Effect of Inhibitory Window on Event-Based Hough Transform for Multiple Lines Detection

Sajjad Seifozzakerini
School of Electrical and Electronic Engineering (EEE), Nanyang Technological University (NTU)
50 Nanyang Ave, S639798
(+65) 84209148
sajjad001@ntu.edu.sg

Wei-Yun Yau
Institute for Infocomm Research (I2R), Agency for Science, Technology and Research (ASTAR)
1 Fusionopolis Way, S138632
(+65) 64082661
wyyau@i2r.a-star.edu.sg

Kezhi Mao
School of Electrical and Electronic Engineering (EEE), Nanyang Technological University (NTU)
50 Nanyang Ave, S639798
(+65) 67904284
ekzmao@ntu.edu.sg

ABSTRACT
Hough Transform is a well-known method in the field of computer vision for detecting simple shapes in a photo. This technique is mostly applied to conventional cameras. Recently a new type of imaging devices called Dynamic Vision Sensors (DVSs) have been introduced which only report pixels with intensity change rather than all pixels' intensity values. The current study is an improvement to a research done previously by the authors in which Hough Transform was employed in Spiking Neural Network (SNN) for multiple lines detection and tracking. In that study, the events received from DVS were transformed from Cartesian space to the parameter space which was implemented in a spiking neural network. In that study, position of the firing neurons shows the detected lines' properties. Moreover, lateral inhibitory connections in a rectangular window in the parameter space were used for suppressing neighboring lines. Finally, an event-based clustering algorithm was applied subsequently in the parameter space for tracking detected lines in the video. As an improvement to the work, the current paper deals with detecting small lines at the frame corners which was not considered in the previous study. In addition, the inhibitory window shape is optimized to suppress the lines which are close together in Cartesian space and are not necessarily close together in parameter space assumed in the previous study. The effectiveness of these improvements is tested and verified by experimental results.

CCS Concepts
• Computing methodologies → Artificial intelligence → Computer vision → Computer vision problems → Shape inference
• Applied computing → Enterprise computing → Event-driven architecture

Keywords
Hough Transform; Event-Based Video; DVS; Parameter Space; Line Detection; Tracking; Spiking Neural Network; Lateral Connection; Inhibitory Window.

1. INTRODUCTION
Dynamic Vision Sensor (DVS) is a new category of imaging devices which has a different structure and a new type of output data compared to conventional cameras [Serrano-Gotarredona and Linares-Barranco 2013; Lichtsteiner et al. 2006; Lichtsteiner et al. 2008]. Output of the DVS camera is termed ‘event-based video’ in the literature. Every pixel in a DVS is an asynchronous recorder sensitive to the intensity variation rather than the intensity value. Any pixel independently reports an ‘event’ if a measurable change is detected in the logarithm of the intensity. The event is a 4-element vector including the instance of the change (t), the position of the pixel (x,y) and a polarity (p) which shows the direction of the change. The main advantage of DVS is the high temporal resolution of 1μs that allows the user to track objects moving at a high speed. Since DVS captures only the variations, there is less redundant data in the output, thus needs less processing power. Moreover, all pixels in DVS operate independently and, as a result able to sense extremely dark or bright points.

Dealing with event-based videos, the conventional algorithms in the field of the computer vision should be modified or new procedures to be developed. As such algorithm modifications and developments have been considered for DVS including object tracking [Valeiras et al. 2015; Piatkowska et al. 2012; Lagorce et al. 2015; Ni et al. 2012] or recognition [Neftci et al. 2013; Pérez-Carrasco et al. 2013; Zhao et al. 2015], motion analysis [Benosman et al. 2014; Brosch et al. 2015; Seifozzakerini et al. 2017; Kohn et al. 2012; Barranco et al. 2014; Barranco et al. 2015], robotics [Delbruck and Lang 2013; Brandli et al. 2013; Clady et al. 2014], etc. Moreover, the conventional Hough Transform for multiple lines detection has been modified and implemented in a Spiking Neural Network (SNN) for event-based videos [Seifozzakerini et al. 2016].

Hough Transform is a well-known method in the field of computer vision for detecting multiple shapes [Illingworth and Kittler 1988]. The main idea is to transform all points from the normal Cartesian space to a new space named ‘parameter space’. The parameter space is defined for detecting a specific shape with different parameters e.g. a circle with different center points and radiiuses. The transformation is such that all points on a specific shape are moved to the same point in the parameter space. Coordinates of this point determine the characteristics of the shape. As a result, the problem is just finding the points with maximum votes in the parameter space. Number of parameters depend on the shape complexity e.g. two parameters for lines, three parameters for circles, four parameters for ellipses, etc.

DVS usually reports the points on the shapes’ boundary in Cartesian space. These points are utilized to find straight lines in the video. Any line is uniquely defined by two parameters in polar coordinate including a normal distance r and angle θ from the origin. Every event (x,y) is converted to a sinusoidal shape in the parameter space as follows:

\[ r = \sqrt{x^2 + y^2}, \quad \theta = \arctan\left(\frac{y}{x}\right) \]
The parameter space can be implemented in a spiking neural network. We use Leaky Integrate-and-Fire (LIF) spiking neurons in the network. As illustrated in Figure 1, the LIF neuron has an internal membrane potential which decays continuously unless it receives an input spike. If the membrane potential exceeds a threshold, the neuron will fire and reset the neuron itself and all laterally connected neurons in a window called ‘inhibitory window’. This window is necessary for suppressing redundant lines detected around the actual line.

\[ x \cos(\theta) + y \sin(\theta) = r \]

Figure 1. The detailed behavior of a LIF spiking neuron (SN)

When an event is received, the neurons belonging to the corresponding sinusoidal curve in the parameter space are excited. Those neurons which receive enough excitation in the network will fire and reset all neurons inside the inhibitory window. The location of firing neurons reports the normal distance \( r \) and angle \( \theta \) of the detected line from the origin. Then an event-based clustering algorithm is applied in the parameter space to segment and track the multiple detected lines in the video stream.

In the previous work [Seifozzakerini et al. 2016], a rectangular inhibitory window is utilized and all spiking neurons in the network have the same characteristics including the threshold, decaying rate, etc. In this paper, we want to investigate the effect of the inhibitory window shape on the result and introduce an optimal window according to the area between two lines. Moreover, the threshold values of the membrane potential are regulated for detecting the lines in the frame corners where they have different lengths. Experiments on both synthetic and real data verify the approach proposed in this paper.

2. CARTESIAN vs PARAMETER SPACE

2.1 Transformation

Any line \( L \) in Cartesian space can be uniquely defined by two parameters \( (\theta, r) \) in parameter space as shown in Figure 2. Every point \( p = (x, y) \) belongs to line \( L \) is transformed to a sinusoidal curve \( P \) in parameter space (Equation 1) passing through the point \( L = (\theta, r) \).

\[ f, p = \begin{bmatrix} \cos\theta \\ \sin\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = r - x \cos\theta + y \sin\theta = r \quad \text{Equation 1} \]

This is a special case of Hough Transform used for line recognition in a photo which is applied after the edge extraction of objects (This stage is not necessary in event-based videos).

2.2 Parameter Space

To cover all possible lines within a 128×128 frame, \( \theta \) is limited between \( \frac{- \pi}{2} \) and \( \pi \) while \( r \) between \( 0 \) and \( 128\sqrt{2} \) in the parameter space. However, the possible area is not a simple rectangle according to the transformation Equation 1. Depending on \( r \) and \( \theta \) values, the corresponding lines in the Cartesian space have variable length and different situation inside the video frame. Figure 3 shows different line positions in the video frame and their corresponding areas in the parameter space. This area covers about 60% of the whole rectangle based on the simulation. As a result, the number of spiking neurons can be reduced by 40% in hardware implementation.

The color intensity in the parameter space represents the corresponding line length within the video frame. Smaller lines generate less events in DVS and the corresponding spiking neuron in parameter space receives less excitations subsequently and therefore is not likely to fire. To solve this issue, the internal threshold for neurons’ membrane potential should be proportional to the line length or the colors intensity in the parameter space. However, these values should be larger than a minimum threshold not to detect extremely short lines at the frame corners.

2.3 Inhibitory Window

The issue with the small lines was discussed in the previous section. Another issue stems from the fact that if two lines are close together in the parameter space, they are not necessarily in a close proximity in the Cartesian space. This causes some of the lines not to be extracted which are close together in the parameter space while they are not in a close proximity in Cartesian space.

The events scattering around the actual line in the video frame causes many neurons to fire unexpectedly around the correct neuron in the parameter space. The inhibitory window is proposed as a solution to force wrong neurons to be reset and not to fire as illustrated in Figure 4. Every neuron’s output is laterally connected to inhibitory inputs of all neurons inside a rectangular window centered at the firing neuron. The inhibitory window shape can be optimized to suppress the lines which are close
together in the Cartesian space and not necessarily in parameter space.

Figure 4. Without lateral connections, three lines are detected. The best fitted one, the red line, is expected to be detected before the others. When the red neuron fires, all neurons inside the yellow inhibitory window are reset. Therefore, the redundant lines are suppressed.

To make it clear, Figure 5 is considered. This figure shows two sets of couple lines. First couple lines are \((\theta_1, r)\) and \((\theta_1 + \Delta \theta, r)\) and second couple lines are \((\theta_2, r)\) and \((\theta_2 + \Delta \theta, r)\) at the left and right side of the figure respectively. The distance between the corresponding points of every couple lines in parameter space are the same despite their distances in Cartesian space which are clearly different. This means that \(\Delta \theta\) is the same in Cartesian space (Figure 5) for two lines couples and thus the distance between the corresponding points of the couple lines is the same in parameter space.

Figure 5. Left figure shows 2 lines which are near in both Cartesian space and parameter space. Right figure shows 2 lines which are near in parameter space, not in Cartesian space.

2.3.1 Distance Metric in Cartesian Space

To determine which neurons to be reset, a distance metric \(D\) is defined. This metric is consistent with human perception of the distance between two lines. It is the average distance obtained by dividing the area between two lines by the length of the lines. Assuming the lines are close together, two situations can be considered.

- If the intersection point is inside the frame:
  \[
  D = \frac{S}{d} = \frac{S_1 + S_2}{d_1 + d_2} = \frac{1}{2} \pi \Delta \theta \frac{d_1^2 + d_2^2}{d_1 + d_2} \tag{2}
  \]
- If the intersection point is outside the frame:
  \[
  D = \frac{S}{d} = \frac{S_2 - S_1}{d_2 + d_1} = \frac{1}{2} \pi \Delta \theta \frac{d_2^2 - d_1^2}{d_2 + d_1} \tag{3}
  \]

\[D = \frac{1}{2} \pi \Delta \theta (d_1 + d_2)\]  

The intersection point of two lines \((\theta, r)\) and \((\theta + \Delta \theta, r + \Delta r)\) can be calculated as follows.

\[
\begin{align*}
    x \cos \theta + y \sin \theta &= r \\
    x \cos (\theta + \Delta \theta) + y \sin (\theta + \Delta \theta) &= r + \Delta r
\end{align*}
\]

\[
\begin{bmatrix}
    \cos (\theta) & \sin (\theta) \\ \\
    \cos (\theta + \Delta \theta) & \sin (\theta + \Delta \theta)
\end{bmatrix}
\begin{bmatrix}
    x \\
    y
\end{bmatrix}
= \begin{bmatrix}
    r \\
    r + \Delta r
\end{bmatrix}
\]

\[
\begin{bmatrix}
    \frac{x}{y} \\
    \frac{\sin (\theta + \Delta \theta)}{\cos (\theta + \Delta \theta)}
\end{bmatrix}
= \begin{bmatrix}
    1 \\
    \frac{\sin (\theta + \Delta \theta)}{\cos (\theta + \Delta \theta)}
\end{bmatrix}
\]

Assuming \(\Delta \theta\) is very small:

\[
\Delta \theta \ll 1 \rightarrow \Delta \theta^k \approx 0 \text{ for } k > 1
\]

\[
\begin{bmatrix}
    \frac{x}{y} \\
    \frac{\sin (\theta)}{\cos (\theta)}
\end{bmatrix}
= \begin{bmatrix}
    \frac{r}{r + \Delta r} \\
    \frac{-\Delta \theta \sin (\theta) - \Delta r \sin (\theta)}{r \Delta \theta \sin (\theta) + \Delta r \cos (\theta)}
\end{bmatrix}
\]

\[
\begin{align*}
    x &= r \cos (\theta) - \frac{\Delta r}{\Delta \theta} \sin (\theta) \\
    y &= r \sin (\theta) + \frac{\Delta r}{\Delta \theta} \cos (\theta)
\end{align*}
\tag{4}
\]

2.3.2 Lateral Inhibitory Connections

The inhibitory connections are set just one time at the beginning of the simulation. The spiking neural network consists of \(N \times M\) neurons as shown in Figure 6 (\(N\) rows for \(\theta\) quantization while \(M\) columns for \(r\) quantization). Any neuron \(s_{ij}\) represents a line \((\theta_i, r_j)\) in the parameter space. Steps of setting the inhibitory connections of \(s_{ij}\) are as follows:

- For any spiking neuron \(s_{ij}\) (red neuron), find all neurons \(s_{kl}\) (green neurons) inside a window centered at \(s_{ij}\) (yellow window).
- Considering any green neuron \(s_{kl}\) and the red neuron \(s_{ij}\), find the intersection point of lines \((\theta_k, r_i)\) and \((\theta_l, r_j)\) using Equation 4.
- Depending on the position of line \((\theta, r)\) in Figure 2, find two points of the frame edge that the line is passing through (Figure 7). Calculate the distances \(d_1\) and \(d_2\) from these points (Figure 5) to the intersection point obtained in the previous step.
- Depending on the position of the intersection point, whether it is inside or outside the frame, obtain the distance metric, \(D\) based on Equation 2 or Equation 3.
• If $D < D_{\text{threshold}}$, set an inhibitory connection between $s_{kl}$ and $s_{ij}$.

\[ \frac{r}{\sin(\theta)} = \frac{w}{\cos(\theta)} \]

\[ r - W \sin(\theta) \cos(\theta) \]

Figure 7. Four possible points of the frame edges that an arbitrary line $(\theta, r)$ can pass through. Any line passes through only two points among four possible points.

3. EXPERIMENTS

3.1 Synthetic Data
The algorithm is applied on the same artificially generated data in the literature [Seifozzakerini et al. 2016]. As shown in Figure 8, this synthetic video includes four different lines overlaid in a single frame travelling from their initial states to the final states. The events are randomly generated on these lines during the whole video time span which is 5s.

Figure 8. The synthetic data includes four different lines overlaid in a single scene. The lines are simultaneously travelling from the initial state to the final state during the video stream of time span of 5s.

Figure 9. The perpendicular distance and the angle of detected lines during the whole video time span. The results (in colors) are well matched with the ground truth (dashed lines).

The parameter space is built up by $200 \times 300$ spiking neurons with the decaying rate of $3 \text{ mV/ms}$. As explained in section 2.2, the thresholds of the membrane potentials are proportional to the line length ranging from $6 \text{ mV}$ to $18 \text{ mV}$. The inhibitory window is a simple rectangle of $10^5 \times 5 \text{ (pixels)}$.

Figure 9 shows the results (in colored lines) of the normal distance and angle of the detected lines superimposed on the ground truth (in dashed lines). Table 1 reports some quantitative results of the experiment. The trivial errors in this table show the accuracy of the proposed algorithm. The most significant outcome of this experiment is that small lines are detected using the proposed algorithm. This is shown when the cyan line is detected from almost beginning of the video whilst in the previous study small lines could not be detected i.e. cyan line was not detected until $t = 2630 \text{ ms}$ in previous study.

3.2 Real Data
The effect of inhibitory window optimization was evaluated on real captured data using DVS. A point and two lines with the angle of five degree were printed on a white paper as shown in Figure 10. The page was pinned at the printed point on a wall and in front of a DVS. DVS position was adjusted to capture the pinned point at the upper left corner of the frame. Then the paper was rotated clockwise around the pin while DVS was capturing the video. All implementation details were the same with the previous experiment except the lateral inhibitory connection which were set by $D_{\text{threshold}} = 5 \text{ pixels}$ according to section 2.3.2.

Figure 10. A point and two lines with the angle of five degree were printed on a white paper. The point was pinned on a wall and DVS captured the clockwise rotation of the paper around the point. The algorithm results are shown for two instances of the video.

The results are shown for two instances of the video. In early instances of rotation, the algorithm detected only one line in the video. After a while and by increasing the area between two lines, both lines were detected as illustrated in Figure 10. The ability of detecting lines which are in a close proximity in parameter space has been significantly improved in the current algorithm compared to the previously proposed algorithm. Previous algorithm [Seifozzakerini et al. 2016] detects only one line from the beginning to the end of this video stream.

<table>
<thead>
<tr>
<th>Line</th>
<th>Input Events #</th>
<th>Output Spikes #</th>
<th>Spike Time (ms)</th>
<th>$\theta$ error (degree) Mean/SD</th>
<th>$r$ error (pixel) Mean/SD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cyan</td>
<td>9100</td>
<td>211</td>
<td>23</td>
<td>4995</td>
<td>0.04/0.25</td>
</tr>
<tr>
<td>Blue</td>
<td>14116</td>
<td>263</td>
<td>14</td>
<td>4974</td>
<td>0.06/0.19</td>
</tr>
<tr>
<td>Green</td>
<td>18022</td>
<td>412</td>
<td>21</td>
<td>4994</td>
<td>-0.02/0.23</td>
</tr>
<tr>
<td>Red</td>
<td>31654</td>
<td>1037</td>
<td>4</td>
<td>4995</td>
<td>0.05/0.32</td>
</tr>
</tbody>
</table>

Table 1. Quantitative analysis of line detection results on the synthetic video of four moving lines.
4. CONCLUSION
In this study, we improve our latest algorithm for detecting and tracking multiple lines in event-based videos. By applying these improvements, we can detect very small lines in frame corners effectively. Moreover, we can suppress the lines which are visually close together as well as distinguish those lines not visually close together while they were not detected previously because of their close proximity in the parameter space. In future, the algorithm can be revised again to detect the linear edges with different contrasts or the partial lines in the frame.

5. ACKNOWLEDGMENTS
The authors would like to thank Dr. Ali Rajabipoor for his suggestions and editing of this manuscript.

6. REFERENCES