Live Demonstration: Incremental Motion Estimation for Event-based Cameras by Dispersion Minimisation

Urbano Miguel Nunes and Yiannis Demiris
Personal Robotics Lab, Imperial College London, UK
{um.nunes, y.demiris}@imperial.ac.uk

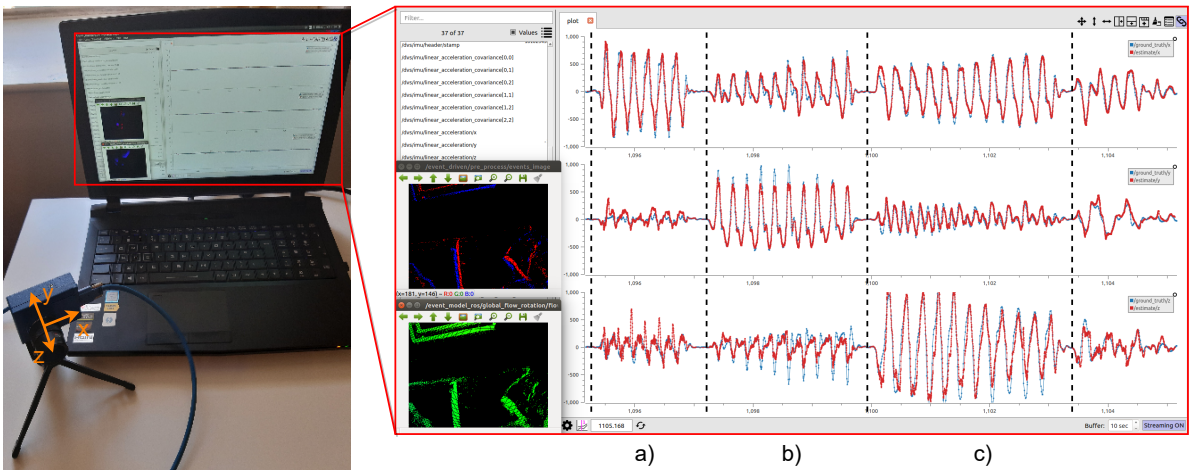

Figure 1: Live demonstration setup. (Left) The setup consists of a DAVIS346B event camera connected to a standard consumer laptop and undergoes some motion. (Right) The motion estimates are plotted in red and, for rotation-like motions, the angular velocities provided by the camera IMU are also plotted in blue. This plot exemplifies an event camera undergoing large rotational motions (up to $\sim$ 1000 deg/s) around the (a) x-axis, (b) y-axis and (c) z-axis. Overall, the incremental motion estimation method follows the IMU measurements. Optionally, the resultant global optical flow can also be shown, as well as the corresponding generated events by accumulating them onto the image plane (bottom left corner).

1. Introduction

Several works have been proposed to use event cameras to estimate motion from vision. Typically, this problem is addressed by processing batches of events [3, 6, 7] since each event does not contain much information by itself. The motion estimates however are obtained with high latency, and the methods can not run in real-time for most scenarios. Some proposed methods can also run in real-time and process one event at a time [1, 4]. However, these methods are typically built for specific motion models [1] or require extra estimations to work (*e.g.* gradients [4]).

The purpose of this demo is to demonstrate event-based vision motion estimation for general motion models in real-time by processing one event at a time, using an incremental version of the method proposed in [6].

2. Brief Description

Here, we briefly describe the event-based incremental motion estimation approach adapted from [6]. For ease of exposition, we will consider minimising the *Potential* energy

$$P(\mathbf{y};\boldsymbol{\theta}) = -\sum_{i,j}^{N_e} \mathcal{K}_{\boldsymbol{\Sigma}}(\mathbf{y}_i,\mathbf{y}_j;\boldsymbol{\theta}) = -\sum_{i,j}^{N_e} w_{i,j}, \qquad (1)$$

where $\boldsymbol{\theta}$ are the motion parameters, $\mathcal{K}_{\boldsymbol{\Sigma}}$ is a Gaussian kernel parameterised by the covariance matrix $\boldsymbol{\Sigma}$, N_e is the number of events considered and $\mathbf{y_i}$ represents the transformed coordinates of the i-th event, according to $\mathbf{y}_i = \mathcal{M}(e_i;\boldsymbol{\theta})$. In this work, an event $e = (\mathbf{x}, t, p)$ is characterised by its image coordinates $\mathbf{x} = (x, y)^\mathsf{T}$, the time-stamp it occurred t and its polarity $p \in \{-1, +1\}$, *i.e.* brightness change.

The *Potential* energy (1) can be (locally) minimised when its derivative w.r.t. the model parameters $\boldsymbol{\theta}$ is zero:

$$\frac{\partial P(\mathbf{y};\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = -\sum_{i,j}^{N_e} w_{i,j} \frac{\partial \Delta \mathbf{y}_{i,j}^{\mathsf{T}}}{\partial \boldsymbol{\theta}} \boldsymbol{\Sigma}^{-1} \Delta \mathbf{y}_{i,j} = \mathbf{0}, \qquad (2)$$

where $\Delta \mathbf{y}_{i,j} = \mathbf{y}_i - \mathbf{y}_j$. By assuming that $\mathbf{y}_i$ can be expressed as a linear dependency on the motion parameters $\boldsymbol{\theta}$, Eq. (2) can be manipulated such that the motion parameters $\boldsymbol{\theta}$ can be iteratively solved by linear least-squares, *i.e.* $\boldsymbol{\theta}^* = \boldsymbol{\Psi}^{-1}\boldsymbol{\psi}$, where $\boldsymbol{\Psi}$ and $\boldsymbol{\psi}$ are a matrix and vector, respectively, that capture the underlying algebraic manipulations. The motion parameters $\boldsymbol{\theta}$ can thus be incrementally estimated by maintaining and updating the components $\boldsymbol{\psi}$ and $\boldsymbol{\Psi}$ separately, and applying some strategy to give more importance to recent events (*e.g.* decay factor).

3. Demo Setup

The setup consists of a laptop and a DAVIS346B event camera (similar to a DVS240 [2]) that undergoes some motion (Fig. 1 left). The motion parameters are then estimated by the incremental method briefly described in Section 2 and plotted using the PlotJuggler library[1] (Fig. 1 right). We also plot the ground-truth for rotation-like motion parameters, corresponding to the angular velocities provided by the camera IMU. Fig. 1 also depicts an example whereby the camera sequentially undergoes a large rotational motion around the (a) x-axis, (b) y-axis and (c) z-axis. Optionally, the generated events can also be shown by accumulating them onto the image plane, as well as the optical flow obtained from the estimated motion parameters.

Several motion models will be considered, including translational motion (2-DOF: up-down and left-right), rotational motion (3-DOF: rotation around the 3D axis), isometry transformation (3-DOF: translational motion plus rotation around the z-axis) and similarity transformation (4-DOF: isometry transformation plus isotropic scaling).

4. Practical Implementation

The demo is implemented in C++, and we use ROS[2] to communicate between the camera, the laptop and PlotJuggler. We maintain a fixed number of the most recent events ($\sim$1500 events for the demo), similar to a FIFO queue policy. The components $\boldsymbol{\Psi}$ and $\boldsymbol{\psi}$ are incrementally computed by considering a 9×9 spatial neighbourhood centred at the incoming event. These components are also decayed according to the factor $\exp\left(-(t_i - t_{i-1})n_{\text{decay}}\right)$, where t_i is the timestamp of the current event, t_{i-1} is the timestamp of the previous event and n_{decay} is incremented by one for each new event and is decayed according to the same factor.

[1] https://github.com/facontidavide/PlotJuggler

[2] Robotic Operating System: https://www.ros.org/

Table 1: Quantitative comparison between real-time (RT) and not real-time (NRT) processing, in terms of Root Mean Square (RMS) error, on real sequences [5].

	boxes_rotation		poster_rotation	
Approach	RMS (deg/s)	RMS %	RMS (deg/s)	RMS %
NRT	12.72	1.35	17.27	1.84
RT	43.81	5.97	44.69	4.77

The incremental motion estimation method has an almost constant processing time per event, which depends on the number of motion parameters. For example, for the rotational motion estimation, it takes approximately $3\mu s$ to process one event. However, in practice, more than one event can be triggered within $3\mu s$. To achieve real-time processing, we do not process certain events, modifying the event's sequential index pattern. To sequentially process one event at a time, an index variable is incremented by one. Instead, we increment the index variable by $\exp(a\Delta t)$, where $a = 5$ was set empirically, and Δt is the time difference between the current time and the time the current received event-based message was generated. Table 1 reports the respective performance on two real sequences [5].

References

[1] Ryad Benosman, Charles Clercq, Xavier Lagorce, Sio Hoi Ieng, Chiara Bartolozzi, Sio-Hoi Ieng, and Chiara Bartolozzi. Event-based visual flow. *IEEE Transactions on Neural Networks and Learning Systems*, 25(2):407–417, 2014. 1

[2] Christian Brandli, Raphael Berner, Minhao Yang, Shih-Chii Liu, and Tobi Delbruck. A 240 × 180 130 dB 3 µs Latency Global Shutter Spatiotemporal Vision Sensor. *IEEE Journal of Solid-State Circuits*, 49(10):2333–2341, 2014. 2

[3] Guillermo Gallego, Henri Rebecq, and Davide Scaramuzza. A Unifying Contrast Maximization Framework for Event Cameras, with Applications to Motion, Depth, and Optical Flow Estimation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3867–3876. IEEE, 2018. 1

[4] Hanme Kim, Stefan Leutenegger, and Andrew J. Davison. Real-Time 3D Reconstruction and 6-DoF Tracking with an Event Camera. In *European Conference on Computer Vision*, pages 349–364, 2016. 1

[5] Elias Mueggler, Henri Rebecq, Guillermo Gallego, Tobi Delbruck, and Davide Scaramuzza. The event-camera dataset and simulator: Event-based data for pose estimation, visual odometry, and SLAM. *International Journal of Robotics Research*, 36(2):142–149, 2017. 2

[6] Urbano Miguel Nunes and Yiannis Demiris. Entropy Minimisation Framework for Event-Based Vision Model Estimation. In *European Conference on Computer Vision*, pages 161–176, 2020. 1

[7] Alex Zihao Zhu, Nikolay Atanasov, and Kostas Daniilidis. Event-based feature tracking with probabilistic data association. In *IEEE International Conference on Robotics and Automation*, pages 4465–4470. IEEE, 2017. 1