High-speed event camera tracking

William Chamorro[1,2]
wchamorro@iri.upc.edu
Juan Andrade-Cetto[1]
cetto@iri.upc.edu
Joan Solà[1]
jsola@iri.upc.edu

[1] Institut de Robòtica i Informàtica Industrial, CSIC-UPC
LLorens i Artigas 4-6
Barcelona - Spain

[2] Universidad UTE
Facultad de Ciencias e Ingeniería
Quito - Ecuador

Abstract

Event cameras are bioinspired sensors with reaction times in the order of microseconds. This property makes them appealing for use in highly-dynamic computer vision applications. In this work, we explore the limits of this sensing technology and present an ultra-fast tracking algorithm able to estimate six-degree-of-freedom motion with dynamics over 25.8 g, at a throughput of 10 kHz, processing over a million events per second. Our method is capable of tracking either camera motion or the motion of an object in front of it, using an error-state Kalman filter formulated in a Lie-theoretic sense. The method includes a robust mechanism for the matching of events with projected line segments with very fast outlier rejection. Meticulous treatment of sparse matrices is applied to achieve real-time performance. Different motion models of varying complexity are considered for the sake of comparison and performance analysis.

1 Introduction

Event cameras send independent pixel information as soon as their intensity change exceeds an upper or lower threshold, generating “ON" or “OFF" events respectively (see Fig.1). In contrast to conventional cameras –in which full images are given at a fixed frame rate–, in event cameras, intensity-change messages come asynchronously per pixel, this happening at the microsecond resolution. Moreover, event cameras exhibit high dynamic range in luminosity (*e.g.* 120dB for the Davis 240C model [1] used in this work). These two assets make them suitable for applications at high-speed and/or with challenging illumination conditions (low illumination levels or overexposure). Emerging examples of the use of these cameras in mobile robotics are: event-based optical flow for micro-aerial robotics [18], obstacle avoidance [3, 15], simultaneous localization and mapping (SLAM) [24] [12], and object recognition [17], among others.

We are interested in accurately tracking high-speed 6DoF motion with an event camera. This type of sensors has been used in the past for the tracking of motion. For instance, 2D position estimates are tracked with the aid of a particle filter in [22]. The method was later extended into an SO(2) SLAM system in which a planar map of the ceiling was reconstructed [23]. Another SLAM system that tracks only camera rotations and builds a

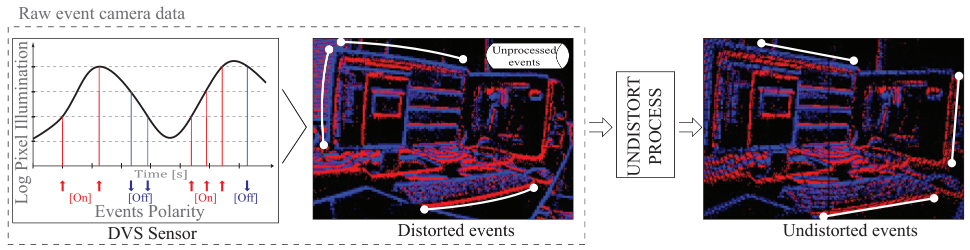

Figure 1: Working principle of event cameras (left) with distorted (center) and undistorted output (right).

high-resolution spherical mosaic of the scene was presented in [9]. Full 3D tracking is proposed in [10] where three interleaved probabilistic filters perform pose tracking, scene depth and log intensity estimation as part of a SLAM system. These systems were not designed with high-speed motion estimation in mind.

More related to our approach is the full 3D tracking for high-speed maneuvers of a quadrotor with an event camera presented in [14], extended later to a continuous-time trajectory estimation solution [16]. The method is similar to ours in that it localizes the camera with respect to a known wire-frame model of the scene by minimizing point-to-line reprojection errors. In that work, the model being tracked is planar, whereas we are able to localize with respect to a 3D model. That system was later modified to work with previously built photometric depth maps [8]. Non-linear optimization was included in a more recent approach [7];in this case, the tracking was performed in a sparse set of reference images, poses and depth maps, by having an a priori initial pose guess and taking into account the event generation model to reduce the number of outliers. This event generation model was initially stated in [7] for tracking position and velocity in textured known environments. In a more recent contribution, a parallel tracking and mapping system following a geometric, semi-dense approach was presented in [19]. The pose tracker is based on edge-map alignment using inverse compositional Lucas-Kanade method; additionally, the scene depth is estimated without intensity reconstruction. In that work, pose estimates are computed at a rate of 500 Hz.

In the long run, we are also interested in developing a full event-based SLAM system with parallel threads for tracking and mapping, that is able to work in real-time on a standard CPU. Since event cameras naturally respond to edges in the scene, the map, in our case, is made of a set of 3D segments sufficiently scattered and visible to be tracked. This work deals with the tracking part, and thus such map is assumed given. With fast motion applications in mind, our tracking thread is able to produce pose updates in the order of tens of kHz on a standard CPU, 20 times faster than [19], is able to process over a million events per second and can track motion direction shifts above 15Hz and accelerations above 25.8 g.

The main contribution of this paper is first to present a new event-driven Lie-EKF formulation to track the 6DOF pose of a camera in very high dynamic conditions -that runs in real time (10kHz throughput). The use of Lie theory in our EKF implementation allows elegant handling of derivatives and uncertainties in the SO(3) manifold when compared to the classical error-state EKF. Then we propose a novel fast data association mechanism that robustly matches events to projected 3D line-based landmarks with fast outlier rejection. It reaches real-time performance for over a million of events per second and hundreds of landmarks on a standard CPU. Finally the benchmarking of several filter formulations including Lie versus classic EKF, three motion models and two projection models, adding up to a total of 12 filter variants.

2 Motion estimation

The lines in the map are parametrized by their endpoints $\mathbf{p}_{\{1,2\}} = (x,y,z)_{\{1,2\}}$ expressed in the object's reference frame. We assume the camera is calibrated, and the incoming events are immediately corrected for lens radial distortion using the exact formula in [5].

The state vector $\mathbf{x}$ represents either the camera state respective to a static object, or the object state respective to a static camera. This model duplicity will be pertinent for the preservation of camera integrity in the experimental validation, where tracking very high dynamics will be done by moving the object and not the camera.

To bootstrap the filter's initial pose, we use the camera's grayscale images. FAST corners [6] are detected in this 2D image and matched to those in the 3D predefined map. The initial pose is then computed using the PnP algorithm [11]. After this initial bootstrapping process, the grayscale images are no longer used.

2.1 Prediction step

The state evolution has the form $\mathbf{x}_k = f(\mathbf{x}_{k-1}, \mathbf{n}_k)$, where $\mathbf{n}_k$ is the system Gaussian perturbation. The error state $\delta\mathbf{x}$ lies in the tangent space of the state, and is modeled as a Gaussian variable with mean $\bar{\delta\mathbf{x}}$ and covariance $\mathbf{P}$.

For the sake of performance evaluation, we implemented three different motion models: constant position (CP), constant velocity (CV), and constant acceleration (CA). These are detailed in Tab. 1, where $\mathbf{r}$ represents position, $\mathbf{R}$ orientation, $\mathbf{v}$ linear velocity, $\boldsymbol{\omega}$ angular velocity, $\mathbf{a}$ linear acceleration and $\boldsymbol{\alpha}$ angular acceleration. Their Gaussian perturbations are $\mathbf{r_n}, \boldsymbol{\theta}_\mathbf{n}, \mathbf{v_n}, \boldsymbol{\omega}_\mathbf{n}, \mathbf{a_n}$, and $\boldsymbol{\theta}_\mathbf{n}$. The orientation $\mathbf{R}$ belongs to the $SO(3)$ Lie group, and thus $\boldsymbol{\omega}$ lies in its tangent space $\mathfrak{so}(3)$, although we express it in the Cartesian $\mathbb{R}^3$. The operator $\oplus$ represents the right plus operation for $SO(3)$, $\mathbf{R} \oplus \boldsymbol{\theta} \triangleq \mathbf{R}\,\mathrm{Exp}(\boldsymbol{\theta})$, with $\mathrm{Exp}(\cdot)$ the exponential map given by the Rodrigues formula [21].

The error's covariance propagation is $\mathbf{P}_k = \mathbf{F}\mathbf{P}_{k-1}\mathbf{F}^T + \mathbf{Q} \in \mathbb{R}^{m \times m}$ with m equal to 6, 12 or 18 for the CP, CV, and CA models, respectively. $\mathbf{F}$ is the Jacobian of f with respect to $\mathbf{x}$, and $\mathbf{Q}$ is the perturbation covariance. Computation for all the motion models is greatly accelerated by exploiting the sparsity of the Jacobian $\mathbf{F}$ and the covariance $\mathbf{Q}$. We partition

Table 1: State transition for **CP**, **CV** and **CA** motion models. Right: error-state partition.

$\mathbf{x}_t$	$=$	$f(\cdots$	**CP**	**CV**	**CA** $)$	$\delta\mathbf{x}_k$
$\mathbb{R}^3 \ni \mathbf{r}_k$	$=$	$\mathbf{r}_{k-1} +$	$\mathbf{r_n}$	$\mathbf{v}_{k-1}\Delta t$	$\mathbf{v}_{k-1}\Delta t + \frac{1}{2}\mathbf{a}_{k-1}\Delta t^2$	$\delta\mathbf{r}_k \in \mathbb{R}^3$
$SO(3) \ni \mathbf{R}_k$	$=$	$\mathbf{R}_{k-1} \oplus ($	$\boldsymbol{\theta}_\mathbf{n}$	$\boldsymbol{\omega}_{k-1}\Delta t$	$\boldsymbol{\omega}_{k-1}\Delta t + \frac{1}{2}\boldsymbol{\alpha}_{k-1}\Delta t^2\ )$	$\delta\boldsymbol{\theta}_k \in \mathbb{R}^3$
$\mathbb{R}^3 \ni \mathbf{v}_k$	$=$	$\mathbf{v}_{k-1} +$		$\mathbf{v_n}$	$\mathbf{a}_{k-1}\Delta t$	$\delta\mathbf{v}_k \in \mathbb{R}^3$
$\mathfrak{so}(3) \ni \boldsymbol{\omega}_k$	$=$	$\boldsymbol{\omega}_{k-1} +$		$\boldsymbol{\omega}_\mathbf{n}$	$\boldsymbol{\alpha}_{k-1}\Delta t$	$\delta\boldsymbol{\omega}_k \in \mathbb{R}^3$
$\mathbb{R}^3 \ni \mathbf{a}_k$	$=$	$\mathbf{a}_{k-1} +$			$\mathbf{a_n}$	$\delta\mathbf{a}_k \in \mathbb{R}^3$
$\mathbb{R}^3 \ni \boldsymbol{\alpha}_k$	$=$	$\boldsymbol{\alpha}_{k-1} +$			$\boldsymbol{\alpha}_\mathbf{n}$	$\delta\boldsymbol{\alpha}_k \in \mathbb{R}^3$

these matrices and $\mathbf{P}$ in 3×3 blocks,

$$\mathbf{F} = \begin{bmatrix} \mathbf{I} & 0 & \mathbf{I}\Delta t & 0 & \mathbf{I}\Delta t^2 & 0 \\ 0 & \mathbf{J}^{\mathbf{R}}_{\mathbf{R}} & 0 & \mathbf{J}^{\mathbf{R}}_{\boldsymbol{\omega}} & 0 & \mathbf{J}^{\mathbf{R}}_{\boldsymbol{\alpha}} \\ 0 & 0 & \mathbf{I} & 0 & \mathbf{I}\Delta t & 0 \\ 0 & 0 & 0 & \mathbf{I} & 0 & \mathbf{I}\Delta t \\ 0 & 0 & 0 & 0 & \mathbf{I} & 0 \\ 0 & 0 & 0 & 0 & 0 & \mathbf{I} \end{bmatrix} \quad \begin{matrix} \text{CP} \\ \\ \text{CV} \\ \\ \text{CA} \end{matrix} \quad \mathbf{P} = \begin{bmatrix} \mathbf{P_{rr}} & \mathbf{P_{rR}} & \mathbf{P_{rv}} & \mathbf{P}_{\mathbf{r}\boldsymbol{\omega}} & \mathbf{P_{ra}} & \mathbf{P}_{\mathbf{r}\boldsymbol{\alpha}} \\ \mathbf{P_{Rr}} & \mathbf{P_{RR}} & \mathbf{P_{Rv}} & \mathbf{P}_{\mathbf{R}\boldsymbol{\omega}} & \mathbf{P_{Ra}} & \mathbf{P}_{\mathbf{R}\boldsymbol{\alpha}} \\ \mathbf{P_{vr}} & \mathbf{P_{vR}} & \mathbf{P_{vv}} & \mathbf{P}_{\mathbf{v}\boldsymbol{\omega}} & \mathbf{P_{va}} & \mathbf{P}_{\mathbf{v}\boldsymbol{\alpha}} \\ \mathbf{P}_{\boldsymbol{\omega}\mathbf{r}} & \mathbf{P}_{\boldsymbol{\omega}\mathbf{R}} & \mathbf{P}_{\boldsymbol{\omega}\mathbf{v}} & \mathbf{P}_{\boldsymbol{\omega}\boldsymbol{\omega}} & \mathbf{P}_{\boldsymbol{\omega}\mathbf{a}} & \mathbf{P}_{\boldsymbol{\omega}\boldsymbol{\alpha}} \\ \mathbf{P_{ar}} & \mathbf{P_{aR}} & \mathbf{P_{av}} & \mathbf{P}_{\mathbf{a}\boldsymbol{\omega}} & \mathbf{P_{aa}} & \mathbf{P}_{\mathbf{a}\boldsymbol{\alpha}} \\ \mathbf{P}_{\boldsymbol{\alpha}\mathbf{r}} & \mathbf{P}_{\boldsymbol{\alpha}\mathbf{R}} & \mathbf{P}_{\boldsymbol{\alpha}\mathbf{v}} & \mathbf{P}_{\boldsymbol{\alpha}\boldsymbol{\omega}} & \mathbf{P}_{\boldsymbol{\alpha}\mathbf{a}} & \mathbf{P}_{\boldsymbol{\alpha}\boldsymbol{\alpha}} \end{bmatrix} \tag{1}$$

We follow [21] to compute all the non-trivial Jacobian blocks of $\mathbf{F}$, which correspond to the $SO(3)$ manifold. Using the notation $\mathbf{J}^{\mathbf{a}}_{\mathbf{b}} \triangleq \partial\mathbf{a}/\partial\mathbf{b}$, we have

$$\mathbf{J}^{\mathbf{R}}_{\mathbf{R}} = \mathrm{Exp}(\boldsymbol{\omega}\Delta t)^{\top} \quad , \quad \mathbf{J}^{\mathbf{R}}_{\boldsymbol{\omega}} = \mathbf{J}_r(\boldsymbol{\omega}\Delta t)\Delta t \quad \text{and} \quad \mathbf{J}^{\mathbf{R}}_{\boldsymbol{\alpha}} = \tfrac{1}{2}\mathbf{J}^{\mathbf{R}}_{\boldsymbol{\omega}}\Delta t \ , \tag{2}$$

where $\mathbf{J}_r(\cdot)$ is the right-Jacobian of $SO(3)$ in [21, eq. (143)], $\boldsymbol{\theta} = \boldsymbol{\omega}\Delta t \in \mathbb{R}^3$ is a rotation vector calculated as the angular velocity per time, and $[\cdot]_\times \in so(3)$ is a skew symmetric matrix. Notice that for CP we have $\mathbf{J}^{\mathbf{R}}_{\mathbf{R}} = \mathrm{Exp}(\mathbf{0})^\top = \mathbf{I}$ and thus $\mathbf{F}_{CP} = \mathbf{I}$.

The perturbation covariance $\mathbf{Q}$ is a diagonal matrix formed by the variances $(\sigma^2_{\mathbf{r}},\sigma^2_{\theta})$ for CP, $(\sigma^2_{\mathbf{v}},\sigma^2_{\omega})$ for CV and $(\sigma^2_{\mathbf{a}},\sigma^2_{\alpha})$ for CA, times Δt. For example, for CV we have $\mathbf{Q} = \mathrm{block\,diag}(\mathbf{0},\mathbf{0},\sigma^2_{\mathbf{v}}\mathbf{I},\sigma^2_{\omega}\mathbf{I})\Delta t$. The 3×3 blocks of $\mathbf{P}$ are propagated in such a way that trivial operations (add $\mathbf{0}$, multiply by $\mathbf{0}$ or by $\mathbf{I}$) are avoided, as well as the redundant computation of the symmetric blocks. For example the blocks $\mathbf{P_{Rr}}$, $\mathbf{P_{rR}}$ and $\mathbf{P_{vv}}$ in CV are propagated as, $\mathbf{P_{Rr}} \leftarrow \mathbf{J}^{\mathbf{R}}_{\mathbf{R}}(\mathbf{P_{Rr}} + \mathbf{P_{Rv}}\Delta t) + \mathbf{J}^{\mathbf{R}}_{\boldsymbol{\omega}}(\mathbf{P}_{\boldsymbol{\omega}\mathbf{r}} + \mathbf{P}_{\boldsymbol{\omega}\mathbf{v}}\Delta t)$, $\mathbf{P_{rR}} = \mathbf{P}^\top_{\mathbf{Rr}}$, and $\mathbf{P_{vv}} \leftarrow \mathbf{P_{vv}} + \sigma^2_{\mathbf{v}}\mathbf{I}\Delta t$.

2.2 Correction step

We have investigated the possibilities of either predicting and updating the filter for each event, or collecting a certain number of events in a relatively small window of time. Single-event updates are appealing for achieving event-rate throughput, but the amount of information of a single event is so small that this does not pay off. Instead, we here collect a number of events in a small window, make a single EKF prediction to the central time t_0 of the window, and proceed with updating with every single event as if it had been received at t_0. This reduces the number of prediction stages greatly and allows us also to perform a more robust data association.

After predicting the state to the center t_0 of the window Δt, all visible segments S_i, $i \in \{1..N\}$ are projected. We consider two projection models: a moving camera in a static world (3a) and a moving object in front of a static camera (3b), which are used according to the experiments detailed in Sec. 3,

$$\text{moving camera:} \quad \underline{\mathbf{u}}_j = \mathbf{K}\mathbf{R}^\top(\mathbf{p}_j - \mathbf{r}) \quad j \in \{1,2\} \qquad \in \mathbb{P}^2 \tag{3a}$$

$$\text{moving object:} \quad \underline{\mathbf{u}}_j = \mathbf{K}(\mathbf{r} + \mathbf{R}\mathbf{p}_j) \quad j \in \{1,2\} \qquad \in \mathbb{P}^2 \tag{3b}$$

$$\text{projected line:} \quad \mathbf{l} = \underline{\mathbf{u}}_1 \times \underline{\mathbf{u}}_2 = (a,b,c)^\top \qquad \in \mathbb{P}^2 \tag{4}$$

where $\underline{\mathbf{u}}_j = (u,v,w)^\top_j$ are the projections of the i-th segment's endpoints $\mathbf{p}_j \in \mathbb{R}^3$, $j \in \{1,2\}$, in projective coordinates, and $\mathbf{K}$ is the camera intrinsic matrix. Jacobians are also computed,

$$\mathbf{J}^{\mathbf{l}}_{\mathbf{r}} = \mathbf{J}^{\mathbf{l}}_{\mathbf{u}_i}\mathbf{J}^{\mathbf{u}_i}_{\mathbf{r}} \quad \text{and} \quad \mathbf{J}^{\mathbf{l}}_{\mathbf{R}} = \mathbf{J}^{\mathbf{l}}_{\mathbf{u}_i}\mathbf{J}^{\mathbf{u}_i}_{\mathbf{R}} \qquad \in \mathbb{R}^{3\times 3} \ , \tag{5}$$

having $\mathbf{J}^{\mathbf{l}}_{\mathbf{u}_1} = -[\mathbf{u}_2]_\times$, $\mathbf{J}^{\mathbf{l}}_{\mathbf{u}_2} = [\mathbf{u}_1]_\times$, $\mathbf{J}^{\mathbf{u}_1}_{\mathbf{r}} = -\mathbf{K}\mathbf{R}^\top$ for (3a), $\mathbf{J}^{\mathbf{u}_i}_{\mathbf{r}} = \mathbf{K}$ for (3b), and $\mathbf{J}^{\mathbf{u}_i}_{\mathbf{R}}$ is the Jacobian of the rotation action computed in the Lie-theoretic sense [21], which for the two projection models becomes

$$\text{moving camera:} \quad \mathbf{J}^{\mathbf{u}_i}_{\mathbf{R}} = \mathbf{K}[\mathbf{R}^\top(\mathbf{p}_i - \mathbf{r})]_\times \quad \in \mathbb{R}^{3\times3} \tag{6a}$$

$$\text{moving object:} \quad \mathbf{J}^{\mathbf{u}_i}_{\mathbf{R}} = -\mathbf{K}\mathbf{R}[\mathbf{p}_i]_\times \quad \in \mathbb{R}^{3\times3}\,. \tag{6b}$$

Then, each undistorted event $\mathbf{e} = (u_e, v_e)^\top$ in the window is matched to a single projected segment $\mathbf{l}$. On success (see Sec. 2.3 below), we define the event's innovation as the Euclidean distance to the matched segment on the image plane, with a measurement noise $n_d \sim \mathcal{N}(0, \sigma_d^2)$,

$$\text{distance innovation}: \quad z = d(\mathbf{e}, \mathbf{l}) = \frac{\underline{\mathbf{e}}^\top \mathbf{l}}{\sqrt{a^2 + b^2}} \quad \in \mathbb{R}\,, \tag{7}$$

where $\underline{\mathbf{e}} = (u_e, v_e, 1)^\top$. The scalar innovation variance is given by $Z = \mathbf{J}^z_\mathbf{x}\mathbf{P}\mathbf{J}^{z\,\top}_\mathbf{x} + \sigma_d^2 \in \mathbb{R}$, where the Jacobian $\mathbf{J}^z_\mathbf{x}$ of the innovation with respect to the state is a sparse row-vector with zeros in the velocity and acceleration blocks for the larger CV and CA models,

$$\mathbf{J}^z_\mathbf{x} = \begin{bmatrix}\mathbf{J}^z_\mathbf{r} & \mathbf{J}^z_\mathbf{R} & \mathbf{0} & \cdots & \mathbf{0}\end{bmatrix} = \begin{bmatrix}\mathbf{J}^z_\mathbf{l}\mathbf{J}^\mathbf{l}_\mathbf{r} & \mathbf{J}^z_\mathbf{l}\mathbf{J}^\mathbf{l}_\mathbf{R} & \mathbf{0} & \cdots & \mathbf{0}\end{bmatrix} \quad \in \mathbb{R}^{1\times m}\,, \tag{8}$$

with $\mathbf{J}^z_\mathbf{l} = \underline{\mathbf{e}}^\top / \sqrt{a^2+b^2}$ the Jacobian of (7). At this point an individual compatibility test on the Mahalanobis norm of the innovation is evaluated, $\frac{z^2}{Z} < n_\sigma^2$, with $n_\sigma \sim 2$. Upon satisfaction, the Lie-EKF correction is applied:

$$\begin{array}{llllll} \text{a)} & \text{Kalman gain}: & \mathbf{k} = \mathbf{P}\mathbf{J}^{z\,\top}_\mathbf{x} Z^{-1} & \text{c)} & \text{State update}: & \mathbf{x} \leftarrow \mathbf{x} \oplus \delta\mathbf{x} \\ \text{b)} & \text{Observed error}: & \delta\mathbf{x} = \mathbf{k}z & \text{d)} & \text{Cov. update}: & \mathbf{P} \leftarrow \mathbf{P} - \mathbf{k}Z\mathbf{k}^\top\,, \end{array} \tag{9}$$

where the state update c) is implemented by a regular sum for the state blocks $\{\mathbf{r}, \mathbf{v}, \boldsymbol{\omega}, \boldsymbol{\alpha}\}$ and by the right-plus $\mathbf{R}\mathrm{Exp}(\delta\theta)$ for $\mathbf{R} \in SO(3)$, as needed for the model in turn (CP, CV, or CA). We remark for implementation purposes affecting execution speed that the Kalman gain $\mathbf{k}$ is an $m-$vector, that to compute Z and (9a) we again exploit the sparsity of $\mathbf{J}^z_\mathbf{x}$, as we did in 2.1, and that Z^{-1} is the inverse of a scalar.

2.3 Fast event-to-line matching

The Lie-EKF update described above is preceded by event outlier detection and rejection. The goal is to discard or validate events rapidly before proceeding to the update. We use image tessellation to accelerate the search for event-to-line candidates. To do so, for each temporal window of events, we first identify the visible segments in the map and project them using either (3a) or (3b) as appropriate. Fig. 2(a) displays a capture of a temporal window of events of $100\mu s$.

The image is tessellated in $m \times n$ (reasonably squared) cells, each one having a list of the segments crossing it. These lists are re-initialized at the arrival of each new window of events. The cells (C_u, C_v) crossed by a segment are identified by computing the segment intersections with the horizontal and vertical tessellation grid lines. We use the initial segment endpoint coordinates (u_0, v_0), the line parameters (4) and the image size $w \times h$, see Fig 2(b). The computation of the crossed cells coordinates departs from a cell given by the initial segment

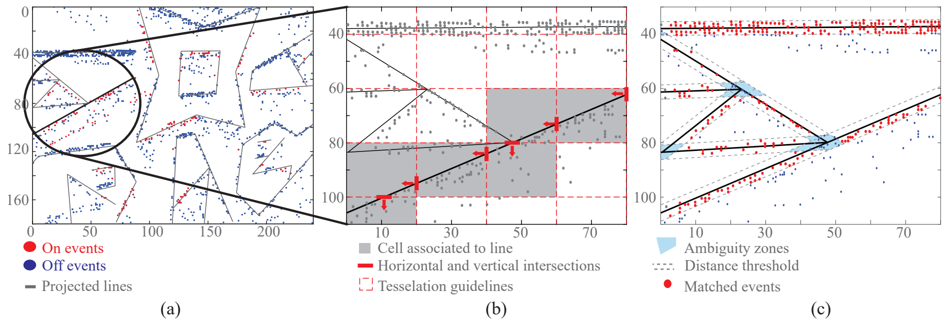

Figure 2: Data association process: (a) event window sample with projected lines, (b) cell identification for a single line based on the tessellation guidelines, (c) thresholding and ambiguity removal.

endpoint, $C_u^0 = \lceil u_0 m/w \rceil$ and $C_v^0 = \lceil (v_0 n/h \rceil$, where $\lceil C \rceil \triangleq \text{ceil}(C)$. Then we sequentially identify all horizontal and vertical intersections,

$$\text{Horiz:} \begin{cases} C_v^i = i \\ C_u^i = \lceil (-bhi - cn)m/anw \rceil \end{cases} \qquad \text{Vert:} \begin{cases} C_u^j = j \\ C_v^j = \lceil (-awj - cm)n/bmh \rceil, \end{cases} \tag{10}$$

where the iterators i and j keep track of the horizontal and vertical intersections. Their values start from C_v^0 and C_u^0, respectively and are increased or decreased by one in each iteration until reaching the opposite endpoint cell location. The sign of the increment depends on the difference between the first and last endpoint cell coordinates.

For each event in the temporal window we must check whether it has a corresponding line match in its corresponding tessellation cell. Although we are capable of processing all events at the rate of millions per second, there might be cases in which this is not achievable due to a sudden surge of incoming events. This depends on the motion model used, the scene complexity, or the motion dynamics. We might need to leave out up to 1/10th of events on average in the most demanding conditions (see last row of Tab. 3), and to do so unbiasedly, we keep track of execution time and skip the event if its timestamp is lagging more than $1\mu s$ from the current time.

Each unskipped event inside each cell is compared only against the segments that are within that cell. This greatly reduces the combinatorial explosion of comparing N segments with a huge number M of events from $O(M \times N)$ to the smaller cost of updating the cells' segments lists, which is only $O(N)$. The (very small) number of match segment candidates for each event are sorted from min to max distance. To validate a match between an event and its closest segment the following three conditions (evaluated in this order) must be met, see Fig 2(c): a) the distance d_1 (7) to the closest segment is *below* a predefined threshold, $d_1 < \alpha$; b) the distance d_2 to the second closest segment is *above* another predefined threshold, $d_2 > \beta$; and c) the orthogonal projection of the event onto the segment falls between the two endpoints, $0 < \frac{\mathbf{v}_1^\top \mathbf{v}_2}{\mathbf{v}_1^\top \mathbf{v}_1} < 1$, where $\mathbf{v}_1 = \mathbf{u}_2 - \mathbf{u}_1$, $\mathbf{v}_2 = \mathbf{e} - \mathbf{u}_1$, and $\mathbf{u}_i$ are the endpoints in pixel coordinates. Events that pass all conditions are used for EKF update as described in Sec. 2.2.

Table 2: Perturbation and noise parameters.

σ	Value	
$\sigma_\mathbf{r}$	0.03	m/s$^{1/2}$
σ_θ	0.3	rad/s$^{1/2}$
$\sigma_\mathbf{v}$	3	m/s$^{3/2}$
σ_ω	10	rad/s$^{3/2}$
$\sigma_\mathbf{a}$	80	m/s$^{5/2}$
σ_α	300	rad/s$^{5/2}$
σ_d	3.5	pixels

Table 3: RMSE mean values and timings. **L**: Lie parameterization, and **Cl**: classic algebra.

Metric	**CP+L**	**CV+L**	**CA+L**	**CP+Cl**	**CV+Cl**	**CA+Cl**
x (m)	0.0149	0.0091	0.0095	0.0162	0.0093	0.0106
y (m)	0.0125	0.0085	0.0081	0.0119	0.0086	0.0088
z (m)	0.0167	0.0111	0.0012	0.0171	0.0121	0.0113
ϕ (rad)	1.2205	0.7522	0.8333	1.2729	0.8613	0.9539
θ (rad)	1.4569	0.9842	1.0209	1.2729	1.2366	1.2645
ψ (rad)	1.2955	0.9252	0.8066	1.1549	1.1201	0.9902
T_{proc} (μs)	0.32	0.46	0.72	0.29	0.42	0.64
N_{events} (%)	97.73	90.96	85.51	98.06	92.68	89.09

3 Experiments and results

Our algorithm is set up with a temporal event-window size of $\Delta t = 100\mu s$. The continuous-time perturbation parameters of the motion models (see Sec. 2.1) are listed in Tab. 2, and the outlier rejection thresholds are set at $\alpha = 2.5$ pixels, and $\beta = 3.5$ pixels. These parameters were set in accordance with the velocities and accelerations expected. The experiments were carried out with different random camera hand-movements and a four-bar mechanism to test the tracking limits of our approach. For future comparisons, the dataset used to generate this results, parametrized maps using endpoints, camera calibration parameters, and detailed information about data format is available at https://www.iri.upc.edu/people/wchamorro/.

We make use of our C++ header-only library "manif" [4] for ease of Lie theory computations. The event-rate tracker runs single-threaded on standard PC hardware with Ubuntu 16.04.5 LTS and ROS Kinetic.

3.1 Position and orientation RMSE evaluation

The RMSE evaluation allow us to statistically determine the accuracy and consistency of the tracker. We execute $N = 10$ Monte Carlo runs of different motion sequences of about 60 s of duration in different conditions such as: speed changes from low (0.5 m/s, 3 rad/s approx. avg.) to fast (1 m/s, 8 rad/s approx. avg.), moving the camera manually inside the scene in random trajectories; aleatory lightning changes, turning on/off the laboratory lights and strong rotation changes.

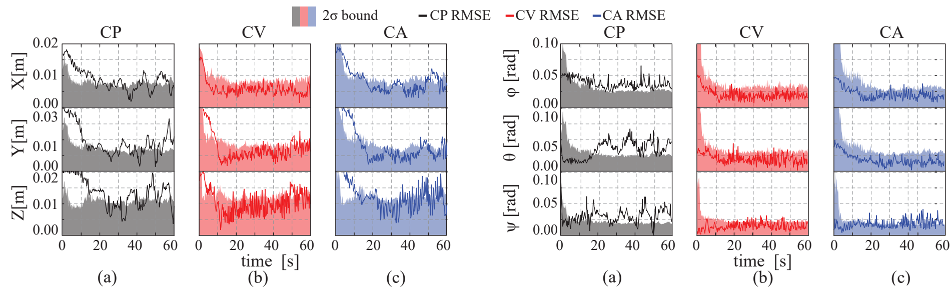

Figure 3: RMS errors and 2-sigma bounds: (a-c) position, (d-f) orientation.

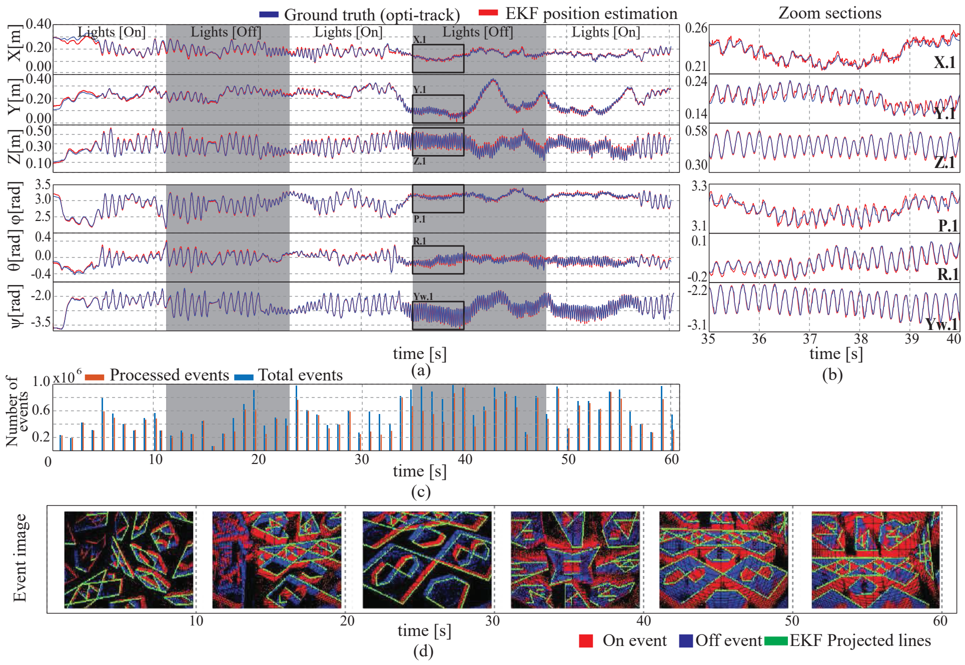

Figure 4: (a) Strong hand shake (∼ 6Hz) sequence example (using **CV+L**), (b) with zoom in the high speed zone, (c) event quantification and (d) visual output snapshots at a given time.

In this evaluation, we use the projection model (3a); i.e., the camera is moving in a static world. From the 10 runs, we measure the root mean square error (RMSE) of each component of the camera pose and plot it in Fig. 3. To analyze consistency of the filter, the errors obtained are compared against their 2-sigma bounds as in [20]. An OptiTrack motion capture system calibrated with spherical reflective references will provide the ground truth to analyze the event-based tracker performance.

For the sake of comparison, we also implemented the classic ES-EKF using quaternions, where Jacobians are obtained using first-order approximations. The error evaluation for the various filter variants tested are summarized in Tab. 3.

The overall results show a small but noticeable improvement in accuracy when the tracker is implemented with Lie groups, where the CV model has the best response. Though the Lie approach is somewhat slower, this can be taken as the price to pay for improved accuracy. During the RMSE evaluation, CV and CA errors were mostly under the 2-sigma bound (see Fig. 3 (b,c,e,f)) indicating a sign of consistency. On the other hand, the error using the CP model is shown to exceed the 2-sigma bound repeatedly. This situation was evidenced during the experiments by observing less resilience of the tracker in high dynamics (see Fig. 3 (a,d)).

In all cases, per-event total processing time T_{proc} falls well bellow the microsecond, where, on average, less than $0.1\,\mu$s of this time is spent performing line-event matching, the rest being spent in prediction and correction operations. With this, the tracker is capable of treating between 89.1% and 97.7% of the incoming data, depending on the motion model and state parameterization used, reaching real-time performance, and producing pose updates at the rate of 10 kHz, limited only by the chosen size of the time window of events of 100 μs.

A comparison of the tracker performance versus the OptiTrack ground truth is shown in Fig. 4 for the best performing motion model and state parameterization combination: constant velocity with Lie groups. In this case, the camera is hand-shaked by a human in front of the scene. The frequency of the motion signal increases from about 1 Hz to 6 Hz, the fastest achievable with a human hand-shake of the camera. The camera pose is accurately tracked despite the sudden changes in motion direction, where the most significant errors –in the order of mm– are observed precisely in these zones where motion changes direction (see Fig 4(a,b)).

Illumination changes were produced by turning on and off the lights in the laboratory with no noticeable performance degradation in the tracking nor the event production (see grey shaded sections in Figs. 4(a),(c)), which reached peaks of about one million events per second with the most aggressive motion dynamics (see zoomed-in region in Fig. 4(b) and (c)). The green lines in the snapshots in Fig. 4(d) are the projected map segments using the estimated camera pose.

3.2 High speed tracking

Our aim is to explore the limits of this sensing technology and submit the event camera to the highest dynamics it is able to track, and provide a mean of comparison with other approaches in terms of speed. To protect the camera from destructive vibrations, we switch now to a motion model in which the camera is stationary and the tracked object moves in

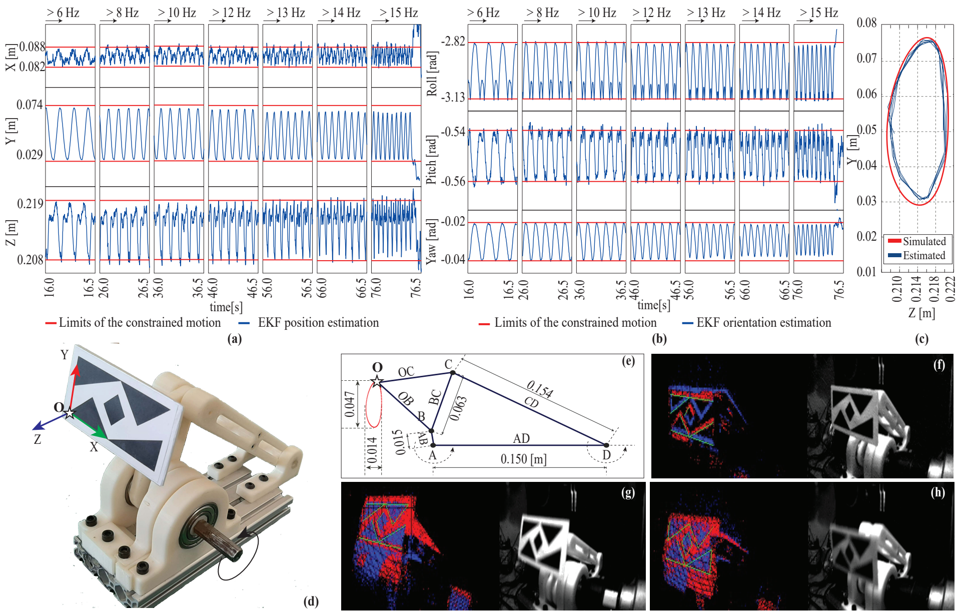

Figure 5: High dynamics position and orientation evaluation using **CV + L**: (a,b), poses up to 950 rpm (15.8 Hz) were accurately estimated before the tracking disengaged, (c) Z-Y trajectory (d) constrained four-bar motion mechanism, (e) mechanism dimensions and (f-h) visual snapshots of the tracker for crank angular speeds of 300, 500 and 800 rpm respectively.

front of it, using Eq. 3b for the projections.

This experiment is reported for our best-achieving version of the tracker, CV with Lie parametrization. We built a constrained motion device (see Fig. 5(d)), consisting of a four-bar mechanism powered with a DC motor and dimensions stated in Fig. 5(e). The mechanism very-rapidly shakes a target made of geometric shapes, delimited by straight segments, in front of the camera.

A kinematic analysis performed for our mechanism using real dimensions gives a 4.7 cm maximum displacement of the target reference frame (peak-to-peak). An evaluation point in the target (e.g. $\boldsymbol{O}$ in Fig. 5(d)) describes a motion with rotational and translational components, and has a simulated trajectory as the one in red dots in Fig. 5(e) (around the axes Y and Z). The estimated trajectory of the object in the axes Y and Z was compared to the simulated one considering the previously known dimensions and constrains of the mechanism. The estimated trajectory is within the motion limits and has a low associated error as can be seen in Fig. 5(c). For visualization purposes we plotted ten motion periods chosen randomly along the running time.

The camera is placed statically at a distance of roughly 20 cm in front of the target while a DC motor drives the mechanism. During the experiment (see Fig. 5(a,b)) its speed was increased gradually until about 950 rpm (15.8 Hz) where tracking performance starts to degrade. At such crank angular velocity, the velocity analysis of our mechanism reports a maximum target speed of 2.59 m/s. Linear target accelerations reach over 253.23 m/s^2 or 25.81 g, which are well above the expected range of the most demanding robotics applications, and above the maximum range of 16 g of the user-programmable IMU chip in the Davis 240c camera [13]. The green lines in Fig.5(f-h) are plot using the estimated camera pose.

4 Conclusions and future work

In this paper, an event-based 6-DoF pose estimation system is presented. Pose updates are produced at a rate of 10 kHz with an error-state Kalman filter. Several motion model variants are evaluated, and it is shown that the best performing motion model and pose parameterization combination is a constant velocity model with Lie-based parameterization. In order to deal with the characteristic *micro second* rate of event cameras, a very fast event-to-line association mechanism was implemented. Our filter is able to process over a million events per second, making it capable of tracking very high-speed camera or object motions without delay. Considering the low resolution of the camera, our system is able to track its position with high accuracy, in the order of a few mm with respect to a calibrated OptiTrack ground truth positioning system. Moreover, when subjected to extreme motion dynamics, the tracker was able to reach tracking performance for motions exceeding linear speeds of 2.5 m/s and accelerations over 25.8 g. Our future work will deal with the integration of this localization module into a full parallel tracking and mapping system based on events.

Acknowledgements

This work was partially supported by the EU H2020 project GAUSS (H2020-Galileo-2017-1-776293), by the Spanish State Research Agency through projects EB-SLAM (DPI2017-89564-P) and the María de Maeztu Seal of Excellence to IRI (MDM-2016-0656, and by a scholarship from SENESCYT, Republic of Ecuador to William Chamorro.

References

[1] Christian Brandli, Raphael Berner, Minhao Yang, Shih Chii Liu, and Tobi Delbruck. A 240 $\times$ 180 130 dB 3 μs latency global shutter spatiotemporal vision sensor. *IEEE J. Solid-State Circuits*, 49(10):2333–2341, 2014.

[2] Samuel Bryner, Guillermo Gallego, Henri Rebecq, and Davide Scaramuzza. Event-based direct camera tracking from a photometric 3D map using nonlinear optimization. In *IEEE Int. Conf. Robotics Autom.*, pages 325–331, 2019.

[3] Davide Scaramuzza Davide Falanga, Kevin Klever. Dynamic obstacle avoidance for quadrotors with event cameras. *Sci. Robotics*, 5(40):eaaz9712, 2020.

[4] Jeremie Deray and Joan Solà. manif: a small C++ header-only library for Lie theory. https://github.com/artivis/manif, jan 2019.

[5] Pierre Drap and Julien Lefèvre. An exact formula for calculating inverse radial lens distortions. *Sensors*, 16(6):807, 2016.

[6] Rosten Edward, Porter Reid, and Drummond Tom. Faster and better: A machine learning approach to corner detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 32(1):105–119, 2010.

[7] Guillermo Gallego, Christian Forster, Elias Mueggler, and Davide Scaramuzza. Event-based camera pose tracking using a generative event model. *arXiv: 1510.01972*, 1:1–7, 2015.

[8] Guillermo Gallego, Jon E.A. Lund, Elias Mueggler, Henri Rebecq, Tobi Delbruck, and Davide Scaramuzza. Event-based 6-DOF camera tracking from photometric depth maps. *pami*, 40(10):2402–2412, 2017.

[9] Hanme Kim, Ankur Handa, Ryad Benosman, Sio-Hoi Ieng, and Andrew J Davison. Simultaneous mosaicing and tracking with an event camera. *IEEE Journal of Solid-State Circuits*, 43:566–576, 2008.

[10] Hanme Kim, Stefan Leutenegger, and Andrew J Davison. Real-time 3D reconstruction and 6-DoF tracking with an event camera. In *Eur. Conf. Comput. Vis.*, pages 349–364, 2016.

[11] Pascal Lepetit, Vincent and Moreno-Noguer, Francesc and Fua. EPnP: An accurate O(n) solution to the PnP problem. *Int. J. Comput. Vision*, 81:155–166, 2009.

[12] Michael Milford, Hanme Kim, Stefan Leutenegger, and Andrew Davison. Towards visual SLAM with event-based cameras. *RSS Workshop on the Problem of Mobile Sensors*, 2015.

[13] Elias Mueggler. *Event-based Vision for High-Speed Robotics*. PhD thesis, University of Zurich, 2017.

[14] Elias Mueggler, Basil Huber, and Davide Scaramuzza. Event-based, 6-DOF pose tracking for high-speed maneuvers. In *IEEE/RSJ Int. Conf. Intell. Robots Syst.*, pages 2761–2768, 2014.

[15] Elias Mueggler, Nathan Baumli, Flavio Fontana, and Davide Scaramuzza. Towards evasive maneuvers with quadrotors using dynamic vision sensors. In *Eur. Conf. Mobile Robots*, pages 1–8, 2015.

[16] Elias Mueggler, Guillermo Gallego, and Davide Scaramuzza. Continuous-time trajectory estimation for event-based vision sensors. In *Robotics Sci. Syst. Conf.*, 2015.

[17] Garrick Orchard, Cedric Meyer, Ralph Etienne-Cummings, Christoph Posch, Nitish Thakor, and Ryad Benosman. HFirst: A temporal approach to object recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 37(10):2028–2040, 2015.

[18] Bas J. Pijnacker Hordijk, Kirk Y.W. Scheper, and Guido C.H.E. de Croon. Vertical landing for micro air vehicles using event-based optical flow. *J. Field Robotics*, 35(1): 69–90, 2018.

[19] Henri Rebecq, Timo Horstschaefer, Guillermo Gallego, and Davide Scaramuzza. EVO: A geometric approach to event-based 6-DOF parallel tracking and mapping in real time. *IEEE Robotics Autom. Lett.*, 2(2):593–600, 2017.

[20] Joan Solà, Teresa Vidal-Calleja, Javier Civera, and Jose Maria Martinez-Montiel. Impact of landmark parametrization on monocular EKF-SLAM with points and lines. *Int. J. Comput. Vision*, 97:339–368, 2011.

[21] Joan Solà, Jeremie Deray, and Dinesh Atchuthan. A micro Lie theory for state estimation in robotics. *arXiv: 1812.01537*, pages 1–16, 2018.

[22] David Weikersdorfer and Jorg Conradt. Event-based particle filtering for robot self-localization. In *IEEE Int. Conf. Robotics Biomim.*, pages 866–870, 2012.

[23] David Weikersdorfer, Raoul Hoffmann, and Jörg Conradt. Simultaneous localization and mapping for event-based vision systems. In *Int. Conf. Comput. Vis. Syst.*, pages 133–142, 2013.

[24] David Weikersdorfer, David Adrian, Daniel Cremers, and Jorg Conradt. Event-based 3D SLAM with a depth-augmented dynamic vision sensor. In *IEEE Int. Conf. Robotics Autom.*, pages 359–364, 2014.