Event-Based Hough Transform in a Spiking Neural Network for Multiple Line Detection and Tracking Using a Dynamic Vision Sensor

Sajjad Seifozzakerini[1]
stusajjad@i2r.a-star.edu.sg
Wei-Yun Yau[1]
wyyau@i2r.a-star.edu.sg
Bo Zhao[1]
zhaob@i2r.a-star.edu.sg
Kezhi Mao[2]
ekzmao@ntu.edu.sg

[1] Institute for Infocomm Research (I2R), Agency for Science, Technology and Research (A*STAR), Singapore

[2] School of Electrical and Electronic Engineering, Nanyang Technological University (NTU), Singapore

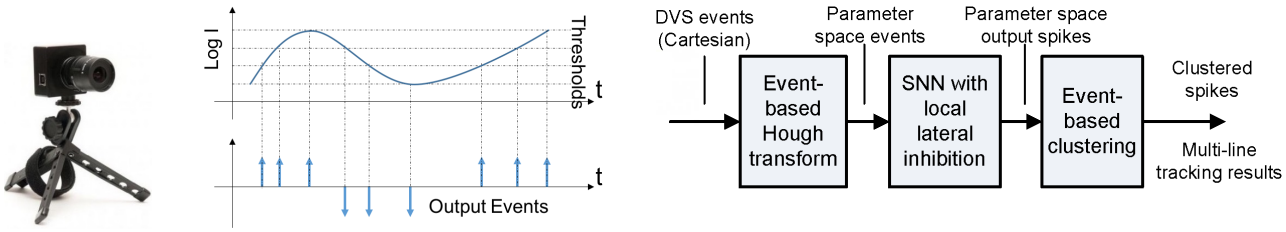
Abstract

Hough Transform has been widely used to detect lines in images captured by conventional cameras. In this paper, we develop an event-based Hough transform and apply it to a new type of camera, namely Dynamic Vision Sensor (DVS). DVS outputs an asynchronous stream of binary events representing illumination change in the scene. We implement the proposed algorithm in a spiking neural network to detect lines on DVS output. Spikes (events) from the DVS sensor are first mapped to Hough transform parameter space and then sent to corresponding spiking neurons for accumulation. A spiking neuron will fire an output spike once it accumulates enough input contributions and then reset itself. The output spikes of the spiking neural network represent the parameters of detected lines. An event-based clustering algorithm is applied on the parameter space spikes to segment multiple lines and track them. In our spiking neural network, a lateral inhibition strategy is applied to suppress noise lines from being detected. This is achieved by resetting a neuron's neighbors in addition to itself once the neuron fires an output spike. The efficacy of the proposed algorithm is shown by extensive experiments on both artificially generated events and various real DVS outputs.

1 Introduction

Dynamic Vision Sensor (DVS) is a relatively new event-based video camera. Comparing to a conventional frame-based camera, DVS offers great advantages in terms of data rate, speed, and dynamic range [11]. Simulating biological retinas, DVS sensors are sensitive to the intensity change rather than the absolute intensity value. Conventional cameras synchronously capture frames (e.g. 30 frames per second) with each frame containing the intensity values

Pages 94.1-94.12
DOI: https://dx.doi.org/10.5244/C.30.94



(a) (b) (c)

Figure 1: (a) Dynamic Vision Sensor and (b) its working principle; (c) Block diagram of the proposed algorithm.

of all the pixels. Pixels with little change are repeatedly reported in different frames, generating a lot of redundant information and seriously limiting the frame rate and temporal accuracy of the camera. In contrast, DVS sensors only report pixels with intensity changes and output them autonomously as an asynchronous stream of binary events. All the pixel circuits of the DVS work in parallel, and whenever there is an enough change in a pixel's intensity, the pixel autonomously reports its position and the polarity of the intensity change as an event. Since redundant background information is discarded at the focal plane, we will have much less data volume as well as a very high temporal resolution. Moreover, due to logarithmic intensity change detection, DVS sensors offer a very high dynamic range, meaning it has no problem in capturing the scenes containing both very dark and very bright areas. These sensors have been used in many applications such as motion estimation [14], tracking [5, 18, 22], object recognition [13, 16, 23], and corner points detection [4]. In this work, we use a DVS to perform line detection based on an event-based Hough transform algorithm. The DVS used in this study has a 128×128 spatial resolution and $1\mu s$ temporal accuracy.

Hough Transform was introduced in 1972 as a feature extraction (especially line detection) method in computer vision [6]. The main idea of this method is first transforming every point from the conventional Cartesian coordinates to the parameter space, in which every point defines a specific shape, and then finding local maximums in the parameter space to obtain the shape parameters through a voting procedure. The dimension of the parameter space depends on the shape that we want to extract and its complexity. A line can be uniquely defined by two parameters and therefore the parameter space for detecting lines is two dimensional. Three parameters (x and y positions of center and radius) can define a circle on a plane, and thus the parameter space for detecting circles is three dimensional. Hough transform can also be used for detecting arbitrary shapes [1]. Hough transform can be accelerated by managing the needed memory size or limiting the search area in the parameter space. The problem dimensionality can be reduced by some tricks [19]. Using line detection as an example, since the line direction is perpendicular to the image gradient vector, the line slope can be obtained immediately according to the gradient vector and a two-dimensional Hough transform is then converted to a simpler one-dimensional problem. Another idea for accelerating the algorithm is using coarse-to-fine searching in the parameter space and is named hierarchical Hough transform [10].

In this paper, we propose an event-based Hough transform algorithm and implement it in a Spiking Neural Network (SNN) to perform line detection on DVS event output. SNN is the third generation of Artificial Neural Network (ANN) models. Compared to conventional ANN, SNN is more biologically plausible [23] since it incorporates spike times into computational models which mimic the information processing in the biological neural system. Recent advances in VLSI technology have solved the spiking neurons' implementation

issues [12, 15] that we faced previously because of their higher computational complexity compared to conventional artificial neurons, and thus have largely boosted the SNN research and development. SNN have been used for many tasks such as learning [17, 20, 21] and classification [3, 8, 23]. Among the various spiking neuron models proposed in the literature [7, 9], the most popular one is Leaky Integrate-and-Fire (LIF) neuron model [2] due to its simplicity. In our work, we use LIF spiking neurons to construct an two-dimensional SNN which represents the parameter space of Hough transform for line detection. As shown in Figure 1, the proposed SNN processes the events from a DVS sensor in an event-driven manner, and generates output spikes which represent the parameters of detected lines. Local lateral inhibition is adopted in our SNN to prevent neurons from firing in the wrong locations and thus suppress noise lines. At the last stage, an event-based clustering procedure is applied on the parameter space spikes to segment lines and track them. The major contribution of this work includes: 1) proposing a fully event-driven SNN-based algorithm for fast line detection, 2) incorporating local lateral inhibition in the SNN for noise line suppression, and 3) applying event-based clustering on SNN output spikes to achieve efficient multiple line tracking.

The rest of the paper is organized as follows. Section 2 describes the DVS sensor, Section 3 summarizes conventional hough transform for line detection. The proposed algorithm is illustrated in Section 4. Section 5 shows the experimental results and Section 6 concludes this paper.

2 Dynamic Vision Sensors

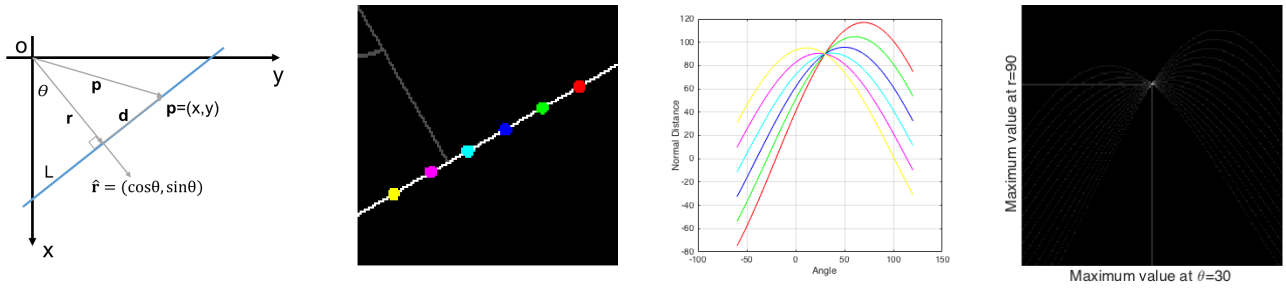
Dynamic Vision Sensors (DVSs) are a new generation of cameras that are sensitive to intensity change, more specifically, to intensity logarithmic change. Let us consider the logarithm of a pixel intensity as shown in Figure 1. Once the logarithmic intensity change is larger than a predefined threshold, a positive or negative event will be generated depending on the direction of the change (dark-to-bright or bright-to-dark). Every event consists of four parameters including the time t, position (x,y) and polarity. All these parameters are integer values except the polarity which is binary (+/-). Figure 1 shows the DVS used in this study which has a 128×128 spatial resolution and $1\mu s$ temporal accuracy. According to the characteristic of a logarithmic function which is more sensitive to small values, DVS is also more sensitive to darker areas. A small change in low intensity (dark pixel) can cause a significant change in the intensity logarithm and generate an event subsequently. As a result, there will be more noise events in darker areas and it is verified by actual experiments as well.

3 Hough transform for line detection

Let us consider the problem of Hough transform for line detection. Every line L can be uniquely defined by two parameters including the normal distance r from the origin and angle θ between the normal vector $\mathbf{r}$ and x-axis (see figure 2). Assuming $\hat{\mathbf{r}} = (\cos\theta, \sin\theta)$ as the unit vector perpendicular to the line L, for every point $\mathbf{p} = (x,y)$ on the line:

$$\hat{\mathbf{r}}.\mathbf{p} = r \rightarrow x\cos\theta + y\sin\theta = r \tag{1}$$

Essentially the parameters chosen for determining lines are r and θ. Equation (1) transforms every point from Cartesian coordinate to parameter space (r,θ). In other words, every



(a) (b) (c) (d)

Figure 2: Hough transform procedure for line detection: (a) A simple line on a plane can be uniquely defined by two parameters r and θ; (b) Six points on a line with parameters $(r,\theta) = (90, 30^\circ)$ in a 128×128 frame; (c) Six sinusoidal curves in parameter space are coincident at $(r,\theta) = (90, 30^\circ)$; (d) Parameter space for 15 sinusoidal curves contains a bright point at $(r,\theta) = (90, 30^\circ)$.

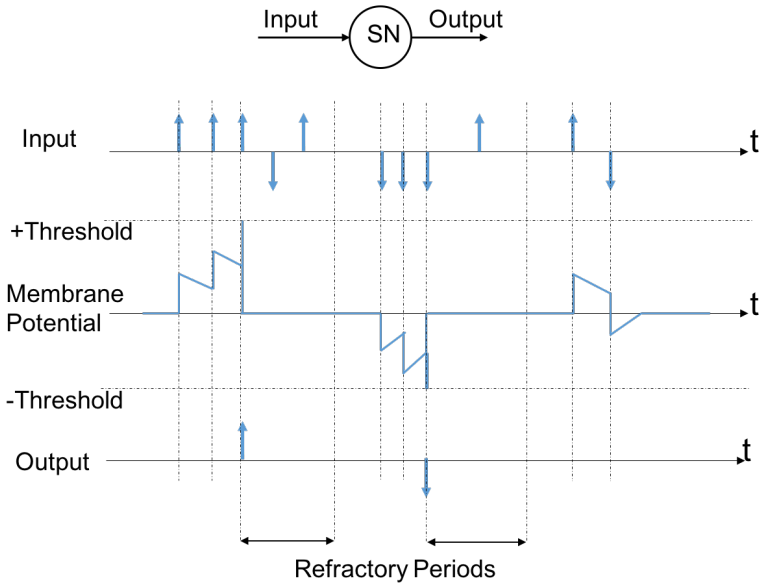


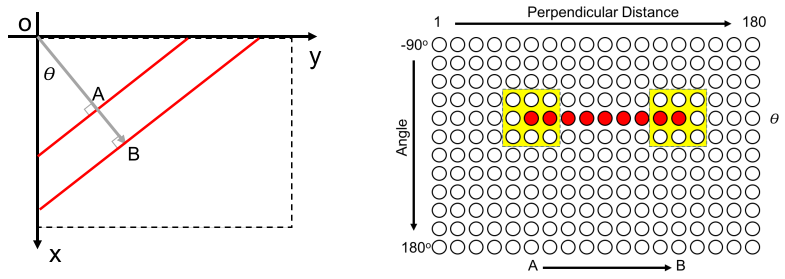
Figure 3: Input and output of the LIF neuron used in this paper.

point in Cartesian space is transformed to a sinusoidal curve in parameter space defined by equation (1).

Referring to Figure 2, let us consider a line with parameters $r = 90$ and $\theta = 30^\circ$ in a frame with 128×128 resolution. There are six points highlighted with six different colors on the line. According to equation (1), every point is transformed to the parameter space as shown in Figure 2. Obviously all six sinusoidal curves are coincident at $(r,\theta) = (90, 30^\circ)$ since their corresponding points in Cartesian space are all on a straight line. If we perform this procedure for more points on the line (e.g. 15) and represent the parameter space as an image, there will be a bright point at $(r,\theta) = (90, 30^\circ)$ as shown in Figure 2. Therefore, lines in a frame can be extracted by finding local maximums in this parameter space.

4 Event-Based Hough Transform in a Spiking Neural Network

This section illustrates the proposed event-based Hough transform algorithm implemented in an SNN for line detection.



(a) (b)

Figure 4: (a) A 128×128 frame and (b) its corresponding parameter space on a 200×300 SNN. θ is limited between $-90°$ and $180°$ while r between 1 and $128\sqrt{2} \approx 180$. Red colored neurons show the firing neurons from left to right during the red line movement from A to B. Yellow window indicates local lateral connection (for inhibition). Each neuron is laterally connected to all neurons that are within a window around that neuron

4.1 Spiking Neuron Model

In this paper, we use LIF spiking neurons to build an SNN that represents the parameter space of Hough transform for line detection. As shown in Figure 3, every Spiking Neuron (SN) has some inputs (one input is used here for simplicity) and an output. The input is a spike train that influences the neuron's Membrane Potential (MP). Each input spike causes an increase (for positive event) or decrease (for negative event) of the MP. Note that MP is always decaying by a fixed rate. Whenever the MP exceeds the + or - threshold, a spike with corresponding polarity is generated in the output. Then MP is reset to zero and the neuron enters a refractory period, during which MP remains zero and input spikes are ignored. In this work, we apply a zero refractory period for simplicity; this can also lead to more output spikes. Note that the firing of other SNs can also force an SN to reset through lateral inhibition in a network (which will be explained in details in Section 4.2).

We update the MPs and neuron states only when an input spike arrives. Let us define s_i as the i^{th} input spike with a time stamp t_i and polarity ($s_i = \pm 1$). Algorithm 1 is performed for each input spike s_i. In this algorithm, v_i is the neuron membrane potential, λ the rate of linear decay and v_{th} the threshold for membrane potential.

Algorithm 1 Updating procedure of a spiking neuron when receiving an input spike ($\lambda = 3mv/ms, v_{th} = 15mv$)

for every input spike s_i at t_i

- $v_i \leftarrow sign(v_{i-1}) min(|v_{i-1}| - \lambda(t_i - t_{i-1}), 0)$
- $v_i \leftarrow v_i + s_i$
- **if** $|v_i| \geq v_{th}$ **then**

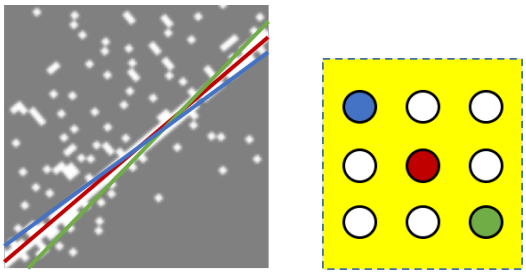
  Generate output spike $f = sign(v_i)$ at t_i

  Reset all laterally connected neurons

  $v_i \leftarrow 0$

  end if

end for



(a) (b)

Figure 5: Local lateral inhibition to suppress noise lines. (a) Without lateral inhibition, SNN detects three lines (red, blue and green). The best fitted one is the red line detected by the red neuron in (b). The red neuron is expected to fire before blue or green ones. With local lateral inhibition, when the red neuron (the correct one) fires, it inhibits all laterally connected neurons and thus blue and green lines can be suppressed.

4.2 The Proposed SNN for Event-Based Hough Transform

Let us consider a line moving from normal distance A to B with a fixed slope as shown in Figure 4. The parameter space is built up by a two dimensional SNN with one dimension for angle θ and the other for normal distance r. To cover all possible lines in a 128×128 frame, θ and r are limited to $(-90^\circ, 180^\circ)$ and $(1, 128\sqrt{2} \approx 180)$. We limit r to just positive values and extend θ instead. A local lateral inhibition strategy is also adopted in our SNN. The output of every neuron is laterally connected to all the neighbouring neurons that are located within a window centred at that neuron. The window size chosen here is $10(degree) \times 5(pixel)$. Whenever a neuron fires, it forces itself as well as all neighbouring neurons within a window to be reset. This window is represented by the yellow rectangles in Figure 4. During the red line movement from normal distance A to B in Figure 4(a), we will have many spikes in the parameter space from left to right highlighted by red color in Figure 4(b).

Local lateral inhibition allows the SNN to suppress noise lines (or redundant lines) from being detected. This is illustrated in Figure 5 which shows the events generated by moving an edge from left to right in front of the DVS sensor. The left side of the edge is totally black while the right side is white. All events, whether positive or negative, are shown by white dots on a gray background. The detected lines using the SNN without local lateral inhibition are superimposed onto the DVS events. The best fitted line is the red line which is detected by the red neuron in Figure 5. However, the blue and green lines also cover many events in the frame and they are detected by the firing of the blue and green neurons in the parameter space, respectively. These two lines can be suppressed by the local lateral inhibition. Each neuron is laterally connected to all its neighbouring neurons that are within a window around that neuron. These connections cause a local competition between the neurons inside the yellow window. Whenever the first neuron (red one in this case) fires, it resets all neighbouring neurons, prevents them from firing, and thus suppresses the noise lines.

The parameter space is built up by $N \times M$ SNs; i.e. N rows for θ quantization and M columns for r quantization. Therefore every neuron located at (j,k) is identified by (θ_j, r_k) values. Algorithm 2 shows the proposed event-based Hough transform and neuron excitation in the parameter space SNN. Assume that an event $e_i = (t_i, x_i, y_i, s_i)$ is received from DVS, meaning that there is an intensity change of pixel (x_i, y_i) at time t_i. The term s_i is either 1 for dark-to-bright change or -1 for bright-to-dark change. For every $\theta_j (1 \leq j \leq N)$, the

Algorithm 2 Event-based Hough transform and neuron excitation in the parameter space SNN

for every received event $e_i = (t_i, x_i, y_i, s_i)$

- Calculate $r_{\theta_j} = \arg\min_{r_k} |r_k - x_i \cos\theta_j - y_i \sin\theta_j|$ for $1 \leq j \leq N$ and $1 \leq k \leq M$
- Excite all neurons (θ_j, r_{θ_j}) at t_i for $1 \leq j \leq N$ with s_i (algorithm 1)
- Reset all neurons in first column $(k = 1)$

end for

corresponding r_{θ_j} is calculated using equation (1) and rounded to the nearest possible $r_k (1 \leq k \leq M)$. Then all neurons $(\theta_j, r_{\theta_j})(1 \leq j \leq N)$ inside the parameter space are excited by s_i based on algorithm 1. We need to ignore all neurons in the first column $(k = 1)$ because they can be wrongly excited by the cases that Hough transform of an event is partly outside of the boundary of the parameter space.

Algorithm 3 Event-based clustering for segmentation and tracking of multiple detected lines

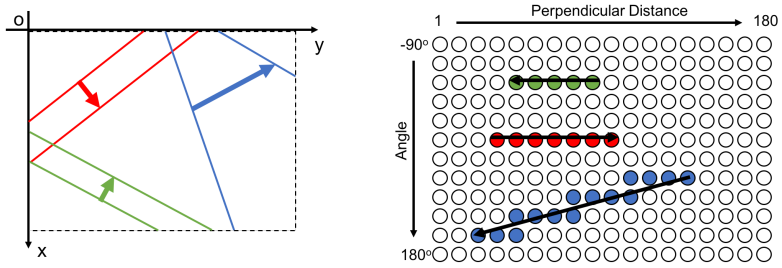
for every received spike $f_i = (t_i, r_i, \theta_i)$ from parameter space

- Delete all clusters $C_l = (T_l, R_l, \Theta_l, Texp_l, N_l)$ that $t_i > Texp_l$
- Find the minimum weighted Euclidean distance D_i between (r_i, θ_i) and all active clusters (R_l, Θ_l). Suppose $C_{l=m}$ is the winner; i.e. $D_i = \min_l ||(r_i, \theta_i) - (R_l, \Theta_l)||_w = ||(r_i, \theta_i) - (R_m, \Theta_m)||_w$
- **if** $D_i \leq D_{threshold}$ **then**
  f_i belongs to cluster $C_{l=m}$
  $(T_m, R_m, \Theta_m) = (1-\alpha)(T_m, R_m, \Theta_m) + \alpha(t_i, r_i, \theta_i) \qquad (\alpha = 0.1)$
  if $N_m \geq (K-1)$ **then**
  $N_m = K$
  $Texp_m = t_i + T_{visible}$
  else
  $N_m = N_m + 1$
  $Texp_m = t_i + T_{hidden}$
  end if
  else
  Generate a new cluster $C_{L+1} = (t_i, r_i, \theta_i, t_i + T_{hidden}, 0)$
  end if

end for

4.3 Segmentation and Tracking

In cases that there are more than one moving line in the frame, we need a segmentation procedure to distinguish between them as shown in Figure 6. Many spikes are generated from the parameter space SNN in different locations. Since every line is moving smoothly in Cartesian space, usually the corresponding spikes in parameter space are "moving" smoothly as well and they produce a cluster. The l^{th} cluster is defined by a 5-elements matrix $C_l =$



(a) (b)

Figure 6: (a) Three moving lines in Cartesian space defined by three colors and (b) their transformations in the parameter space. Spikes are segmented in three different series correspond to three lines.

Table 1: Quantitative analysis of line detection results on artificially generated line events

Line	Input Events #	Output Spikes #	Spike Time (ms) First/Last		θ error (degree) Mean/SD		r error (pixel) Mean/SD	
Cyan	9100	158	2630	4994	0.04	0.18	0.01	0.22
Blue	14116	278	6	4953	0.07	0.21	0.12	0.24
Green	18022	418	20	4993	-0.02	0.23	-0.03	0.21
Red	31654	923	0	4970	0.04	0.31	0.06	0.31

$(T_l, R_l, \Theta_l, Texp_l, N_l)$ in which the first 3 parameters show the time and location of the cluster; i.e. detected line while the last two show the expiry time and visibility situation. We use an event-based clustering method [5, 22] to do the segmentation and tracking of different lines. Every cluster is hidden when it is generated for the first time ($N_l = 0$). With any input spike, N_l increments till it reaches a maximum value K (5 in our case) and remains unchanged afterward. A hidden cluster ($0 \leq N_l < K$) becomes visible ($N_l = K$) if it receives enough support from the network. Every cluster, whether visible or hidden, will be deleted at the expiry time ($Texp_l$) which is set based on the time of the last received spike. The validity period for visible clusters ($40ms$ in our case) is chosen longer than the validity period for hidden clusters ($10ms$ in our case); i.e. $T_{visible} > T_{hidden}$.

For every spike $f_i = (t_i, r_i, \theta_i)$ from the parameter space SNN, we calculate the weighted Euclidean distances from this spike location to all clusters, and find the smallest distance and corresponding cluster. If this distance is less than a threshold, the spike f_i is considered as belonging to that corresponding cluster, and the corresponding cluster parameters $C_m = (T_m, R_m, \Theta_m, Texp_m, N_m)$ are then updated; otherwise, a new cluster is generated by this spike as seen in algorithm 3.

5 Experimental Results

5.1 On Artificially Generated Line Events

We first evaluated our algorithm on some artificially generated line events. As shown in figure 7, we generated artificial events by moving 4 different lines in a single scene. The first row shows the moving traces of these lines and the line detection results at two different time instances. Detected lines are shown in colors, and the input events (within 200 ms around the time instance) are shown in grey. The second row shows the detection results (in colors) superimposed on ground truth (dashed lines) during the whole time course (0-5 s) for angle θ and normal distance r, respectively. We can see that the propose algorithm can accurately

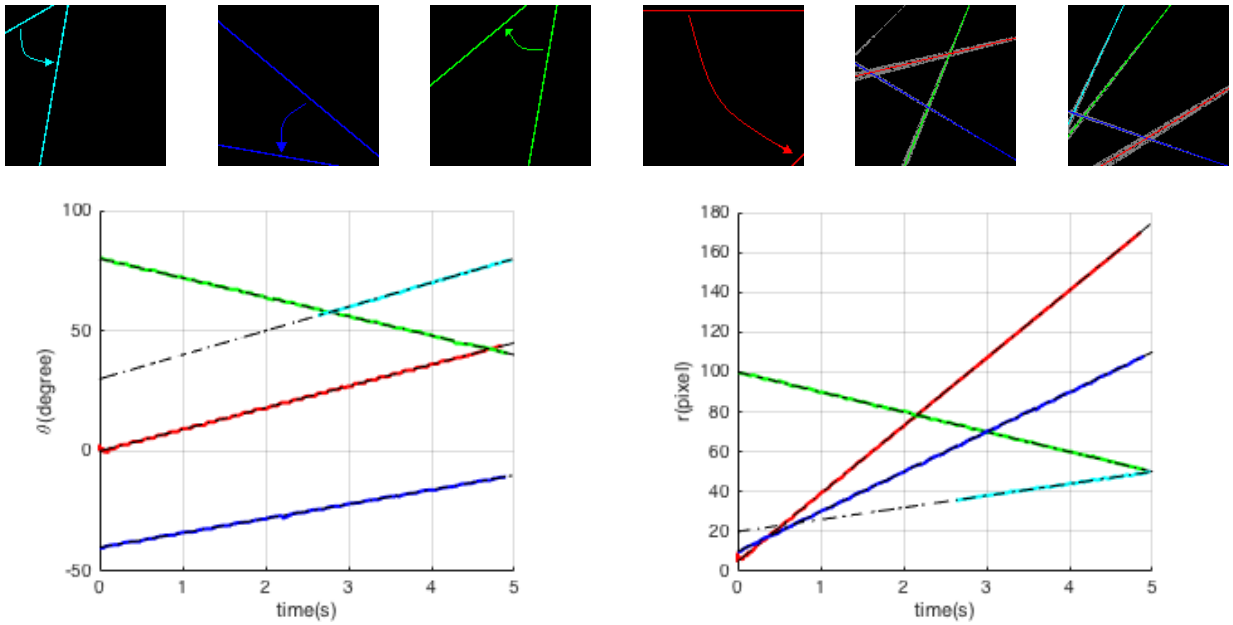


Figure 7: Line detection results on artificially generated line events. 1st row: Traces of 4 different lines and results (in colors) overlaid on input events (grey pixels) at certain time instances (1.5s and 3.5s). 2nd row: Results (in colors) superimposed on ground truth (dashed lines) during the whole time course.

detect and track the four moving lines. Table 1 reports the statistics of the line detection results. As we can see, the errors are very trivial, which quantitatively demonstrates the accuracy of the proposed algorithm. Note that the first line (in color cyan) is not detected until t = 2630 ms. This is because the initial length of this line is quite short, leading to a small number of input events and thus few output spikes in the corresponding cluster. A cluster with few spikes at the beginning will remain hidden in order to prevent trivial lines from being detected/reported.

5.2 On Real DVS Events

We also evaluated the proposed algorithm on various real DVS event streams, and compared the results with those of conventional frame-based hough transform. The results are illustrated in Figure 8. The first row shows the images captured by a conventional camera; they depict the various scenes that the DVS sensor was recording. The second row illustrates the propose algorithm's line detection results (yellow) superimposed onto DVS events (gray). The third row shows the results of conventional frame-based hough transform algorithm applied on frames reconstructed from DVS events. For each scenario, we let the frame-based hough transform detect the same number of lines as our event-based algorithm does. As we can see, the proposed event-based algorithm provides better detection results, especially in scenes 3, 5, and 6. In DVS output, there is usually a burst of events on edge pixels which can be effectively utilized by our proposed framework for better results. In addition, tracking of the detected multiple lines is very easy using our algorithm, whereas it is quite hard to achieve using conventional frame-based algorithm.

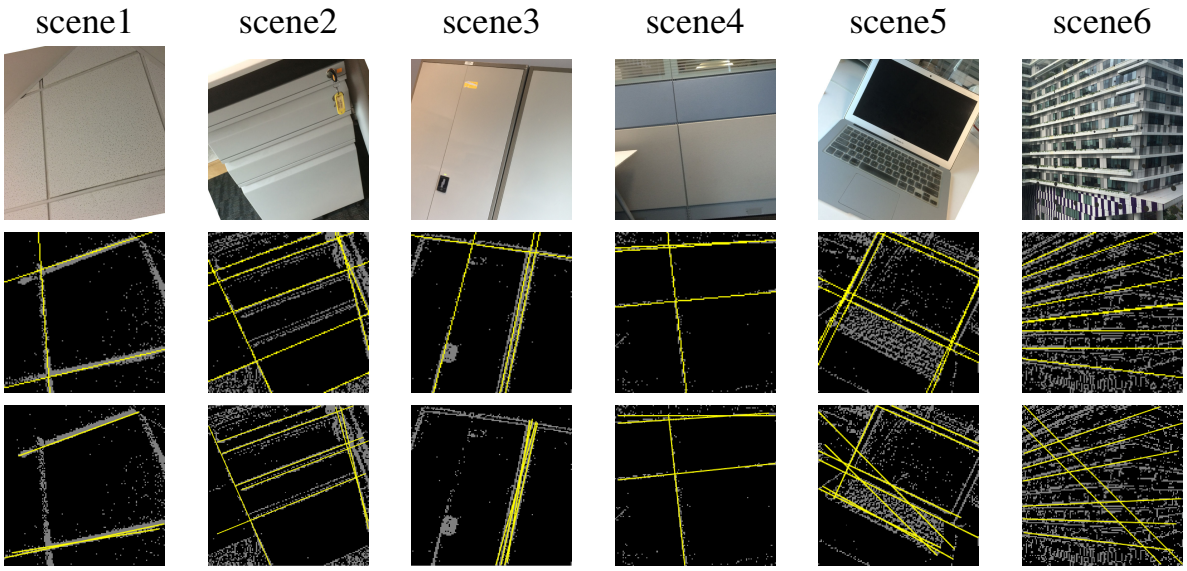


Figure 8: Line detection results on various scenarios; 1st row: images captured by a conventional camera, depicting various scenes that the DVS sensor was recording; 2nd row: The proposed event-based algorithm's line detection results (yellow) superimposed onto DVS events (grey); 3rd row: Conventional frame-based hough transform's results using MATLAB standard functions for line detection with the same number of the lines.

6 Conclusion

An event-based Hough transform approach based on a spiking neural network have been proposed to perform multiple line detection and tracking on Dynamic Vision Sensors. Extensive experiments on both artificial and real DVS event streams have demonstrated its efficacy. SNN with local lateral inhibition is efficient in detecting correct lines as well as suppressing incorrect ones, while event-based clustering on the SNN output spikes allows efficient tracking of the detected multiple lines.

References

[1] D.H. Ballard. Generalizing the Hough transform to detect arbitrary shapes. *Pattern Recognition*, 13(2):111–122, jan 1981. ISSN 00313203. doi: 10.1016/0031-3203(81)90009-1.

[2] A N Burkitt. A review of the integrate-and-fire neuron model: I. Homogeneous synaptic input. *Biological cybernetics*, 95(1):1–19, jul 2006. ISSN 0340-1200. doi: 10.1007/s00422-006-0068-6.

[3] Shoushun Chen, Polina Akselrod, Bo Zhao, Jose Antonio Perez Carrasco, Bernabe Linares-Barranco, and Eugenio Culurciello. Efficient feedforward categorization of objects and human postures with address-event image sensors. *IEEE transactions on pattern analysis and machine intelligence*, 34(2):302–14, feb 2012. ISSN 1939-3539. doi: 10.1109/TPAMI.2011.120.

[4] Xavier Clady, Sio-Hoi Ieng, and Ryad Benosman. Asynchronous event-based corner detection and matching. *Neural Networks*, 66:91–106, 2015. ISSN 08936080. doi: 10.1016/j.neunet.2015.02.013.

[5] T Delbruck and P Lichtsteiner. Fast sensory motor control based on event-based hybrid neuromorphic-procedural system. *Circuits and Systems, 2007. ISCAS ...*, (80 cm): 845–848, 2007.

[6] Richard O. Duda and Peter E. Hart. Use of the Hough transformation to detect lines and curves in pictures. *Communications of the ACM*, 15(1):11–15, jan 1972. ISSN 00010782. doi: 10.1145/361237.361242.

[7] Robert Gütig and Haim Sompolinsky. The tempotron: a neuron that learns spike timing-based decisions. *Nature neuroscience*, 9(3):420–8, mar 2006. ISSN 1097-6256. doi: 10.1038/nn1643.

[8] Jun Hu, Huajin Tang, K C Tan, Haizhou Li, and Luping Shi. A spike-timing-based integrated model for pattern recognition. *Neural computation*, 25(2):450–72, feb 2013. ISSN 1530-888X. doi: 10.1162/NECO_a_00395.

[9] E M Izhikevich. Simple model of spiking neurons. *IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council*, 14(6):1569–72, jan 2003. ISSN 1045-9227. doi: 10.1109/TNN.2003.820440.

[10] Hungwen Li, Mark A Lavin, and Ronald J Le Master. Fast Hough transform: A hierarchical approach. *Computer Vision, Graphics, and Image Processing*, 36(2-3):139–161, nov 1986. ISSN 0734189X. doi: 10.1016/0734-189X(86)90073-3.

[11] Patrick Lichtsteiner, Christoph Posch, and Tobi Delbruck. A 128 X 128 120 dB 15 us latency asynchronous temporal contrast vision sensor. *IEEE Journal of Solid-State Circuits*, 43(2):566–576, feb 2008. ISSN 00189200. doi: 10.1109/JSSC.2007.914337.

[12] Paul A Merolla et al. Artificial brains. A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science (New York, N.Y.)*, 345(6197): 668–73, aug 2014. ISSN 1095-9203. doi: 10.1126/science.1254642.

[13] Emre Neftci, Srinjoy Das, Bruno Pedroni, Kenneth Kreutz-Delgado, and Gert Cauwenberghs. Event-driven contrastive divergence for spiking neuromorphic systems. *Frontiers in neuroscience*, 7:272, jan 2013. ISSN 1662-4548. doi: 10.3389/fnins.2013.00272. URL /pmc/articles/PMC3922083/?report=abstract.

[14] Garrick Orchard and Ralph Etienne-Cummings. Bioinspired Visual Motion Estimation. *Proceedings of the IEEE*, 102(10):1520–1536, oct 2014. ISSN 0018-9219. doi: 10.1109/JPROC.2014.2346763.

[15] Eustace Painkras et al. SpiNNaker: A 1-W 18-Core System-on-Chip for Massively-Parallel Neural Network Simulation. *IEEE Journal of Solid-State Circuits*, 48(8):1943–1953, aug 2013. ISSN 0018-9200. doi: 10.1109/JSSC.2013.2259038.

[16] José Antonio Pérez-Carrasco, Bo Zhao, Carmen Serrano, Begoña Acha, Teresa Serrano-Gotarredona, Shouchun Chen, and Bernabé Linares-Barranco. Mapping from frame-driven to frame-free event-driven vision systems by low-rate rate coding and coincidence processing–application to feedforward ConvNets. *IEEE transactions on pattern analysis and machine intelligence*, 35(11):2706–19, nov 2013. ISSN 1939-3539. doi: 10.1109/TPAMI.2013.71.

[17] Filip Ponulak and Andrzej Kasiński. Supervised learning in spiking neural networks with ReSuMe: sequence learning, classification, and spike shifting. *Neural computation*, 22(2):467–510, feb 2010. ISSN 1530-888X. doi: 10.1162/neco.2009.11-08-901.

[18] David Reverter Valeiras, Xavier Lagorce, Xavier Clady, Chiara Bartolozzi, Sio-Hoi Ieng, and Ryad Benosman. An Asynchronous Neuromorphic Event-Driven Visual Part-Based Shape Tracking. *IEEE transactions on neural networks and learning systems*, mar 2015. ISSN 2162-2388. doi: 10.1109/TNNLS.2015.2401834.

[19] Raymond K.K. Yip, Peter K.S. Tam, and Dennis N.K. Leung. Modification of hough transform for circles and ellipses detection using a 2-dimensional array. *Pattern Recognition*, 25(9):1007–1022, sep 1992. ISSN 00313203. doi: 10.1016/0031-3203(92)90064-P.

[20] Qiang Yu, Huajin Tang, Kay Chen Tan, and Haizhou Li. Precise-spike-driven synaptic plasticity: learning hetero-association of spatiotemporal spike patterns. *PloS one*, 8 (11):e78318, jan 2013. ISSN 1932-6203. doi: 10.1371/journal.pone.0078318.

[21] Qiang Yu, Huajin Tang, Kay Chen Tan, and Haizhou Li. Rapid feedforward computation by temporal encoding and learning with spiking neurons. *IEEE transactions on neural networks and learning systems*, 24(10):1539–52, oct 2013. ISSN 2162-2388. doi: 10.1109/TNNLS.2013.2245677.

[22] Bo Zhao, Xiangyu Zhang, Shoushun Chen, Kay-Soon Low, and Hualiang Zhuang. A 64 X 64 CMOS Image Sensor With On-Chip Moving Object Detection and Localization. *IEEE Transactions on Circuits and Systems for Video Technology*, 22(4):581–588, apr 2012. ISSN 1051-8215. doi: 10.1109/TCSVT.2011.2170119.

[23] Bo Zhao, Ruoxi Ding, Shoushun Chen, Bernabe Linares-Barranco, and Huajin Tang. Feedforward Categorization on AER Motion Events Using Cortex-Like Features in a Spiking Neural Network. *IEEE Trans. on Neural Networks and Learning Systems*, 26 (9):1963–1978, oct 2015. ISSN 2162-2388. doi: 10.1109/TNNLS.2014.2362542.