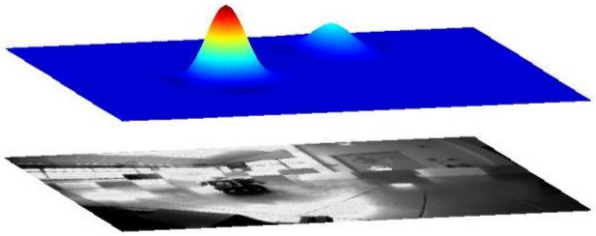

Fig. 6. Smoothed confidence map over the tested frame. The color code from red to blue means confidence from high to low (0.97 to 0).

E. Particle filter tracking

The tracking algorithm is based on particle filtering [17]. We define $Z_t = \{z_1, \ldots, z_t\}$ as the observation inputs of the particle filter up to frame t, where $z_t = (x_t^m, y_t^m, x_t^m - x_{t-1}^m, y_t^m - y_{t-1}^m)$ and (x_t^m, y_t^m) are maximum heatmap location of the frame t; i.e. each observation consists of a position and a position shift (velocity) from the previous frame. We then determine the posterior probability by recursively applying the Bayes' theorem as in (1):

$$p(s_t \mid Z_t) \propto p(z_t \mid s_t) \int p(s_t \mid s_{t-1}) p(s_{t-1} \mid Z_{t-1}) ds_{t-1} \quad (1)$$

where $s_t = (x_t, y_t, \dot{x}_t, \dot{y}_t)^T$ is the target state with location (x_t, y_t) and velocity $(\dot{x}_t, \dot{y}_t)$. $p(z_t \mid s_t)$ is the observation model that estimates the likelihood of observing z_t in the state s_t. The variances of the location and the velocity are given by $Q_t = (\sigma_x, \sigma_y, \sigma_{\dot{x}}, \sigma_{\dot{y}})$. The parameters of Q_t are estimated from the training frames. The observation z_t is the location and shift from the previous frame of the peak over all the ROI confidence maps for frame t. Given particle i, the predicted state of each particle is updated according to the dynamic model with noise Q_t resulting in particles $\hat{s}_t^i$. Next the observation noise R_t is added to each particle to get the predicated observations $\hat{z}_t^i$. The weights of the particles are computed from (2) where w_t^i is the weight for particle i at time t, and p_e is the Gaussian distribution with variances R_t. Thus particles that are closer to the newest measurement are more heavily weighted.

$$w_t^i = w_{t-1}^i p(z_t \mid s_t^i) = w_{t-1}^i p_e(z_t - \hat{z}_t^i), \quad i = 1, 2, \ldots N \quad (2)$$

Particles are resampled when the number of effective particles is less than a threshold number $N_{\text{eff}} < N_{\text{th}}$.

The final estimate of the target position and velocity is the weighted mean of all the particles (1000 for this work). A snapshot of the particle filter state is shown in Fig. 7.

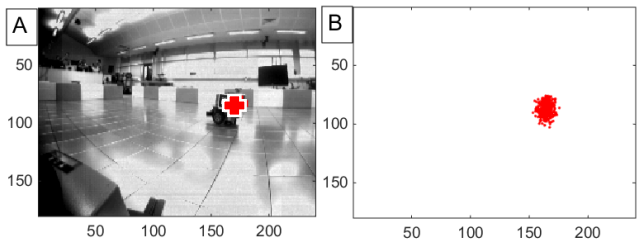

Fig. 7. **A**: Estimated location of the target in one frame (red plus). **B**: Locations of the 1000 particles.

III. RESULTS

The tracking algorithm was implemented in Matlab and tested on the Ulster dataset.

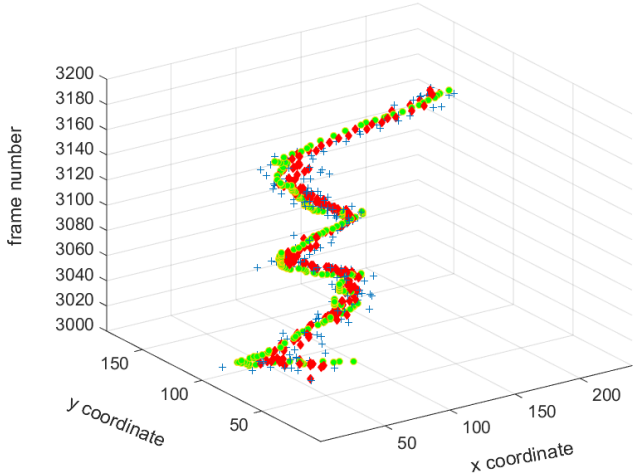

Fig. 8. Trajectory of the prey robot over 200 frames indicated by the ground truth position (green circle), the frame and event-ROI method without particle filtering (blue plus) and the particle filter method (red diamond).

Fig. 8 shows the estimated target location over 200 frames in 24.7s. The green circles are the ground truth. The particle filtered tracking result (red diamond), achieves higher accuracy than the one without particle filtering (blue plus).

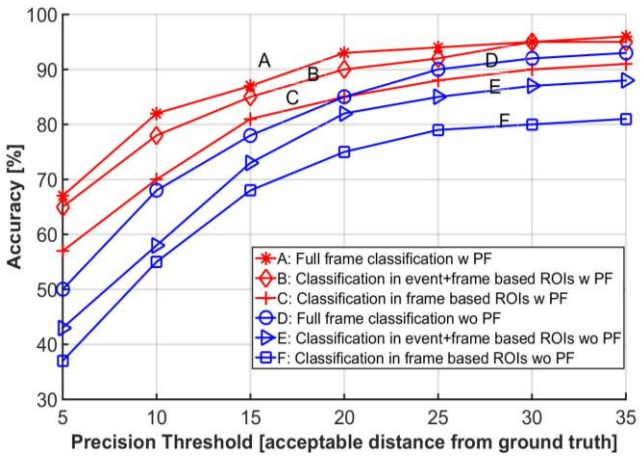

Fig. 9. Tracking accuracy vs the allowed distance error in pixels for methods A to F in Table 1.

Fig. 9 shows the tracking accuracy versus the allowed error distance. We use this precision plot [18] to measure the performance of our algorithm. It shows the percentage of frames whose estimated target location is within a given threshold distance (the precision) of the ground truth. The highest-accuracy method (method A in Table 1) is achieved by full frame convolution with particle filtering (PF) using 1000 particles. The method that does convolution only in the ROIs (method B) achieves slightly lower accuracy than the method that does full frame convolutions (method A). When we replace the event-generated ROIs by additional area around the previous target location estimate (C) the accuracy is lower than using event- and frame- generated ROIs(B). Using the particle filter (method B) clearly improves on just using ROIs (methods D, E and F).

TABLE 1. COMPUTATION TIME AND TRACKING ACCURACY FOR DIFFERENT METHODS. DISTANCE THRESHOLD IS SET TO 20 PIXELS FOR THE ACCURACY MEASUREMENT. PF IS PARTICLE FILTERING.

Tracking Methods	Computation Time/Frame	Accuracy
A: Full Frame Classification with PF	1.45s	93%
B: Event- and frame-based ROIs with PF	0.020s	90%
C: Frame-based ROIs with PF	0.018s	85%
D: Full Frame Classification without PF	1.25s	85%
E: Event- and frame-based ROIs without PF	0.016s	82%
F: Frame-based ROIs without PF	0.015s	75%

Table. 1 compares the computational cost and accuracy of the four methods using a threshold error of 20 pixels. With ROIs, the speed is boosted by 70X comparing to that of doing sliding window convolution over the whole image with a stride of 5 pixels, although the accuracy slightly drops from 85% to 82%. With particle filtering, the accuracy is improved from 82% to 90%. The algorithm was implemented on a Quad core Intel 3.5Ghz PC consuming 15% CPU and 30% of the 8G RAM while the full frame convolution method consumes 30% CPU and 55% of the RAM, though the implementation is not yet optimized.

IV. CONCLUSION

This paper proposes a more efficient detection and tracking algorithm based on a CNN combined with DVS event-based candidate ROI selection. It was designed to combine the advantages of both the frame and event outputs of the DAVIS sensor. The tracking framework combines a conventional CNN based tracker with ROIs from a cluster-based DVS tracker. The system is tested on the Ulster dataset to solve the task of tracking the prey in a cluttered background with ego motion. The result shows a 90% tracking accuracy with 20 pixel precision for the Ulster dataset. The tracking cost of 20ms/frame provides a speedup of 70X compared with a full-frame CNN-based tracking.

ACKNOWLEDGMENT

The authors thank Christian Brandli and Ilya Kiselev for discussions.

REFERENCES

[1] C. Brandli, R. Berner, M. Yang, S.-C. Liu, and T. Delbruck, "A 240x180 130 dB 3 us Latency Global Shutter Spatiotemporal Vision Sensor," *IEEE J. Solid-State Circuits*, vol. 49, no. 10, pp. 2333–2341, Oct. 2014.

[2] P. Lichtsteiner, C. Posch, and T. Delbrück, "A 128 x 128 120dB 15us Latency Asynchronous Temporal Contrast Vision Sensor," *IEEE J Solid-State Circuits*, vol. 43, no. 2, pp. 566–576, 2008.

[3] K. Zhang, Q. Liu, Y. Wu, and M.-H. Yang, "Robust Visual Tracking via Convolutional Networks," *ArXiv150104505 Cs*, Jan. 2015.

[4] J. Fan, W. Xu, Y. Wu, and Y. Gong, "Human Tracking Using Convolutional Neural Networks," *IEEE Trans. Neural Netw.*, vol. 21, no. 10, pp. 1610–1623, Oct. 2010.

[5] M. Litzenberger, C. Posch, D. Bauer, A. N. Belbachir, P. Schon, B. Kohn, and H. Garn, "Embedded Vision System for Real-Time Object Tracking using an Asynchronous Transient Vision Sensor," in *Digital Signal Processing Workshop, 12th - Signal Processing Education Workshop, 4th*, 2006, pp. 173–178.

[6] T. Delbruck, "Frame-free dynamic digital vision," in *Proceedings of Intl. Symp. on Secure-Life Electronics*, Tokyo, Japan, 2008, vol. 1, pp. 21–26.

[7] D. Drazen, P. Lichtsteiner, P. Häfliger, T. Delbrück, and A. Jensen, "Toward real-time particle tracking using an event-based dynamic vision sensor," *Exp. Fluids*, vol. 51, no. 5, pp. 1465–1469, Nov. 2011.

[8] J. Conradt, M. Cook, R. Berner, P. Lichtsteiner, R. J. Douglas, and T. Delbruck, "A Pencil Balancing Robot Using a Pair of AER Dynamic Vision Sensors," in *IEEE International Symposium on Circuits and Systems (ISCAS) 2009*, Taipei, 2009, pp. 781–784.

[9] D. R. Valeiras, X. Lagorce, X. Clady, C. Bartolozzi, S.-H. Ieng, and R. Benosman, "An Asynchronous Neuromorphic Event-Driven Visual Part-Based Shape Tracking," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. PP, no. 99, pp. 1–1, 2015.

[10] T. Delbruck, M. Pfeiffer, R. Juston, G. Orchard, E. Muggler, A. Linares-Barranco, and M. W. Tilden, "Human vs. computer slot car racing using an event and frame-based DAVIS vision sensor," in *2015 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2015, pp. 2409–2412.

[11] D. J. Borer, "4D Flow Visualization with Dynamic Vision Sensors," PhD Thesis, ETH Zurich, Zurich, Switzerland, 2014.

[12] Z. Ni, C. Pacoret, R. Benosman, S. Ieng, and S. RéGnier, "Asynchronous Event-Based High Speed Vision for Microparticle Tracking," *J. Microsc.*, vol. 245, no. 3, pp. 236–244, Mar. 2012.

[13] T. Delbruck and M. Lang, "Robotic Goalie with 3ms Reaction Time at 4% CPU Load Using Event-Based Dynamic Vision Sensor," *Front. Neurosci.*, vol. 7, p. 223, Nov. 2013.

[14] Z. Ni, S.-H. Ieng, C. Posch, S. Régnier, and R. Benosman, "Visual Tracking Using Neuromorphic Asynchronous Event-Based Cameras," *Neural Comput.*, vol. 27, no. 4, pp. 925–953, Feb. 2015.

[15] "jAER Open Source Project," *jAER Open Source Project*. [Online]. Available: http://jaerproject.org. [Accessed: 17-Sep-2013].

[16] "rasmusbergpalm/DeepLearnToolbox," *GitHub*. [Online]. Available: https://github.com/rasmusbergpalm/DeepLearnToolbox. [Accessed: 15-Mar-2015].

[17] F. Gustafsson, F. Gunnarsson, N. Bergman, U. Forssell, J. Jansson, R. Karlsson, and P.-J. Nordlund, "Particle filters for positioning, navigation, and tracking," *IEEE Trans. Signal Process.*, vol. 50, no. 2, pp. 425–437, Feb. 2002.

[18] Y. Wu, J. Lim, and M.-H. Yang, "Online Object Tracking: A Benchmark," presented at the Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2013, pp. 2411–2418.