Real-time Visual-Inertial Odometry for Event Cameras using Keyframe-based Nonlinear Optimization

Rebecq, Henri ; Horstschaefer, Timo ; Scaramuzza, Davide

Real-time Visual-Inertial Odometry for Event Cameras using Keyframe-based Nonlinear Optimization

Henri Rebecq
rebecq@ifi.uzh.ch
Timo Horstschaefer
horstschaefer@ifi.uzh.ch
Davide Scaramuzza
sdavide@ifi.uzh.ch

Robotics and Perception Group
University of Zurich
Zurich, Switzerland

Abstract

Event cameras are bio-inspired vision sensors that output pixel-level brightness changes instead of standard intensity frames. They offer significant advantages over standard cameras, namely a very high dynamic range, no motion blur, and a latency in the order of microseconds. We propose a novel, accurate tightly-coupled visual-inertial odometry pipeline for such cameras that leverages their outstanding properties to estimate the camera ego-motion in challenging conditions, such as high-speed motion or high dynamic range scenes. The method tracks a set of features (extracted on the image plane) through time. To achieve that, we consider events in overlapping spatio-temporal windows and align them using the current camera motion and scene structure, yielding motion-compensated event frames. We then combine these feature tracks in a keyframe-based, visual-inertial odometry algorithm based on nonlinear optimization to estimate the camera's 6-DOF pose, velocity, and IMU biases. The proposed method is evaluated quantitatively on the public Event Camera Dataset [19] and significantly outperforms the state-of-the-art [28], while being computationally much more efficient: our pipeline can run much faster than real-time on a laptop and even on a smartphone processor. Furthermore, we demonstrate qualitatively the accuracy and robustness of our pipeline on a large-scale dataset, and an extremely high-speed dataset recorded by spinning an event camera on a leash at 850 deg/s.

Multimedia Material

Multimedia Material. A supplemental video for this work is available: https://youtu.be/F3OFzsaPtvI

Henri Rebecq and Timo Horstschaefer contributed equally to this work.

1 Introduction

Event cameras, such as the Dynamic Vision Sensor (DVS) [16], work very differently from a traditional camera. They have *independent* pixels that only send information (called "events") in presence of brightness changes in the scene at the time they occur. Thus, the output is not an intensity image but a stream of asynchronous events at microsecond resolution, where each event consists of its space-time coordinates and the *sign* of the brightness change (i.e., no intensity). Event cameras have numerous advantages over standard cameras: a latency in the order of microseconds, low power consumption, and a very high dynamic range (130 dB compared to 60 dB of standard cameras). Most importantly, since all the pixels are independent, such sensors do not suffer from motion blur.

The task of estimating a sensor's ego-motion from a combination of images and measurements from an Inertial Measurement Unit (IMU), called Visual-Inertial Odometry (VIO), has important applications in various fields, for example augmented/virtual reality (AR/VR) applications. VIO has been thoroughly studied in the past decades, and is today a mature research field [4]. State-of-the-art VIO pipelines have shown impressive large-scale tracking results, with an overall drift below 0.5 % of the travelled distance ([15], [6]). However, VIO still fails in a number of situations, such as high-speed motions or high-dynamic range scenes. In the first case, large amounts of motion blur on the images spoil the visual information, forcing the pipeline to rely on integration of the IMU, resulting in large amounts of accumulated drift[1]. In the second case, due to the limited dynamic range of standard cameras, large regions on the image are either over-, or under-exposed, which reduces drastically the amount of information exploitable. It is in these challenging scenarios that the above-mentioned advantages of event cameras could be exploited to yield accurate and robust ego-motion estimation.

In this paper, we present a novel visual-inertial odometry (VIO) algorithm for event cameras. Our algorithm takes as input a stream of events and inertial measurements, and outputs camera poses at a rate proportional to the camera velocity. To achieve this, we track a set of features in the events, and fuse these feature tracks with the IMU measurements using a keyframe-based visual-inertial pipeline that uses nonlinear optimization.

Contribution. Our main contribution is a tightly-coupled visual-inertial odometry pipeline for event cameras that is significantly more accurate than the state-of-the-art [28], while being more efficient. More precisely, our contributions include:

- A novel feature tracker for event cameras that works on event frames, synthesized by fusing the events in a spatio-temporal window using the current estimate of the camera motion and the scene structure.
- The integration of these feature tracks in a keyframe-based, visual-inertial pipeline based on nonlinear optimization, yielding a robust and accurate VIO pipeline for event cameras.
- A quantitative evaluation of our pipeline compared to the state-of-the-art [28] on the public Event Camera Dataset [19], and some qualitative results on a large scale and a high-speed sequence.

[1]http://www.vectornav.com/support/library/imu-and-ins , see the first table under Case 1

2 Related Work

In the past decade, many works have considered to use event cameras for ego-motion estimation. Early works focused on addressing restricted, and easier instances of the problem: [5], [11], [9] and [22] showed how to do rotation-only (3 DOF) pose estimation, and [26] proposed a simultaneous tracking and mapping algorithm for event cameras that works for planar (2D) motion and planar scenes. Other authors have used complementary sensing modalities, additionally to an event camera: [27] used an event camera equipped with a depth sensor to jointly estimate the camera pose and 3D scene structure, and [13] proposed a low-latency, feature-based 6 DOF visual odometry pipeline that uses a frame-based sensor, where features are detected in the frames and tracked in the event stream. Event-based, 6-DOF visual odometry (using only an event camera) has been first shown only very recently: [12] proposed three parallel filters that jointly estimate the camera pose, 3D map of the scene, and image intensity, and [21] proposed a geometric approach that combines a global image alignment technique with an event-based reconstruction algorithm [20] to estimate the camera pose and 3D map of the scene without requiring image reconstruction. Few works have considered using an event camera with an IMU (the problem of event-based, visual-inertial odometry). [18] showed how to fuse events and inertial measurements into a continuous time framework. However, their approach is not suited for real-time usage because of the expensive optimization required to update the spline parameters upon receiving every event. Very recently, [28] proposed an event-based visual-inertial odometry algorithm, EVIO, that works in real-time (albeit, for limited motion speeds, and number of features). They proposed to track a set of features in the event stream using an iterative Expectation-Maximization scheme, that jointly solves for the feature appearance and optical flow. The feature tracks are then fed to an EKF filter to produce new pose estimates. EVIO is the closest approach to this work. In Section 5, we compare our approach to [28] in terms of accuracy, and show significant improvements compared to it.

3 Preliminaries

In this section, we introduce the notation that we will use throughout the rest of the paper. We also introduce the IMU model used, and provide formulas for discrete integration of the equations of motion.

Coordinate Frame Notation. A point P represented in a coordinate frame A is written as position vector ${}_A\mathbf{r}_P$. A transformation between frames is represented by a homogeneous matrix $\mathbf{T}_{AB}$ that transforms points from frame B to frame A. Its rotational part is expressed as a rotation matrix $\mathbf{R}_{AB} \in SO(3)$. Our algorithm uses a sensor composed of an event camera and an IMU rigidly mounted together. The sensor body is represented relative to an inertial world frame W. Inside it, we distinguish the camera frame C and the IMU-sensor frame S. To obtain $\mathbf{T}_{SC}$, an extrinsic calibration of the camera + IMU system must be performed, using for example the Kalibr toolbox [8].

IMU Model and Motion Integration. An IMU usually includes a 3-axis accelerometer and a 3-axis gyroscope, and allows measuring the rotational rate and the acceleration of the sensor with respect to an inertial frame. The measurements, ${}_S\tilde{\mathbf{a}}(t)$ and ${}_S\tilde{\omega}(t)$, are affected by additive white noise η and slowly varying sensor biases $\mathbf{b}$:

$$ {}_S\tilde{\omega}(t) = {}_W\omega(t) + \mathbf{b}_g(t) + \eta_g(t), \qquad {}_S\tilde{\mathbf{a}}(t) = \mathbf{R}_{WB}^T(t)\left({}_W\mathbf{a}(t) - {}_W\mathbf{g}\right) + \mathbf{b}_a(t) + \eta_a(t), \tag{1} $$

where ${}_W\mathbf{g}$ is the gravity vector in world coordinates.

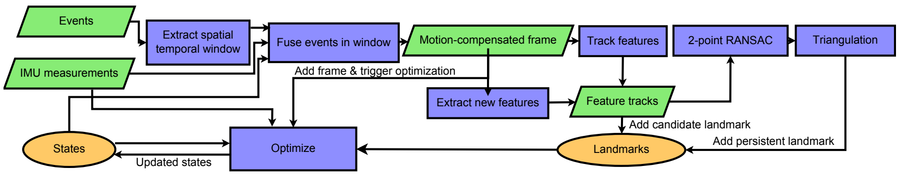


Figure 1: Overview of the proposed pipeline: (i) Events are grouped in spatio-temporal windows, and fused to build motion-compensated frames. (ii) New features are extracted if necessary; all feature tracks are updated, and outliers filtered using 2-point RANSAC. (iii) Feature tracks that can be triangulated are converted to persistent, and added to the map. The remaining ones are kept as candidate tracks. (iv) Selected frames are added to the optimizer, and trigger a global optimization.

Denoting a position vector and velocity, respectively as ${}_A\mathbf{r}$ and ${}_A\mathbf{v}$, the equations of motion can be numerically integrated as follows [6]:

$$\begin{aligned}
\mathbf{R}_{WB}(t+\Delta t) &= \mathbf{R}_{WB}(t)\exp\left({}_S\tilde{\boldsymbol{\omega}}(t)-\mathbf{b}_g(t)-\eta_{gd}(t)\Delta t\right)\\
{}_W\mathbf{v}(t+\Delta t) &= {}_W\mathbf{v}(t)+{}_W\mathbf{g}\Delta t+\mathbf{R}_{WB}(t)\left({}_S\tilde{\mathbf{a}}(t)-\mathbf{b}_a(t)-\eta_{ad}(t)\right)\Delta t\\
{}_W\mathbf{r}(t+\Delta t) &= {}_W\mathbf{r}(t)+{}_W\mathbf{v}(t)\Delta t+\tfrac{1}{2}{}_W\mathbf{g}\Delta t^2+\tfrac{1}{2}\mathbf{R}_{WB}(t)\left({}_S\tilde{\mathbf{a}}(t)-\mathbf{b}_a(t)-\eta_{ad}(t)\right)\Delta t^2)
\end{aligned} \tag{2}$$

where $\exp : se(3) \rightarrow SE(3)$ denotes the exponential map, and η_{ad}, η_{gd} are the noise variables.

Event Data. Let us denote the set of events observed as $\varepsilon = \{e_k\}$. The k^{th} event is represented as a tuple $\mathbf{e}_k = (\mathbf{x}_k, t_k, p_k)$, where $\mathbf{x}_k = (x_k, y_k)$ is the event location on the image plane, t_k its timestamp, and p_k its polarity.

4 Visual-Inertial Odometry with an Event Camera

Our visual-inertial odometry pipeline is classically composed of two parallel threads:

- the front-end (Section 4.1) takes a stream of events as input. It establishes feature tracks and triangulates landmarks, both of which are passed to the back-end.
- the back-end (Section 4.2) fuses the feature tracks, landmarks, and IMU measurements to continuously update the current and past sensor states.

Figure 1 gives an overview of the modules involved and their interactions. The rest of this section is organized as follows. Section 4.1.1 describes how we partition the event stream in spatio-temporal windows and synthesize motion-corrected event images, Section 4.1.2 provides details about feature tracking and landmark triangulation, and Section 4.1.3 gives additional implementation details. The back-end of our algorithm, a keyframe-based nonlinear optimization algorithm, is described in Section 4.2.

4.1 Front-end

Our pipeline takes a stream of events as input, and produces a set of motion-corrected event images (Section 4.1.1) that are fed to a visual odometry front-end (Sections 4.1.2 and 4.1.3).

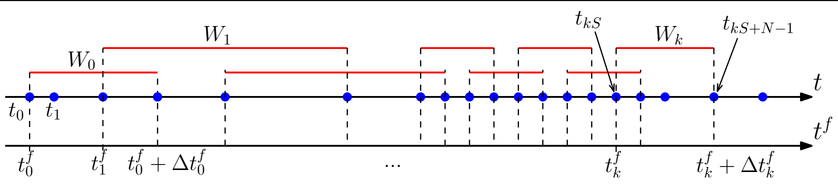


Figure 2: We split the event stream in a set of overlapping spatio-temporal windows. Events are depicted as blue dots on the timeline. The windows $\{W_k\}$ are marked in red ($N=4$, $S=2$ here). Note that the temporal size of each window is automatically adapted to the event rate.

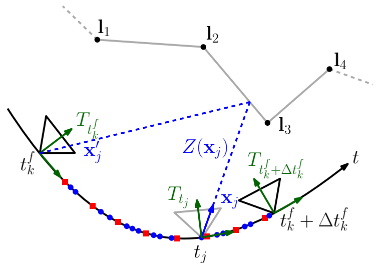


Figure 3: Motion correction: the inertial measurements in $[t_k^f, t_k^f+\Delta t_k^f]$ (red squares) are integrated to compute the relative transformation $T_{t_k^f, t_k^f+\Delta t_k^f}$. Each event (blue dot) $\mathbf{e}_j$ is reprojected to camera frame $C_{t_k^f}$ using the linearly interpolated pose T_{t_j} and the linearly interpolated depth $Z(\mathbf{x}_j)$.

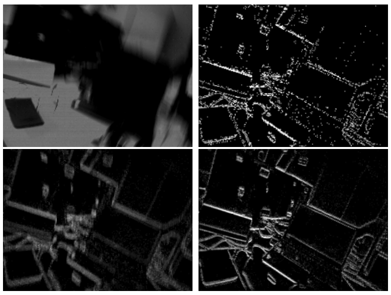


Figure 4: Synthesized event frames. From top left to bottom right: standard camera image; 3 000 events (noisy information); 30 000 events (motion-blurred image); 30 000 events with motion-correction.

4.1.1 Synthesis of Motion-Corrected Event Frames

Spatio-temporal Windows of Events. The set of observed events ε is split in a set of overlapping spatio-temporal windows $\{W_k\}$ (Fig. 2). The k^{th} window is defined as the set of events $W_k = \{e_{kS}, ..., e_{kS+N-1}\}$, where N is the window size parameter, and S a step size parameter that controls the amount of overlap between successive windows. Note that the start time $t_k^f := t_{kS}$ and duration of each window Δt_k^f are controlled by the events, which preserves the data-driven nature of the sensor. With this notation, W_k spans the time interval $[t_{kS}, t_{kS+N-1}] := [t_k^f, t_k^f + \Delta t_k^f]$.

From Events to Event Frames. A naive way to synthesize an event image from a window of events W_k would be to accumulate them as follows: $I_k(\mathbf{x}) = \sum_{e_j \in W_k} \delta(\mathbf{x} - \mathbf{x}_j)$, i.e., the intensity I at pixel $\mathbf{x}$ is simply the sum of the events that fired at the pixel location $\mathbf{x} = \mathbf{x}_j$. However, this yields event images that are not usable for reliable feature detection or tracking, as illustrated in Fig. 4: small window sizes do not convey enough information, while large window sizes induce motion blur.

Inspired by [9], we propose to locally correct the motion of each event according to its individual time stamp. This allows to synthesize motion-corrected event frames, used subsequently to establish feature tracks.

Motion Compensation of Events. We synthesize a motion-corrected event image I_k (see Figs. 3 and 4) as follows: $I_k(\mathbf{x}) = \sum_{e_j \in W_k} \delta(\mathbf{x} - \mathbf{x}'_j)$, where $\mathbf{x}'_j$ is the *corrected* event position, obtained by transferring event $\mathbf{e}_j$ to the reference camera frame $C_{t_k^f}$:

$$\mathbf{x}'_j = \pi\left(T_{t_k^f, t_j}(Z(\mathbf{x}_j)\pi^{-1}(\mathbf{x}_j))\right), \tag{3}$$

where $\pi(.)$ is the camera projection model, obtained from prior intrinsic calibration.

Adopting the short-hand notations: $T_{t_j} := T_{WC(t_j)}$, and $T_{t_i,t_j} := T_{C(t_i^f)C(t_j^f)}$, the incremental transformation $T_{t_k^f, t_k^f + \Delta t_k^f}$ is obtained through integration of the IMU measurements in $[t_k^f, t_k^f + \Delta t_k^f]$ using (2). The necessary starting pose $T_{t_k^f}$, and the IMU biases $\mathbf{b}_g, \mathbf{b}_a$ are known from the state estimation thread. The remaining quantities required to evaluate (3) are:

- $T_{t_k^f, t_j}$, which we linearly interpolate from $T_{t_k^f}$ and $T_{t_k^f + \Delta t_k^f}$, in the space of rigid-body motion transformations $SE(3)$.
- $Z(\mathbf{x}_j)$, which we estimate using 2D linear interpolation (on the image plane) of the current landmarks $\{\mathbf{l}_j\}$, reprojected on the current camera frame C_{t_j}.

In practice, we observed that using the median depth of the current landmarks instead of linearly interpolating the depth gives satisfactory results, at a lower computational cost.

4.1.2 Feature Detection and Tracking. Landmark Triangulation

Feature Detection. New features are detected whenever the number of feature tracks falls below a certain threshold, or if the current frame is selected as a keyframe (see Section 4.1.3). We use the FAST corner detector [23] on a motion-compensated event frame. We use a bucketing grid to ensure that the features are evenly distributed over the image.

Feature Tracking and Landmark Triangulation. We maintain two sets of landmarks: candidate landmarks, and persistent landmarks, whose 3D position in space has been successfully triangulated. Newly extracted features are initialized as candidate landmarks, and are tracked across event frames. As soon as a candidate landmark can be reliably triangulated, it is converted to a persistent landmark, and added to the local map.

Both types of landmarks are tracked from I_k to I_{k+1} using pyramidal Lukas-Kanade tracking [2]. The incremental transformation $T_{t_k^f, t_{k+1}^f}$ (integrated from the IMU measurements in $[t_k^f, t_{k+1}^f]$) is used to predict the feature position in I_{k+1}. The patches around each features are warped through an affine warp, computed using $T_{t_k^f, t_{k+1}^f}$, prior to pyramidal alignment. Landmarks that are not successfully tracked in the current frame are discarded immediately.

Outlier Filtering. We use two-point RANSAC [25] (using the relative orientation between the current frame and the last keyframe) to further filter out outlier feature tracks.

4.1.3 Additional Implementation Details

Keyframe Selection. A new keyframe is selected either when the number of tracked features falls below a threshold, or when the distance to the last keyframe (scaled by the median scene depth) reaches a minimum threshold.

Initialization. To initialize our pipeline we add the first frames to the back-end without initializing any feature track. The back-end in turn estimates the initial attitude of the sensor by observing the gravity direction. The displacement between the following frames is then estimated by integrating IMU measurements.

4.2 Back-end

In this section, we describe how we fuse feature tracks from the event stream obtained in Section 4.1.2 to update the full sensor state over time.

As opposed to the EKF-based filtering employed in [28], we prefer to rely on a full smoothing approach based nonlinear optimization on selected keyframes. Indeed, such approaches have been shown to outperform pipelines in terms of accuracy [24]. This has recently been made computationally tractable by the development of the pre-integration theory [17], [7], that consists of combining many inertial measurements between two keyframes into a single relative motion constraint, thus avoiding to reintegrate inertial measurements in each step of the optimization.We base our back-end implementation on OKVIS [15]. For space reasons, we omit the details of the pipeline and refer the reader to the original publications [14], [15].

The visual-inertial localization and mapping problem is formulated as a joint optimization of a cost function that contains weighted reprojection errors $\mathbf{e}_r$ and inertial error terms $\mathbf{e}_s$:

$$J := \sum_{k=1}^{K} \sum_{j \in \mathcal{J}(k)} {\mathbf{e}^{j,k}}^T \mathbf{W}_r^{j,k} \mathbf{e}^{j,k} + \sum_{k=1}^{K-1} {\mathbf{e}_s^k}^T \mathbf{W}_s^k \mathbf{e}_s^k$$

where k denotes the frame index, and j denotes the landmark index. The set $\mathcal{J}(k)$ contains the indices of landmarks visible in the k^{th} frame. Additionally, $W_r^{j,k}$ is the information matrix of the landmark measurement $\mathbf{l}_j$, and W_s^k that of the k^{th} IMU error. The optimization is carried out, not on all the frames observed, but on a bounded set of frames composed of M keyframes (selected by the front-end, see Section 4.1.3), and a sliding window containing the last K frames. In between frames, the prediction for the sensor state is propagated using the IMU measurements that fall in between the frames. We employ the Google Ceres [1] optimizer to carry out the optimization.

Reprojection Error. $\mathbf{e}_r^{j,k} = \mathbf{z}^{j,k} - \pi\left(\mathbf{T}_{CS}^k \mathbf{T}_{SW}^k \mathbf{l}^j\right)$ where $\mathbf{z}^{j,k}$ is the measured image coordinate of the j^{th} landmark on the k^{th} frame.

IMU Measurement Error Term. We use the IMU kinematics and biases model introduced in (2) to predict the current state based on the previous state. Then, the IMU error terms are simply computed as the difference between the prediction based on the previous state and the actual state. For orientation, a simple multiplicative minimal error is used.

Keyframe Marginalization. Keeping all keyframes in the Gauss-Newton system state quickly becomes untractable. However, simply discarding measurements from past keyframes neglects valuable information. To overcome this problem we partially marginalize out old keyframes using the Schur complement on the corresponding observations. This turns old measurements into a prior for our system, represented as summands in the construction of the Schur complement. Details are provided in [15].

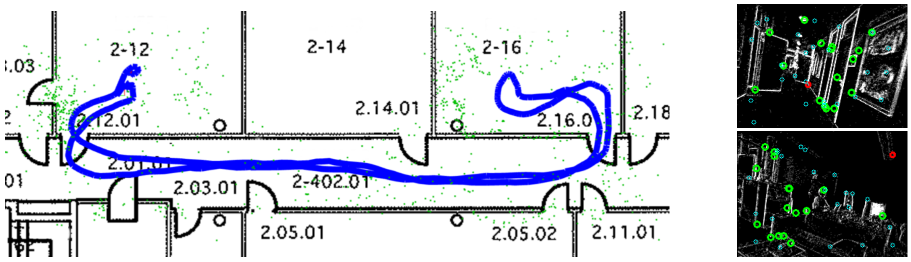


Figure 5: `corridor_dataset`. Left: trajectory and point cloud estimated by our pipeline, overlaid with a map of the building. Right: Two motion-corrected event frames used by our pipeline, with overlaid persistent landmarks (green) and candidate landmarks (blue). Red circles are RANSAC outliers.

5 Evaluation

For all the experiments presented below, we used the DAVIS [3] sensor, which embeds a 240×180 pixels event camera with a 1 kHz IMU. In addition to the event stream and IMU measurements, the sensor provides standard images, which are *not* used by our pipeline.

5.1 Quantitative: Accuracy and Performance

We use the Event Camera Dataset [19] to evaluate quantitatively the accuracy of our pipeline. The dataset features various scenes with ground truth tracking information. In particular, it contains extremely fast motions and scenes with very high-dynamic range.

Evaluation Metrics. The estimated and ground truth trajectories were aligned with a 6-DOF transformation (e.g. in SE3), using the subset $[5-10s]$. We computed the mean position error (Euclidean distance) and the yaw error as percentages of the total travelled distance. Due to the observability of the gravity direction, the error in pitch and roll direction is constant and comparable between our approach and EVIO. Thus we omit them for compactness. Additionally, we use the relative error metrics proposed in [10], which evaluate the relative error by averaging the drift over trajectories of different lengths (Fig. 6).

Parameters. The window size N was manually selected for all datasets, always in the range of $[10^4 - 10^5]$ events, except for the `shapes_translation` and `shapes_6dof` for which we used $N = 3000$ (since the global event rate is much lower in those). This translates to a temporal window size of about 5 to 10 milliseconds. We used $S = N$ for all the experiments, e.g., no overlap between successive windows. The patch size used for feature tracking was 32×32 pixels, with 2 pyramid levels.

Accuracy and Performance. Table 1 and Fig. 6 demonstrate the remarkable accuracy of our pipeline compared to EVIO [28], the state-of-the-art. We ran the same evaluation code (alignment and computation of error metrics) for our method and EVIO, using raw trajectories provided by the authors.

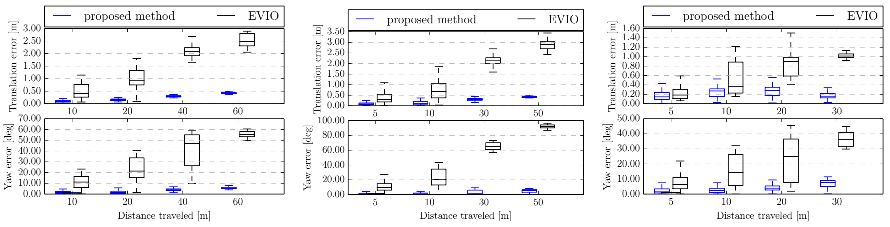
Our method runs on average 50% faster than real-time on a laptop, even for fast motions that yield very high event rates. For example the `boxes_6dof` dataset was processed in

Sequence	Our proposed method		EVIO [28] (CVPR'17)	
	Mean Position* Error (%)	Mean Yaw Error (deg/m)	Mean Position Error (%)	Mean Yaw Error (deg/m)
boxes_6dof	**0.69**	**0.09**	4.13	0.92
boxes_translation	**0.57**	**0.04**	3.18	0.67
dynamic_6dof	**0.54**	**0.26**	3.38	1.20
dynamic_translation	**0.47**	**0.11**	1.06	0.25
hdr_boxes	**0.92**	**0.01**	3.22	0.15
hdr_poster	**0.59**	**0.09**	1.41	0.13
poster_6dof	**0.82**	**0.11**	5.79	1.84
poster_translation	**0.89**	**0.03**	1.59	0.38
shapes_6dof	**1.15**	**0.08**	2.52	0.61
shapes_translation	**1.28**	**0.41**	4.56	2.60

Table 1: Accuracy of the proposed approach against EVIO [28], the state-of-the-art.

	Time (ms)
synthesize event frame	4.23
feature detection	0.69
feature tracking	0.90
two-point RANSAC	0.08
add frame to back-end	1.47
wait for back-end	1.29
total time	**8.23**

Table 2: Time spent in different modules, for a single spatio-temporal window.



(a) `boxes_6dof` (b) `poster_6dof` (c) `dynamic_6dof`

Figure 6: Comparison of the proposed approach versus EVIO on three datasets from [19]. Relative errors are measured over different segments of the trajectory. Additional plots for all the datasets are provided in the supplementary material.

41.7 s, which corresponds to 3.2 million events/s or 1.45 times faster than real-time. Table 2 shows the time spent per spatio-temporal window on an Intel Core i5-4278U@2.60GHz. The total time per event frame is 8.23 ms. We also run our algorithm on a smartphone CPU (Odroid-XU4@2GHz) and we measured a total time of 20ms.

5.2 Qualitative Results

To further demonstrate the capabilities of our method, we present two additional datasets: `corridor`, and `spinning_leash`. The `corridor` dataset was recorded by walking in our building with a DAVIS sensor, bringing it back to its exact start position. Fig. 5 shows a top-view of the estimated trajectory and the accumulated landmarks. In the absence of ground truth, we estimate the accumulated drift as the distance between the first position and the last position: about 50 cm for a 55 m trajectory, i.e. less than 1 % drift.

The `spinning_leash` dataset was recorded by spinning really fast a DAVIS camera, attached to a leash, in our office (Fig. 7). Despite the extreme velocity of the motion, our pipeline successfully tracks the camera pose with low drift.

5.3 Discussion

Our outstanding results compared to [28] can be explained in part because we use a nonlinear optimization approach, as opposed to a filtering approach, whose accuracy is known to

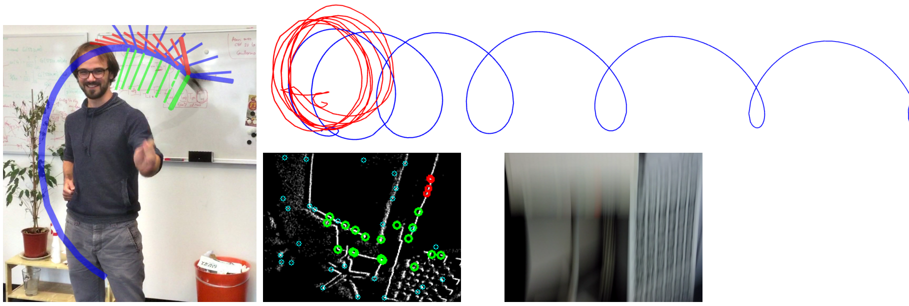


Figure 7: `spinning_leash` dataset. Left: Person spinning an event camera attached to a leash. The camera is barely visible due to motion blur. The trajectory estimated in real-time by our algorithm is superimposed on the image. Top-right: Trajectory estimated by our method (red) and plain IMU integration (blue). Bottom-right: motion-corrected event frame (same legend as Fig. 5) compared to an image obtained by spinning a standard camera at the same speed.

quickly deteriorate due to the accumulation of linearization errors.

Due to its simplicity, our feature tracker can be implemented efficiently, making real-time tracking possible on the CPU. By contrast, the feature tracker used in [28] relies on an expensive, iterative Expectation-Maximization scheme which severely throttles the speed of the overall pipeline.

6 Conclusion

We presented a novel, tightly-coupled visual-inertial odometry pipeline for event cameras. Our method significantly outperforms the state-of-the-art [28] on the challenging Event Camera Dataset [19], while being computationally more efficient; it can run on average 50 % faster than real-time on a laptop. We also demonstrated qualitatively the accuracy and robustness of our pipeline on a large-scale dataset, and an extremely high-speed dataset. We believe this work makes a significant step towards the use of event cameras for high-impact applications, such as the navigation of mobile robots, or AR/VR applications.

7 Acknowledgments

We thank Guillermo Gallego for valuable discussions and suggestions, and the authors of [28] for sharing the raw trajectories of their algorithm on the Event Camera Dataset.

This research was funded by the DARPA FLA Program, the National Center of Competence in Research (NCCR) Robotics through the Swiss National Science Foundation and the SNSF-ERC Starting Grant.

References

[1] A. Agarwal, K. Mierle, and Others. Ceres solver. `http://ceres-solver.org`.

[2] S. Baker and I. Matthews. Lucas-kanade 20 years on: A unifying framework. *Int. J. Comput. Vis.*, 56(3):221–255, 2004.

[3] Christian Brandli, Raphael Berner, Minhao Yang, Shih-Chii Liu, and Tobi Delbruck. A 240x180 130dB 3us latency global shutter spatiotemporal vision sensor. *IEEE J. Solid-State Circuits*, 49(10):2333–2341, 2014. ISSN 0018-9200. doi: 10.1109/JSSC.2014.2342715.

[4] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. D. Reid, and J. J. Leonard. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Trans. Robot.*, 32(6):1309–1332, 2016.

[5] Matthew Cook, Luca Gugelmann, Florian Jug, Christoph Krautz, and Angelika Steger. Interacting maps for fast visual interpretation. In *Int. Joint Conf. Neural Netw. (IJCNN)*, pages 770–776, 2011. doi: 10.1109/IJCNN.2011.6033299.

[6] Christian Forster, Luca Carlone, Frank Dellaert, and Davide Scaramuzza. IMU preintegration on manifold for efficient visual-inertial maximum-a-posteriori estimation. In *Robotics: Science and Systems (RSS)*, 2015. doi: 10.15607/RSS.2015.XI.006.

[7] Christian Forster, Luca Carlone, Frank Dellaert, and Davide Scaramuzza. On-manifold preintegration for real-time visual-inertial odometry. *IEEE Trans. Robot.*, 33(1):1–21, 2017. doi: 10.1109/TRO.2016.2597321.

[8] P. Furgale, J. Rehder, and R. Siegwart. Unified temporal and spatial calibration for multi-sensor systems. In *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*, 2013.

[9] Guillermo Gallego and Davide Scaramuzza. Accurate angular velocity estimation with an event camera. *IEEE Robot. Autom. Lett.*, 2:632–639, 2017. ISSN 2377-3766. doi: 10.1109/LRA.2016.2647639.

[10] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Proc. IEEE Int. Conf. Comput. Vis. Pattern Recog.*, 2012.

[11] Hanme Kim, Ankur Handa, Ryad Benosman, Sio-Hoi Ieng, and Andrew J. Davison. Simultaneous mosaicing and tracking with an event camera. In *British Machine Vis. Conf. (BMVC)*, 2014. doi: 10.5244/C.28.26.

[12] 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. (ECCV)*, pages 349–364, 2016. doi: 10.1007/978-3-319-46466-4_21.

[13] Beat Kueng, Elias Mueggler, Guillermo Gallego, and Davide Scaramuzza. Low-latency visual odometry using event-based feature tracks. In *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*, pages 16–23, Daejeon, Korea, October 2016. doi: 10.1109/IROS.2016.7758089.

[14] S. Leutenegger, P. Furgale, V. Rabaud, M. Chli, K. Konolige, and R. Siegwart. Keyframe-based visual-inertial SLAM using nonlinear optimization. In *Robotics: Science and Systems (RSS)*, 2013.

[15] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale. Keyframe-based visual-inertial SLAM using nonlinear optimization. *Int. J. Robot. Research*, 2015.

[16] Patrick Lichtsteiner, Christoph Posch, and Tobi Delbruck. A 128×128 120 dB 15 μs latency asynchronous temporal contrast vision sensor. *IEEE J. Solid-State Circuits*, 43 (2):566–576, 2008. doi: 10.1109/JSSC.2007.914337.

[17] T. Lupton and S. Sukkarieh. Visual-inertial-aided navigation for high-dynamic motion in built environments without initial conditions. *IEEE Trans. Robot.*, 28(1):61–76, February 2012.

[18] Elias Mueggler, Guillermo Gallego, Henri Rebecq, and Davide Scaramuzza. Continuous-time visual-inertial trajectory estimation with event cameras. `arXiv:1702.07389`, 2017.

[19] 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. *Int. J. Robot. Research*, 36:142–149, 2017. doi: 10.1177/0278364917691115.

[20] Henri Rebecq, Guillermo Gallego, and Davide Scaramuzza. EMVS: Event-based multi-view stereo. In *British Machine Vis. Conf. (BMVC)*, September 2016.

[21] Henri Rebecq, Timo Horstschäfer, Guillermo Gallego, and Davide Scaramuzza. EVO: A geometric approach to event-based 6-DOF parallel tracking and mapping in real-time. *IEEE Robot. Autom. Lett.*, 2:593–600, 2017. ISSN 2377-3766. doi: 10.1109/LRA.2016.2645143.

[22] C. Reinbacher, G. Munda, and T. Pock. Real-time panoramic tracking for event cameras. In *IEEE Int. Conf. Computational Photography (ICCP)*, 2017.

[23] E. Rosten and T. Drummond. Machine learning for high-speed corner detection. In *Eur. Conf. Comput. Vis. (ECCV)*, pages 430–443, 2006. doi: 10.1007/11744023_34.

[24] H. Strasdat, J.M.M. Montiel, and A.J. Davison. Real-time monocular SLAM: Why filter? In *IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2010.

[25] C. Troiani, A. Martinelli, C. Laugier, and D. Scaramuzza. 2-point-based outlier rejection for camera-imu systems with applications to micro aerial vehicles. In *IEEE Int. Conf. Robot. Autom. (ICRA)*, 2014.

[26] David Weikersdorfer, Raoul Hoffmann, and Jörg Conradt. Simultaneous localization and mapping for event-based vision systems. In *Int. Conf. Comput. Vis. Syst. (ICVS)*, pages 133–142, 2013. doi: 10.1007/978-3-642-39402-7_14.

[27] David Weikersdorfer, David B. Adrian, Daniel Cremers, and Jörg Conradt. Event-based 3D SLAM with a depth-augmented dynamic vision sensor. In *IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 359–364, June 2014. doi: 10.1109/ICRA.2014.6906882.

[28] A. Zhu, N. Atanasov, and K. Daniilidis. Event-based visual inertial odometry. In *Proc. IEEE Int. Conf. Comput. Vis. Pattern Recog.*, 2017.