Continuous-Time Trajectory Estimation for Event-based Vision Sensors

Elias Mueggler, Guillermo Gallego and Davide Scaramuzza
Robotics and Perception Group, University of Zurich
{mueggler,guillermo.gallego,sdavide}@ifi.uzh.ch

***Abstract*—Event-based vision sensors, such as the Dynamic Vision Sensor (DVS), do not output a sequence of video frames like standard cameras, but a stream of asynchronous events. An event is triggered when a pixel detects a change of brightness in the scene. An event contains the location, sign, and precise timestamp of the change. The high dynamic range and temporal resolution of the DVS, which is in the order of *micro*-seconds, make this a very promising sensor for high-speed applications, such as robotics and wearable computing. However, due to the fundamentally different structure of the sensor's output, new algorithms that exploit the high temporal resolution and the asynchronous nature of the sensor are required. In this paper, we address ego-motion estimation for an event-based vision sensor using a continuous-time framework to directly integrate the information conveyed by the sensor. The DVS pose trajectory is approximated by a smooth curve in the space of rigid-body motions using cubic splines and it is optimized according to the observed events. We evaluate our method using datasets acquired from sensor-in-the-loop simulations and onboard a quadrotor performing flips. The results are compared to the ground truth, showing the good performance of the proposed technique.**

I. Introduction

Standard frame-based CMOS cameras operate at fixed frame rates, sending entire images at constant time intervals that are selected based on the considered application [14]. Contrary to standard cameras, where pixels are acquired at regular time intervals (e.g., global shutter or rolling shutter), event-based vision sensors, such as the Dynamic Vision Sensor (DVS) [19], have asynchronous pixels. Each pixel of the DVS immediately triggers an *event* in case of changing brightness. The temporal resolution of such events is in the order of *micro*-seconds. It is only these changes that are transmitted, and, consequently, the DVS is also referred to as an event-based vision sensor. Since the output it produces—an event stream—is fundamentally different from video streams of standard CMOS cameras, new algorithms are required to deal with this data. Event-based adaptations of iterative closest points [24] and optical flow [5] have already been proposed. Recently, event-based visual odometry [9, 17], tracking [28, 23], and Simultaneous Localization And Mapping (SLAM) [29] algorithms have also been presented. The design goal of such algorithms is that each incoming event can asynchronously change the estimated state of the system, thus, preserving the event-based nature of the sensor and allowing the design of highly-reactive systems, such as pen balancing [10] or particle tracking in fluids [12].

We aim to use the DVS for ego-motion estimation. The approach provided by traditional visual-odometry frameworks, which estimate the camera pose at discrete times (naturally, the times the images are acquired), is no longer appropriate for event-based vision sensors, mainly due to two issues. First, a single event does not contain enough information to estimate the sensor pose given by the six degrees of freedom (DOF) of a calibrated camera. We cannot simply consider several events to determine the pose using standard computer-vision techniques (e.g., [18]), because the events typically all have different timestamps, and so the resulting pose will not correspond to any particular time. Second, a DVS typically transmits 10^5 events per second, and so it is intractable to estimate the DVS pose at the discrete times of all events due to the rapidly growing size of the state vector needed to represent all such poses. Instead, we adopt a continuous-time framework approach [7] that solves the previous issues and has additional advantages. Regarding the first issue, an explicit continuous temporal model is a natural representation of the pose trajectory $\mathtt{T}(t)$ of the DVS since it unambiguously relates each event, occurring at time t_k, with its corresponding DVS pose, $\mathtt{T}(t_k)$. To solve the second issue, the DVS trajectory is described by a smooth parametric model, with significantly fewer parameters than events, hence achieving state space size reduction and computational efficiency. For example, to remove unnecessary states for the estimation of the trajectory of dynamic objects, [7] proposed to use cubic splines. In the experiments, they could on average reduce the size of the state space by 70-90%. Cubic splines are also used in [13] to model continuous-time trajectories. However, instead of introducing all states and removing them a posteriori, the estimation problem was directly stated in continuous time. Wavelets have also been considered as basis for continuous-time trajectories [3]. The continuous-time framework was also motivated to allow data fusion of multiple sensors working at different rates and to enable increased temporal resolution [7]. For example, [13] adopted it for dealing with two unsynchronized sensors, such as vision and inertial ones. In the same context of visual-inertial fusion, the framework was used by [21] to take into account the different timestamps of the lines of the images acquired by rolling-shutter cameras. Similarly, the continuous-time framework has been applied to actuated lidar [2]. Recently, connections between continuous-time trajectory estimation and Gaussian process regression have been shown both in batch [4] and incremental forms [31].

A. Contribution

To the authors' knowledge, this work is the first one where the continuous-time framework is utilized to represent the trajectory of event-based vision sensors (e.g., the DVS), which are asynchronous by design. Previous works used sensors of different rates, but these were not asynchronous by design, but rather *unsynchronized* with respect to each other. Since events occur at high frequency and do not carry enough information to estimate the full pose of the DVS, this formulation allows us to naturally incorporate all of the information they contain while limiting the size of the state space. It especially allows us to exploit the high temporal resolution of the DVS and enables us to compute the pose at any point in time along the trajectory, with no additional sensing. We describe the DVS pose trajectory using cubic B-splines as temporal basis functions and we estimate it as a *whole* in a principled optimization approach, as opposed to being estimated from individually-optimized poses. We minimize a geometrically meaningful measure in the image plane of the DVS with respect to the parameters describing the trajectory. The experiments show the good performance of the proposed approach. The method has been tested on datasets acquired from a quadrotor performing flips.

The remainder of the paper is organized as follows. In Section II, we characterize the Dynamic Vision Sensor (DVS). In Section III, we review previous work on ego-motion methods with event-based vision sensors. The method developed for DVS trajectory estimation is described in Section IV and evaluated in Section V. Section VI draws conclusions and points out future work.

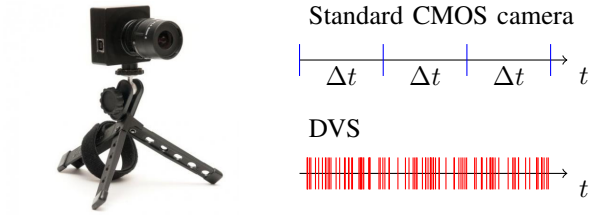
II. Dynamic Vision Sensor (DVS)

Standard CMOS cameras send full frames at fixed frame rates. On the other hand, event-based vision sensors such as the DVS (Fig. 1(a)) have independent pixels that fire events at local relative brightness changes in continuous time. Specifically, if $I(x, y)$ is the brightness or intensity at point $\mathbf{u} = (x, y)^\top$ in the image plane, the DVS generates an event at that location if the change in logarithmic brightness is greater than a threshold [9] (typically 10-15% relative brightness change),

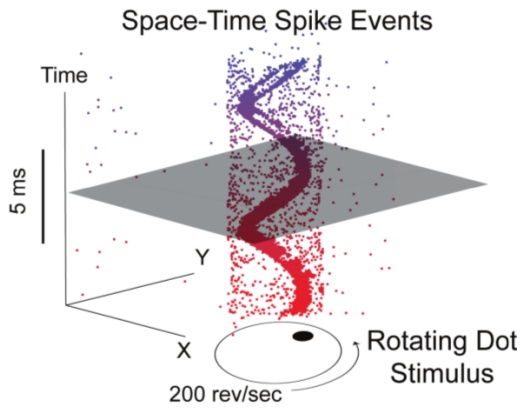
$$|\Delta \log(I)| \approx |-\langle \nabla \log(I), \dot{\mathbf{u}} \Delta t \rangle| > C, \qquad (1)$$

where ∇ computes the gradient (with respect to spatial coordinates), $\dot{\mathbf{u}}$ is the *image motion field* [27, p. 183], $\langle \cdot, \cdot \rangle$ is the dot product, and Δt is the time since the previous event at the same pixel location.

These events are timestamped and transmitted asynchronously at the time they occur using a sophisticated digital circuitry. Each event is a tuple $e_k = \langle x_k, y_k, t_k, p_k \rangle$, where x_k, y_k are the pixel coordinates of the event, t_k is the timestamp of the event, and $p_k \in \{-1, +1\}$ is the polarity of the event, which is the sign of the brightness change. This representation is sometimes also referred to as Address-Events Representation [19]. The set of all events is denoted as $\mathcal{E} = \{e_k\}, k = 1, \ldots, n_\mathcal{E}$, where $n_\mathcal{E}$ is the total number of events.



(a) Left: The Dynamic Vision Sensor. Right: A standard CMOS camera sends images at a fixed frame rate (blue). A DVS instead sends spike events at the time they occur (red). Each event corresponds to a local, pixel-level change of brightness.



(b) Visualization of the output of a DVS looking at a rotating dot. Colored dots mark individual events. The polarity of the events is not shown. Events that are not part of the spiral are caused by sensor noise. Figure adapted from [20].

Fig. 1. The output of a Dynamic Vision Sensor (DVS).

The DVS has the same optics as traditional perspective cameras, therefore, standard camera models (e.g., pinhole) still apply. The sensor's spatial resolution is 128×128 pixels and it is connected via USB. A visualization of the output of the DVS is shown in Fig. 1(b). An additional advantage of the DVS is its high dynamic range of 120 dB (compared to 60 dB of high quality traditional image sensors). Current research efforts [8] are being carried towards increasing the spatial resolution of the sensor as well as offering the possibility to return the absolute pixel brightness (at standard frame rates) in addition to the events.

III. Related Work: Ego-Motion Estimation with Event-Based Vision Sensors

A particle-filter approach for robot self-localization using the DVS was introduced in [28] and later extended to SLAM in [29]. However, the system was limited to planar motions and 2-D maps. In the experiments, they used an upward-looking DVS mounted on a ground robot moving at low speed. An externally provided map consisting of line segments on the ceiling was used for navigation.

In several ego-motion–estimation applications, the DVS has been used in combination with other vision sensors. For example, in [9], an event-based pipeline for visual odometry with the DVS and a regular (CMOS) camera was demonstrated.

They used a probabilistic framework that processes the events from the DVS to update the relative pose displacement of a mobile platform since the time of the previous CMOS frame. As another example, in the context of SLAM, the DVS was combined with a frame-based RGB-D camera in [30]. The algorithm used a modified particle filter for tracking the current position and orientation of the sensor while at the same time incrementally creating a probabilistic voxel grid map of the previously unknown environment.

Simultaneous mosaicing and tracking with the DVS was presented in [17]. In that approach, pose tracking was limited to 3-D rotations and they were able to reconstruct super-resolution panoramic image mosaics (in absolute grayscale) from estimated brightness gradients. Their probabilistic filtering algorithm was operating on an event-by-event basis. They used a SLAM-like method of two parallel (Bayesian) filters to jointly estimate the camera's rotational motion and a gradient map of a scene.

In our previous work [23], we demonstrated robot localization in 3-D (with arbitrary 6-DOF motions) using a DVS, with no additional sensing, during high-speed maneuvers, where rotational speeds of up to $1{,}200\,^\circ$/s were measured during quadrotor flips. The focus was to enable a perception pipeline whose latency is negligible compared to the dynamics of the robot. This was done by tracking a set of gradients on a given map on an event-by-event basis, minimizing the reprojection error.

None of these reviewed ego-motion references has a continuous-time representation of the trajectory of the DVS, which is the approach leveraged in this paper and introduced in the next section.

IV. Continuous-Time Trajectories

Traditional visual odometry and Simultaneous Localization and Mapping (SLAM) formulations use a discrete-time approach, i.e., the camera pose is calculated at the time the image was acquired. Recent works have shown that, for high-frequency data, a continuous-time formulation is preferable to keep the size of the optimization problem bounded [13, 21]. Temporal basis functions, such as B-splines, were proposed for camera-IMU calibration, where the frequencies of the two sensor modalities differ by an order of magnitude. While previous approaches use continuous-time representations mainly to reduce the computational complexity, in the case of an event-based sensor this representation is required to cope with the asynchronous nature of the events. Unlike a standard camera image or an IMU reading, an event does not carry enough information to estimate the sensor pose by itself. A continuous-time trajectory can be evaluated at any time, in particular at each event's timestamp, yielding a well-defined pose for every event. Thus, our method is not only computationally effective, but it is also necessary for a proper formulation.

Following [21], we represent Euclidean space transformations between finite cameras [15, p. 157] by means of 4×4 matrices of the form

$$\mathtt{T}_{b,a} = \begin{bmatrix} \mathtt{R}_{b,a} & \mathbf{t}_a \\ \mathbf{0}^\top & 1 \end{bmatrix}, \tag{2}$$

where $\mathtt{R} \in SO(3)$ (the rotation group) and $\mathbf{t} \in \mathbb{R}^3$ are the rotational and translational components of the rigid-body motion, respectively. In homogeneous coordinates, a 3-D point in frame a is mapped to a point in frame b by the change of coordinates $\mathbf{X}_b \sim \mathtt{T}_{b,a}\mathbf{X}_a$, where $\sim$ means equality up to a non-zero scale factor. Transformations (2) form the special Euclidean group $SE(3)$ [22, p. 30], which has the structure of both a group and a differentiable manifold, i.e., a Lie group. A curve on $SE(3)$ physically represents the motion of a rigid body, e.g., the DVS. The tangent space of $SE(3)$ at the identity is $se(3)$, which has the structure of a Lie algebra. It corresponds to the space of *twists*, represented by 4×4 matrices of the form

$$\widehat{\boldsymbol{\xi}} = \begin{bmatrix} \widehat{\boldsymbol{\omega}} & \mathbf{v} \\ \mathbf{0}^\top & 0 \end{bmatrix}, \tag{3}$$

where $\mathbf{v} \in \mathbb{R}^3$ and $\widehat{\boldsymbol{\omega}}$ is the 3×3 skew-symmetric matrix representing the cross product: $\widehat{\boldsymbol{\omega}}\mathbf{b} = \boldsymbol{\omega} \times \mathbf{b},\ \forall \boldsymbol{\omega}, \mathbf{b} \in \mathbb{R}^3$. Variables $\boldsymbol{\omega}$ and $\mathbf{v}$ physically represent the angular and linear velocity vectors of the moving DVS.

Based on the theory of Lie groups, the exponential map from $se(3)$ to $SE(3)$ can be defined, which gives the Euclidean transformation associated to a twist, $\mathtt{T} = \exp(\widehat{\boldsymbol{\xi}})$. The inverse of the exponential map is the logarithmic map $\widehat{\boldsymbol{\xi}} = \log(\mathtt{T})$. Moreover, every rigid-body motion $\mathtt{T} \in SE(3)$ can be represented in such an exponential parametrization, but the resulting twist may not be unique [22, p. 33]. However, to avoid this ambiguity, we adopt a local-chart approach (on the manifold $SE(3)$) by means of incremental rigid-body motions ($\mathtt{T} = \exp(\widehat{\boldsymbol{\xi}})$ with small matrix norm $\|\widehat{\boldsymbol{\xi}}\|$) given by the relative transformation between two nearby poses along the trajectory of the DVS (see (5)). In addition, this parametrization is free from singularities. Closed-form formulas for the the $\exp$ and $\log$ maps are given in [22].

A. Cumulative B-Splines

Following the approaches in [13, 21], we use the continuous trajectory representation given by cubic splines since they are characterized by valuable properties: (i) local dependency of the trajectory with respect to the control points defining it, (ii) simple analytical derivatives and integrals, and (iii) the possibility of having C^2 continuity. To this end, we adopt cumulative B-spline basis functions formed using the Lie algebra [11], which produce smooth trajectories in the manifold of rigid-body motions $SE(3)$.

The continuous trajectory of the DVS is parametrized by control camera poses $\mathtt{T}_{w,i}$ at times t_i, $i \in \{0, \ldots, n\}$, where, following the sub-index notation in (2), $\mathtt{T}_{w,i}$ is the transformation from the DVS frame at time t_i to a world frame (w). We assume that the control poses are uniformly distributed in time, in intervals of size Δt. Due to the locality of the cubic B-spline basis, the value of the spline curve at any time t only

depends on four control poses. Specifically, for $t \in [t_i, t_{i+1})$ such control poses occur at times $\{t_{i-1}, \ldots, t_{i+2}\}$, and we use one absolute pose (in the world frame), $\mathtt{T}_{w,i-1}$, and three incremental poses, parameterized by twists (3) $\widehat{\boldsymbol{\xi}}_q \equiv \Omega_q$, according to the mentioned local approach on $SE(3)$.

The pose in the spline trajectory at time $t \in [t_i, t_{i+1})$ is

$$\mathtt{T}_{w,s}\big(u(t)\big) = \mathtt{T}_{w,i-1} \prod_{j=1}^{3} \exp\left(\tilde{\mathbf{B}}_j(u(t))\Omega_{i+j-1}\right), \quad (4)$$

where $u(t) = (t - t_i)/\Delta t \in [0, 1)$, the incremental pose from frame at time t_{q-1} to frame at time t_q is encoded in the twist

$$\Omega_q = \log(\mathtt{T}_{w,q-1}^{-1}\mathtt{T}_{w,q}) \quad (5)$$

in terms of world-referenced poses, and

$$\tilde{\mathbf{B}}(u) = \frac{1}{6}\begin{bmatrix} 6 & 0 & 0 & 0 \\ 5 & 3 & -3 & 1 \\ 1 & 3 & 3 & -2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ u \\ u^2 \\ u^3 \end{bmatrix} \quad (6)$$

are the cumulative basis functions for the B-splines, derived from the matrix representation of the De Boor-Cox formula [25]. $\tilde{\mathbf{B}}_j$ is the j-th entry (0 based) of the cubic polynomial vector.

B. Map Representation

To focus on the DVS trajectory estimation problem, we assume that the map of the scene is given and is time invariant. Specifically, the map $\mathcal{M}$ is a set of 3-D line segments,

$$\mathcal{M} = \{\ell_j\}. \quad (7)$$

Line segments ℓ_j may be parametrized in different ways, for example by their start and end points $\mathbf{X}_j^s, \mathbf{X}_j^e \in \mathbb{R}^3$. As it will be shown, the objective function (15) measures point-to-line distances in the image plane, so we may relax the requirement of precisely known endpoints of the segments by considering alternative parametrizations, such as Plücker coordinates of 3-D lines and rough estimates of the segments lengths. The solution of the data association sub-problem (section IV-C2) between events and line segments also confers robustness to our method, which further supports the relaxation of the above representation.

Given a 3×4 projection matrix $\mathtt{P}$ modeling the perspective projection carried out by the DVS, the lines of the map $\mathcal{M}$ can be projected to the image plane by using Plücker coordinates [26] or by projecting the endpoints of the segments (if they are available) and computing the line through them. The homogeneous coordinates of the projected line through the j-th segment are, respectively,

$$l_j \sim \mathcal{P}\Omega\boldsymbol{\ell}_j, \quad (8)$$

where ℓ_j are the Plücker coordinates of the j-th 3-D line, Ω is the Klein quadric [15, p.72], and $\mathcal{P}$ is the line projection matrix (obtained from $\mathtt{P}$), or

$$l_j \sim (\mathtt{P}\mathbf{X}_j^s) \times (\mathtt{P}\mathbf{X}_j^e). \quad (9)$$

C. DVS Trajectory Estimation

1) Probabilistic Approach: In general, the trajectory estimation problem over an interval $[0, T]$ can be cast in a probabilistic form [13], seeking an estimate of the joint posterior density $p(\mathbf{x}(t)|\mathcal{M}, \mathbf{z}_{1:N})$ of the DVS state $\mathbf{x}(t)$ (pose trajectory) over the interval, given the map $\mathcal{M}$ and the set of all measured events $\mathbf{z}_{1:N} = \{\mathbf{z}_1, \ldots, \mathbf{z}_N\}$, $\mathbf{z}_k = (x_k, y_k)^\top$ being the measured event location at time t_k. Using Bayes' rule, and assuming that the map is independent of the DVS trajectory, we may rewrite the posterior as

$$p(\mathbf{x}(t)|\mathcal{M}, \mathbf{z}_{1:N}) \propto p(\mathbf{x}(t))\, p(\mathbf{z}_{1:N}|\mathbf{x}(t), \mathcal{M}). \quad (10)$$

In the absence of prior belief for the state, $p(\mathbf{x}(t))$, the optimal trajectory is the one maximizing the likelihood $p(\mathbf{z}_{1:N}|\mathbf{x}(t), \mathcal{M})$. Under the standard assumptions that the measurements are independent of each other (given the trajectory and the map) and that the measurement error in the image coordinates of the events follows a Gaussian distribution, the logarithmic likelihood becomes

$$\log\left(p(\mathbf{z}_{1:N}|\mathbf{x}(t), \mathcal{M})\right) \quad (11)$$

$$= \log\left(\prod_k p(\mathbf{z}_k|\mathbf{x}(t_k), \mathcal{M})\right) \quad (12)$$

$$= \log\left(\prod_k K \exp\left(-\frac{\|\mathbf{z}_k - \hat{\mathbf{z}}_k(\mathbf{x}(t_k), \mathcal{M})\|^2}{2\sigma^2}\right)\right) \quad (13)$$

$$= \tilde{K} - \frac{1}{2\sigma^2}\sum_k \|\mathbf{z}_k - \hat{\mathbf{z}}_k(\mathbf{x}(t_k), \mathcal{M})\|^2 \quad (14)$$

where $K, \tilde{K}, \sigma^2$ are constants. Given the map (7), the predicted value of the event location $\hat{\mathbf{z}}_k(\mathbf{x}(t_k), \mathcal{M})$ is a point on the projection of one of the 3-D line segments ℓ_j, and so the reprojection error given by the norm in (14) becomes the Euclidean (perpendicular) distance from the point to a line segment, $d_\perp(\mathbf{z}, l)$. The maximization of the likelihood (11) becomes the minimization of the objective function given by the sum of squared distances in the image plane

$$f \coloneqq \sum_k d_\perp^2\big(\mathbf{z}_k, l_j(\mathbf{x}(t_k))\big), \quad (15)$$

where $l_j(\mathbf{x}(t_k))$ is the projection of the line segment $\ell_j \in \mathcal{M}$ according to the pose specified in the DVS trajectory at the time of the event, t_k. Of course, this implies that there is a data association sub-problem consisting of establishing correct correspondences between points and line segments.

2) Constrained Optimization in Finite Dimensions: The objective function (15) is optimized with respect to the trajectory $\mathbf{x}(t)$ of the DVS, which in general is represented by an arbitrary curve in $SE(3)$, i.e., a "point" in an infinite-dimensional function space. However, because we represent the curve in terms of a finite set of known temporal basis functions (B-splines, formalized in (4)), the trajectory is parametrized by control poses $\mathtt{T}_{w,i}$ and, therefore, the optimization problem becomes finite dimensional. In particular, it is a non-linear least squares (NLLS) problem, for which standard numerical

solvers such as Gauss-Newton or Levenberg-Marquardt can be applied.

Hence, we estimate the trajectory by minimizing the objective function (15) over the control poses,

$$\{\mathtt{T}^*_{w,i}\} = \arg\min_{\mathtt{T}} f. \tag{16}$$

For each event e_k, triggered at time t_k in the interval $[t_i, t_{i+1})$, we compute its pose $\mathtt{T}_{w,s}(u_k)$ using (4), where $u_k = (t_k - t_i)/\Delta t$. We then project each line segment into the current image plane and compute the distance between the event location $\mathbf{z}_k = (x_k, y_k)^\top$ and the corresponding imaged line segment l_j, which is computed using projection matrices $\mathtt{P}(t_k) \sim \mathtt{K}(\mathtt{I}|\mathbf{0})\mathtt{T}^{-1}_{w,s}(t_k)$, $\mathtt{K}$ being the time-invariant intrinsic parameter matrix of the DVS (once the radial distortion has been removed). To solve the data association sub-problem that establishes correspondences between events and line segments, we use an Iterative Closest Point technique [6]. The optimization problem (16) is then solved in an iterative way using the Ceres solver [1], an efficient numerical implementation for NLLS problems.

3) Trajectory Initialization and Extrapolation: We assume the first DVS pose to be known: considering that the map consists of line segments (7), the position of the DVS can be computed by integrating the events caused by the segments and using the Hough transform to detect the corresponding lines (see [23]). Due to the local convergence property of the iterative solving strategy, the control poses must be initialized in the basin of attraction of the optimal value. Since we cannot optimize the entire trajectory in one run without prior knowledge, we adopt a growing-window approach, optimizing over all existing control poses. We build up the trajectory by initializing the first four control poses to the known initial pose and run the optimization. We then use the last two control poses to extrapolate the new control pose that we add at the end. Then, we optimize again and repeat until the entire trajectory is approximated. Specifically, for a cubic spline trajectory, to evaluate the pose corresponding to an event with timestamp in the interval $L = [t_i, t_{i+1})$ we need the two control poses at t_{i+1} and t_{i+2} (see section IV-A). A new control pose at time t_{i+3} is extrapolated when the first event with a timestamp outside the interval L arrives. Extrapolation is performed assuming constant velocity: $T_{i+3} = T_{i+2}dT$ with $dT = T^{-1}_{i+1}T_{i+2}$. In the experiments we found that the influence of new events on the optimization of previous poses decays rapidly, hence if the number of poses becomes very large, one may switch to a sliding-window approach after building up an initial window.

V. Experiments

For the experiments, we used the datasets from [23] and compare the results of our continuous-time approach with those achieved by the event-based reprojection error minimization algorithm in [23]. The first experiment uses data captured in a sensor-in-the-loop simulation. The second experiment uses data captured by a DVS mounted on a quadrotor that was performing flips, reaching rotational speeds of $1{,}200\,°/\text{s}$. In both cases, the edges of a black square of known size on a white background constitute the map $\mathcal{M}$. The simplicity of the map facilitates the data association problem (see section IV-C2) and allows us to focus on the continuous-time trajectory estimation that naturally incorporates the asynchronous information of event-based cameras. These datasets fulfill all the requirements to test and quantify the results of our method because: (1) the DVS motion is in 6-DOF, (2) the apparent motion is large (this would cause significant motion blur in standard cameras, which would produce a breakdown of tracking algorithms), and (3) ground truth is available.

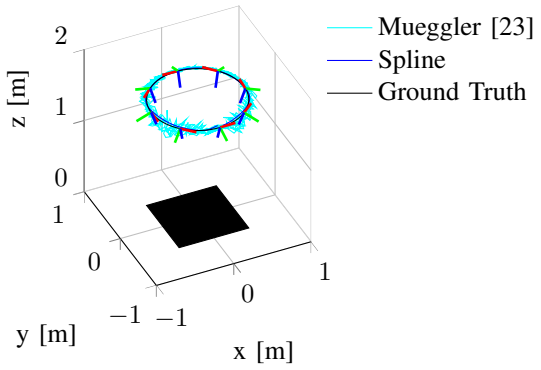


Fig. 2. Estimated trajectories for sensor-in-the-loop simulation with respect to the synthetic map (black square). The DVS only senses the apparent motion of the edges of this square.

A. Sensor-in-the-Loop Simulation

In the first experiment, a 3-D simulation on a computer screen was filmed by an actual DVS. The simulation shows a virtual flight in a circle over a black square (see Fig. 2), of which the DVS only senses the apparent motion of the edges. Since the DVS was calibrated with respect to the screen, the ground-truth trajectory is known. The trajectory is shown in Figs. 2 and 3. Control poses were placed every $0.1\,\text{s}$. The position and orientation errors are shown in Fig. 4, summarized in Table I, and compared to the results of [23]. Among the plots for the six degrees of freedom, those corresponding to x and y positions and roll angle are the most relevant. They reflect the DVS trajectory on a circle while always pointing at the center of the square (cf. Fig 2). The mean reprojection error was $0.49\,\text{pixels}$. Our results are consistently better than the ones achieved with the algorithm in [23].

To measure the error between an estimated orientation $\tilde{\mathtt{R}}$ and that of ground truth $\mathtt{R}_{gt}$, we use the angle θ of the relative rotation $\tilde{\mathtt{R}}\mathtt{R}^\top_{gt}$, computed as (cf. [15, p. 584])

$$\theta = \arccos\left((\text{trace}(\tilde{\mathtt{R}}\mathtt{R}^\top_{gt}) - 1)/2\right), \tag{17}$$

which is the geodesic distance in $SO(3)$ that comes naturally with the Lie group structure [16]. We compute the position and orientation errors for the pose of every event along the trajectory and then compute the statistics reported in Table I: mean (μ), standard deviation (σ), and root mean square (RMS).

The current implementation of our method is not optimized and, similarly to [13] and [21], not real-time. For example,

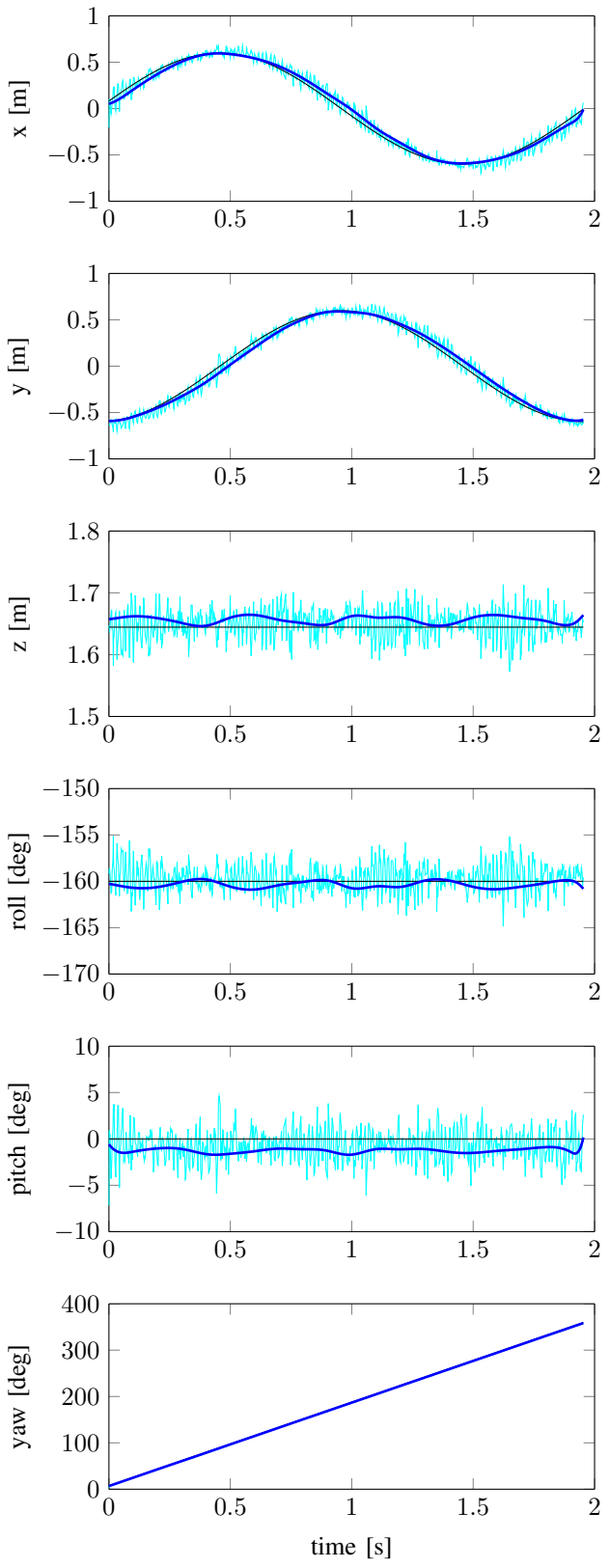


Fig. 3. Plots of the six degrees of freedom for the sensor-in-the-loop experiment showing results of the method in [23] (cyan), our method (blue), and ground truth (black).

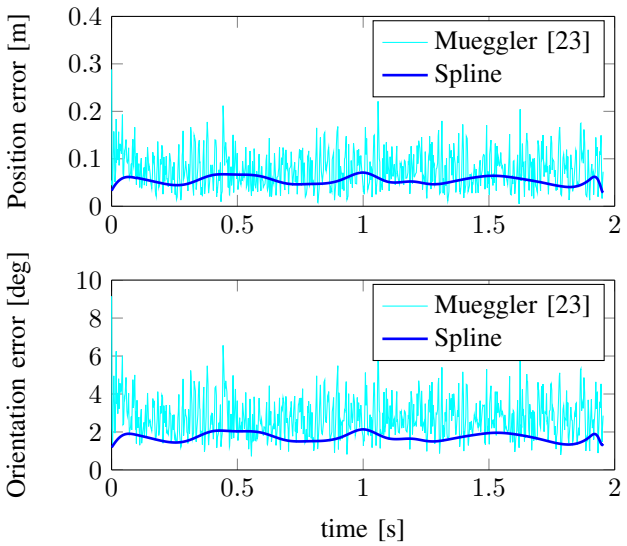


Fig. 4. Position and orientation errors for the sensor-in-the-loop experiment.

TABLE I
RESULTS OF THE SENSOR-IN-THE-LOOP EXPERIMENT.

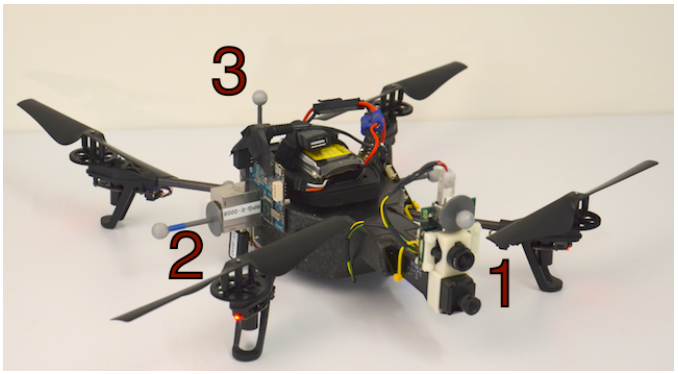
	Position error [cm]			Orientation error [°]		
	μ	σ	RMS	μ	σ	RMS
[23]	7.79	3.94	8.73	2.74	1.07	2.94
Spline	5.47	0.81	5.53	1.72	0.21	1.74

it takes 87.7 s to optimize a trajectory with 23 control poses from 40,518 events. However, our formulation results in an optimization problem with very few variables (i.e., the control poses) compared to the number of observations (i.e., the events) and, thus, is potentially real-time capable: possible optimizations include the computation of analytic derivatives (instead of numerical ones) and motion-dependent control-pose placement.

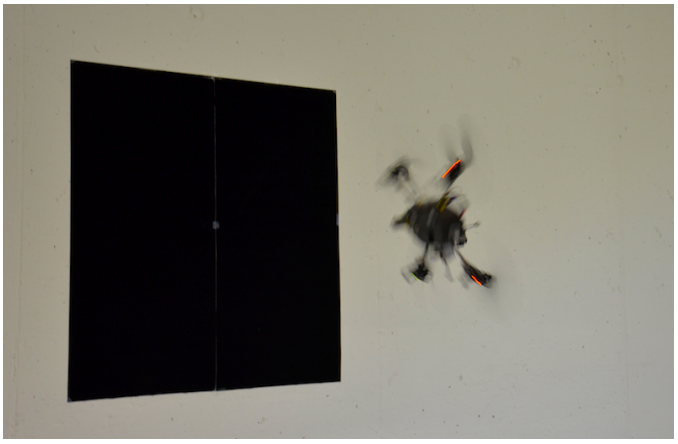
B. Quadrotor Experiment

The second experiment uses data provided by a DVS mounted on a quadrotor that performed flips around the optical axis of the DVS. The experimental setup is shown in Fig. 5. During high-speed maneuvers of mobile robots, images from standard cameras suffer from strong motion-blur effects (see Fig. 5(c)). However, the high temporal resolution of the DVS, which is in the order of *micro*-seconds, allows us to track such fast motions. In the present case, the rotational speed of the quadrotor reached 1,200 °/s. Fig. 5(d) shows all events in a time interval of 2 ms during a flip. The color (red or blue) corresponds to the sign of the brightness change.

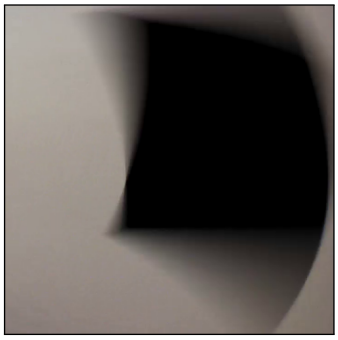
Fig. 6 shows the trajectories of our approach together with the estimated poses of [23] and the ground truth, which was measured with an OptiTrack motion capture system. Control poses were placed every 0.05 s. The errors for both algorithms are shown in Fig. 7 and analyzed in Table II. The most relevant plots of the six degrees of freedom are the height (z) and roll angle. The quadrotor accelerates upwards, performs the flip, and stabilizes as it goes down. The mean reprojection error



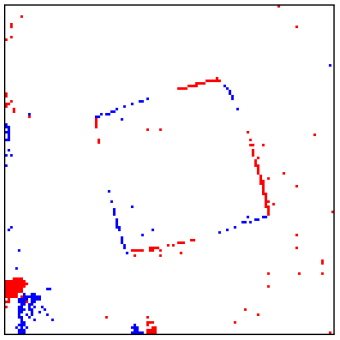
(a) The DVS mounted on a quadrotor: (1) DVS (top) and a standard camera (bottom), (2) single-board computer for data recording, and (3) fiducial markers for tracking.



(b) Quadrotor performing a flip.



(c) Standard CMOS camera.



(d) Integrated DVS events (2 ms).

Fig. 5. Experimental Setup.

TABLE II
RESULTS OF THE QUADROTOR EXPERIMENT.

	Position error [cm]			Orientation error [°]		
	μ	σ	RMS	μ	σ	RMS
[23]	7.3	4.3	8.5	2.8	1.6	3.3
Spline	4.6	3.0	5.5	1.8	1.1	2.1

after our optimization was 0.61 pixels. Again, these results outperform those by algorithm [23].

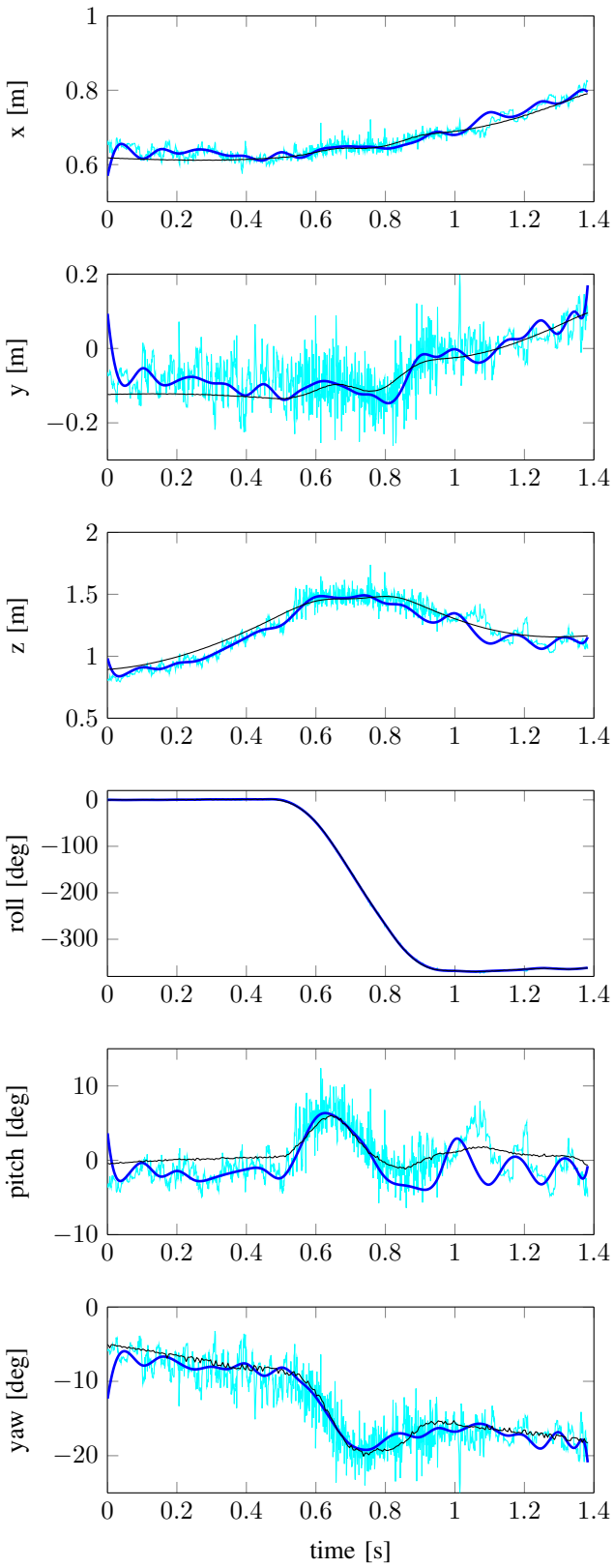


Fig. 6. Plots of the six degrees of freedom for the quadrotor dataset showing the results of the method in [23] (cyan), our method (blue), and ground truth (black).

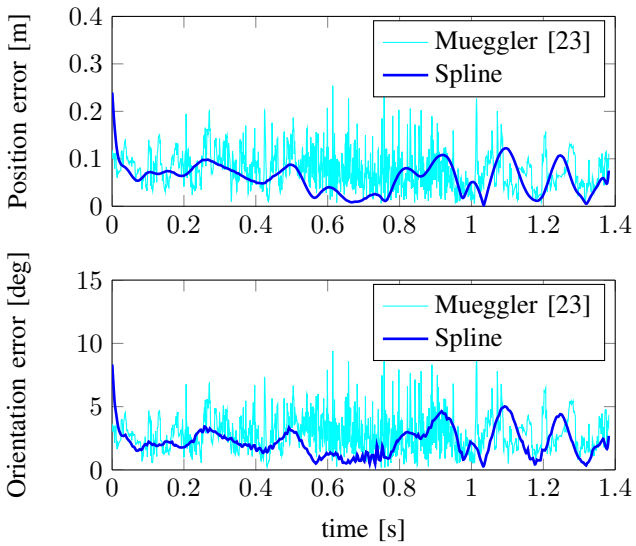


Fig. 7. Position and orientation errors for the quadrotor experiment.

VI. Conclusions and Future Work

In this paper, we presented a method to estimate the trajectory of an event-based vision sensor using a continuous-time framework, which constitutes a first step towards event-based visual SLAM in 6-DOF and without additional sensing. This approach can deal with the high temporal resolution and asynchronous nature of the DVS' events in a principled way, while providing a compact and smooth representation of the trajectory using a parametric cubic spline model. We optimized the approximated trajectory according to a geometrically meaningful error measure in the image plane, which has a probabilistic justification. We tested our method on real sensor data from two experiments. In both the sensor-in-the-loop and flipping-quadrotor datasets, our method outperformed previous algorithms when comparing to the ground truth. While the experiments were carried out with a simplified map, the method can cope with arbitrary scenes composed of line segments, which are common in man-made environments.

Control poses are currently placed equidistant in time, but a more sensible strategy would be to add new control poses according to the event rate and scene complexity. Future work may also extend the method to remove the need for a given map, in the spirit of SLAM. For robotic applications with event-based vision sensors on a mobile platform, our method could be extended to incorporate robot-dynamics motion models.

Acknowledgments

This research was supported by the Swiss National Science Foundation through project number 200021-143607 (Swarm of Flying Cameras), the National Centre of Competence in Research (NCCR) Robotics, and Google.

References

[1] Sameer Agarwal, Keir Mierle, and Others. Ceres solver. http://ceres-solver.org.

[2] H. Alismail, L.D. Baker, and B. Browning. Continuous trajectory estimation for 3D SLAM from actuated lidar. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2014.

[3] S. Anderson, F. Dellaert, and T.D. Barfoot. A hierarchical wavelet decomposition for continuous-time SLAM. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2014.

[4] T.D. Barfoot, C.H. Tong, and S. Särkkä. Batch Continuous-Time Trajectory Estimation as Exactly Sparse Gaussian Process Regression. In *Robotics: Science and Systems (RSS)*, 2014.

[5] R. Benosman, C. Clercq, X. Lagorce, S.-H. Ieng, and C. Bartolozzi. Event-Based Visual Flow. *IEEE Trans. Neural Networks and Learning Systems*, 25(2):407–417, 2014.

[6] P.J. Besl and Neil D. McKay. A method for registration of 3-D shapes. *IEEE Trans. Pattern Anal. Machine Intell.*, 14(2):239–256, 1992.

[7] C. Bibby and I.D. Reid. A hybrid SLAM representation for dynamic marine environments. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2010.

[8] C. Brandli, R. Berner, M. Yang, S.-C. Liu, and T. Delbruck. A 240x180 130dB 3us Latency Global Shutter Spatiotemporal Vision Sensor. *IEEE J. of Solid-State Circuits*, 2014.

[9] A. Censi and D. Scaramuzza. Low-Latency Event-Based Visual Odometry. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2014.

[10] J. Conradt, M. Cook, R. Berner, P. Lichtsteiner, RJ. Douglas, and T. Delbruck. A Pencil Balancing Robot using a Pair of AER Dynamic Vision Sensors. In *Intl. Conf. on Circuits and Systems (ISCAS)*, 2009.

[11] P. Crouch, G. Kun, and F. Silva Leite. The De Casteljau Algorithm on Lie Groups and Spheres. *Journal of Dynamical and Control Systems*, 5(3):397–429, 1999.

[12] D. Drazen, P. Lichtsteiner, P. Hafliger, T. Delbruck, and A. Jensen. Toward real-time particle tracking using an event-based dynamic vision sensor. *Experiments in Fluids*, 51(5):1465–1469, 2011. ISSN 0723-4864.

[13] P. Furgale, T.D. Barfoot, and G. Sibley. Continuous-Time Batch Estimation using Temporal Basis Functions. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2012.

[14] A. Handa, R.A. Newcombe, A. Angeli, and A.J. Davison. Real-Time Camera Tracking: When is High Frame-Rate Best? In *Eur. Conf. on Computer Vision (ECCV)*, 2012.

[15] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2003. Second Edition.

[16] D. Q. Huynh. Metrics for 3D Rotations: Comparison and Analysis. *Journal of Mathematical Imaging and Vision*, 35(2):155–164, 2009.

[17] H. Kim, A. Handa, R. Benosman, S.-H. Ieng, and A. J. Davison. Simultaneous Mosaicing and Tracking with an

Event Camera. In *British Machine Vision Conf. (BMVC)*, 2014.

[18] L. Kneip, D. Scaramuzza, and R. Siegwart. A novel parametrization of the perspective-three-point problem for a direct computation of absolute camera position and orientation. In *Proc. IEEE Int. Conf. Computer Vision and Pattern Recognition*, pages 2969–2976, 2011.

[19] P. Lichtsteiner, C. Posch, and T. Delbruck. A 128×128 120 dB 15 μs latency asynchronous temporal contrast vision sensor. *IEEE J. of Solid-State Circuits*, 43(2): 566–576, 2008.

[20] S.-C. Liu and T. Delbruck. Neuromorphic sensory systems. *Current Opinion in Neurobiology*, 20(3):288–295, 2010.

[21] S. Lovegrove, A. Patron-Perez, and G. Sibley. Spline Fusion: A continuous-time representation for visual-inertial fusion with application to rolling shutter cameras. In *British Machine Vision Conf. (BMVC)*, 2013.

[22] Y. Ma, S. Soatto, J. Košecká, and S. Shankar Sastry. *An Invitation to 3-D Vision: From Images to Geometric Models*. Springer, 2004.

[23] E. Mueggler, B. Huber, and D. Scaramuzza. Event-based, 6-DOF Pose Tracking for High-Speed Maneuvers. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2014.

[24] Z. Ni, A. Bolopion, J. Agnus, R. Benosman, and S. Regnier. Asynchronous Event-Based Visual Shape Tracking for Stable Haptic Feedback in Microrobotics. *IEEE Trans. Robotics*, 28:1081–1089, 2012.

[25] K. Qin. General matrix representations for B-splines. *The Visual Computer*, 16(3–4):177–186, 2000.

[26] J. I. Ronda, A. Valdés, and G. Gallego. Line Geometry and Camera Autocalibration. *J. of Math. Imaging Vis.*, 32(2):193–214, 2008.

[27] E. Trucco and A. Verri. *Introductory Techniques for 3-D Computer Vision*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1998. ISBN 0132611082.

[28] D. Weikersdorfer and J. Conradt. Event-based Particle Filtering for Robot Self-Localization. In *IEEE Intl. Conf. on Robotics and Biomimetics (ROBIO)*, 2012.

[29] D. Weikersdorfer, R. Hoffmann, and J. Conradt. Simultaneous Localization and Mapping for event-based Vision Systems. In *Intl. Conf. on Computer Vision Systems (ICVS)*, 2013.

[30] D. Weikersdorfer, D. B. Adrian, D. Cremers, and J. Conradt. Event-based 3D SLAM with a depth-augmented dynamic vision sensor. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 359–364, June 2014.

[31] Xinyan Yan, Vadim Indelman, and Byron Boots. Incremental Sparse GP Regression for Continuous-time Trajectory Estimation & Mapping. In *NIPS Workshop on Autonomously Learning Robots*, 2014.