Event-based, Direct Camera Tracking from a Photometric 3D Map using Nonlinear Optimization

Samuel Bryner, Guillermo Gallego, Henri Rebecq, Davide Scaramuzza

***Abstract*— Event cameras are novel bio-inspired vision sensors that output pixel-level intensity changes, called "events", instead of traditional video images. These asynchronous sensors naturally respond to motion in the scene with very low latency (in the order of microseconds) and have a very high dynamic range. These features, along with a very low power consumption, make event cameras an ideal sensor for fast robot localization and wearable applications, such as AR/VR and gaming. Considering these applications, we present a method to track the 6-DOF pose of an event camera in a known environment, which we contemplate to be described by a photometric 3D map (i.e., intensity plus depth information) built via classic dense 3D reconstruction algorithms. Our approach uses the raw events, directly, without intermediate features, within a maximum-likelihood framework to estimate the camera motion that best explains the events via a generative model. We successfully evaluate the method using both simulated and real data, and show improved results over the state of the art. We release the datasets to the public to foster reproducibility and research in this topic.**

SUPPLEMENTARY MATERIAL

Video: https://youtu.be/ISgXVgCR-lE
Code, Datasets: http://rpg.ifi.uzh.ch/direct_event_camera_tracking

I. INTRODUCTION

In contrast to standard cameras, which output intensity images at a constant rate, event cameras, such as the Dynamic Vision Sensor (DVS) [1], have independent pixels that output only *intensity changes* (called "events") at the time they occur, with microsecond resolution. Hence, the output of an event camera is a *stream of asynchronous events*. This bio-inspired way of sensing visual information offers several advantages: high temporal resolution and low latency (in the order of microseconds), a very high dynamic range (140dB vs. 60dB of standard cameras), lack of motion blur (since pixels are independent of each other), and low power and bandwidth requirements. Hence, event cameras have a large potential for robotics and wearable applications in challenging scenarios for standard cameras, such as high speed and high dynamic range. Recent plans for mass production claimed by companies, such as Samsung [2] and Prophesee, highlight that there is a big commercial interest in exploiting these new vision sensors for mobile robotic as well as augmented and virtual reality (AR/VR) applications.

The authors are with the Robotics and Perception Group, Depts. of Informatics and Neuroinformatics, University of Zurich and ETH Zurich, Switzerland, http://rpg.ifi.uzh.ch. This work was supported by the Swiss National Center of Competence Research Robotics, through the Swiss National Science Foundation, the SNSF-ERC Starting Grant and the Qualcomm Innovation Fellowship Award 2018.

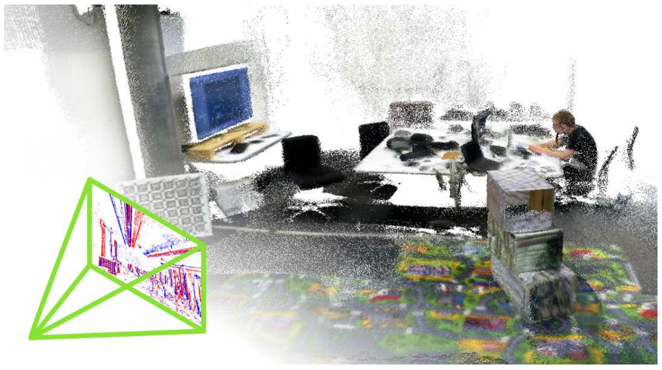


Fig. 1. Considered scenario: the pose of the event camera is computed with respect to a photometric 3D map of the environment, represented as a dense, colored point cloud or mesh. The event camera is represented as a green frustum whose vertex is at its optical center. The image plane of the event camera (at the base of the frustum) displays the events, colored according to polarity (red and blue). Events are caused by the apparent motion of edges on the image plane.

Motivated by these recent developments, in this work, we tackle the problem of tracking the six degree-of-freedom (6-DOF) pose of an event camera in a known environment, as illustrated in Fig. 1. This is the typical scenario of robots traversing previously-mapped spaces, and AR/VR applications (users wear head mounted displays and move freely in the previously-mapped environment). Event cameras offer great potential in this scenario, since they excel at sensing motion, and they do so with very low latency and consuming very little power. We envision that the mobile robot or user (AR/VR) would first use a standard sensor to build a high resolution and high quality map of the environment, and then would take advantage of an event camera to achieve robustness to high-speed motion and low-power consumption.

Because the output of event cameras is fundamentally different from that of standard cameras, traditional vision algorithms cannot be applied, and so, new methods are required. Previous work [3] presented a filter-based approach that requires to maintain a history of past camera poses which are continuously being refined. This led to an unnecessarily complex scheme to update many unknowns: the full history of camera poses. In contrast, we focus on designing a simple yet principled scheme for camera tracking. To this end, we develop a maximum likelihood approach to explain the observed events based on a generative model of them given the map of the environment. This results in a nonlinear optimization approach, as opposed to filtering [3], that allows to achieve accurate camera tracking without having to maintain a history of past poses.

Our method works by comparing the events acquired by the camera to predictions of them, as they would be generated according to the map of the scene and candidate values for the camera pose and its velocity. The comparison error, defined by means of an objective function over the entire image plane, drives the estimation of the unknowns of the problem: the pose of the event camera. Our framework allows to take advantage of the asynchronous, high dynamic range, and low-latency nature of the events to produce low-latency camera poses with high temporal resolution.

In summary, our contributions are the following:

- A novel method for 6-DOF camera tracking with an event camera given a photometric 3D map of the scene. We (*i*) leverage a generative model to explain how events are related to intensity patterns in the map, fully exploiting the strength of the intensity gradients causing the events, and (*ii*) simultaneously estimate the camera pose as well as its linear and angular velocities.
- We thoroughly evaluate the proposed pose tracker on a diversity of scenes and provide a comparison with a state-of-the-art method [3], showing that our approach provides more accurate camera poses.
- We release the datasets used in the experiments, including the acquired images, ground truth camera trajectories, as well as the built photometric maps of the scenes, to foster reproducibility and research in this topic.

Outline: The rest of the paper is organized as follows: Section II reviews the related work on event-based pose tracking, Section III presents our approach, consisting of a maximum likelihood formulation of the observed events in terms of the camera pose, its velocity and the photometric 3D map of the scene. Experiments are carried out and discussed in Section IV to assess the performance of our method, and finally, conclusions are drawn in Section V.

II. RELATED WORK

Since the invention and commercialization of the Dynamic Vision Sensor (DVS) in 2008 [1], the problem of ego-motion estimation with event cameras has been addressed by multiple researchers. Event-based SLAM has been investigated in [4]–[6], and extended to 6-DOF motions in [7]–[13]. These systems, however, target a different scenario than the one we address, namely, they focus on simultaneous map building and localization, which serves to constantly explore new regions of the environment.

In this work, we focus on localization with respect to a given map. This has important applications in robotic navigation (robot moving in a known environment, such as a warehouse) and augmented or virtual reality. Hence, the map of the environment needs to be built only occasionally (e.g., if new objects appear in the scene). Our work is most related to [3], where the authors propose a probabilistic filtering approach to track the 6-DOF pose of the event camera in natural scenes. They use a photometric map of the scene consisting of a sparse set of keyframes (poses, intensity images and depth maps), and consider a robust likelihood function for the generation of the events in terms of the intensity images.

From a high-level point of view, both [3] and this work target the same scenario, and therefore, share the same inputs (events, photometric map) and outputs (camera poses). The differences between them lie in the solution procedure, where [3] adopts a filtering approach, whereas we propose a nonlinear least-squares optimization approach.

Specifically, the method in [3] operates on an event-by-event basis, updating the filter state on every incoming event, thus virtually eliminating latency. This requires to keep a history of past camera poses, which are refined (included in the filter state) as newer events are processed. Past camera poses are utilized to compute the predicted intensity change (i.e., "contrast") of the event, according to a per-event generation model. However, having to update or interpolate poses on a per-event basis has a non-negligible computational cost, regardless of the cost of later processing stages. In contrast, our approach trades off some latency for a simpler event generation model that does not require any reference to past poses. This is sensible because localization does not need to have the granularity of an event (1 microsecond); it can have the granularity of a few events (within 1 millisecond or less). Hence, we process events in small groups (or "windows") to produce a pose. This grouping is not a drawback since we could still produce a pose for every incoming event by sliding the window by a single event. We define the likelihood of a group of events given a pose and seek its maximization, leading to an equivalent, nonlinear least squares problem. Additionally, in contrast to previous approaches, we jointly estimate the camera pose as well as its velocity (linear and angular).

III. METHODOLOGY

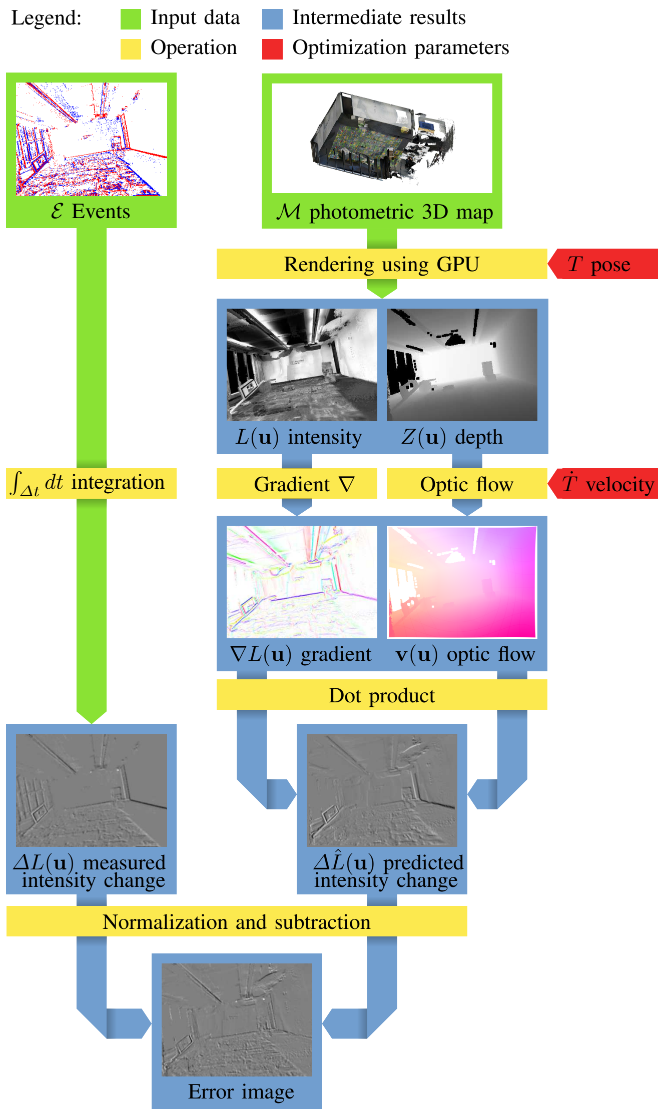
Our method is inspired by [14], where the principled idea of predicting intensity changes and comparing them to those given by actual events was used for feature tracking. Here, instead, we adapt the framework and apply it to the estimation of quantities in 3D space: the 6-DOF pose of the event camera and its velocity. In a nutshell, we cast the pose estimation problem as that of *registering two intensity-change images* that are related by a complex, depth-dependent geometric transformation. In the following, we present the two images being registered and the objective function used to compare them. Our method is summarized in Fig. 2, which illustrates how to compute intensity-change images using the `room` scene of Fig. 8.

A. Event-Camera Working Principle

Each pixel of an event camera produces an event $e_k = (\mathbf{u}_k, t_k, p_k)$ whenever it detects that the logarithmic luminance L at that pixel, $\mathbf{u}_k = (x_k, y_k)^\top$, changes by a specified amount C (called contrast sensitivity) [1]:

$$\Delta L(\mathbf{u}_k, t_k) \doteq L(\mathbf{u}_k, t_k) - L(\mathbf{u}_k, t_k - \Delta t_k) = p_k\, C, \quad (1)$$

where the polarity $p_k = \{+1, -1\}$ is the sign of the brightness change, and Δt_k is the time elapsed since the last



Legend:
Input data
Intermediate results
Operation
Optimization parameters
$\mathcal{E}$ Events
$\mathcal{M}$ photometric 3D map
Rendering using GPU
T pose
$L(\mathbf{u})$ intensity
$Z(\mathbf{u})$ depth
$\int_{\Delta t} dt$ integration
Gradient ∇
Optic flow
$\dot{T}$ velocity
$\nabla L(\mathbf{u})$ gradient
$\mathbf{v}(\mathbf{u})$ optic flow
Dot product
$\Delta L(\mathbf{u})$ measured intensity change
$\Delta \hat{L}(\mathbf{u})$ predicted intensity change
Normalization and subtraction
Error image

Fig. 2. Block diagram of the proposed objective function, which includes the generative event model. The left branch illustrates how the observed intensity-change image is computed from the events (2). The right branch shows how the predicted intensity-change image (8) is computed from the photometric map and the camera motion (pose T and velocity $\dot{T}$). The difference between both intensity-change images is used to define a likelihood function (or, equivalently, an objective function) to be optimized with respect to the event camera motion parameters.

event at the same pixel. The event e_k is a tuple consisting of the space-time coordinates (on the image plane) of the intensity change and its polarity. Event timestamps t_k have microsecond resolution.

B. Intensity-Change Image obtained from the Events

As anticipated in Section II, we process events in groups $\mathcal{E} \doteq \{e_k\}_{k=1}^{N_e}$, spanning a small time interval $\Delta t = t_{N_e} - t_1$. We pixel-wise collect the event polarities over Δt on the whole image plane, producing an image $\Delta L(\mathbf{u})$ with the amount of intensity change that occurred during the interval:

$$\Delta L(\mathbf{u}) = \sum_{t_k \in \Delta t} p_k C\, \delta(\mathbf{u} - \mathbf{u}_k), \qquad (2)$$

where the Kronecker delta δ selects the pixel lattice. This is illustrated on the left branch of the block diagram in Fig. 2. Pixels of $L(\mathbf{u})$ that do not change intensity are represented in gray in ΔL, whereas pixels that increased or decreased intensity are represented in bright and dark, respectively.

C. Intensity-Change Image from the Map and Sensor Pose

The photometric 3D map of the environment is used to produce a prediction of the events that would be generated if the camera moved with some velocity. First, let us define, based on (1), a linearized event generation model, and then let us show how its elements can be computed using the information available: photometric map and candidate camera pose and velocity.

a) Linearized Event Generation Model: For small Δt, such as in the above-mentioned example of Fig. 2, the intensity increments (2) are due to moving edges, as we show next. Substituting the brightness constancy assumption (i.e., optical flow constraint)

$$\frac{\partial L}{\partial t}\big(\mathbf{u}(t), t\big) + \nabla L\big(\mathbf{u}(t), t\big) \cdot \dot{\mathbf{u}}(t) = 0, \qquad (3)$$

with image-point velocity $\mathbf{v} \equiv \dot{\mathbf{u}}$, in Taylor's approximation

$$\Delta L(\mathbf{u}, t) \doteq L(\mathbf{u}, t) - L\big(\mathbf{u}, t - \Delta t\big) \approx \frac{\partial L}{\partial t}(\mathbf{u}, t)\Delta t, \qquad (4)$$

gives (see [15])

$$\Delta L(\mathbf{u}) \approx -\nabla L(\mathbf{u}) \cdot \mathbf{v}(\mathbf{u}) \Delta t, \qquad (5)$$

that is, increments (2) are caused by intensity gradients $\nabla L \doteq (\frac{\partial L}{\partial x}, \frac{\partial L}{\partial y})^\top$ moving with velocity $\mathbf{v}(\mathbf{u})$ over a displacement $\Delta \mathbf{u} \doteq \mathbf{v}\Delta t$. As the dot product in (5) conveys, if the motion is parallel to the edge ($\mathbf{v} \perp \nabla L$), the increment vanishes, and, therefore, no events are generated. From now on, let us denote the modeled intensity change (5) using a hat, $\Delta \hat{L}$. This is illustrated on the right branch of the diagram in Fig. 2.

b) Image $\Delta \hat{L}$ in terms of the Map and the Camera Pose: The elements on the right-hand side of (5), namely the intensity gradient ∇L and the image-point velocity $\mathbf{v}$ (i.e., the motion field, also called "optic flow" in spite of their different nature) can be computed from the photometric map of the scene and the unknowns of the problem.

The image gradient ∇L (5) is obtained by differentiating the intensity image produced by projecting the photometric map $\mathcal{M}$ onto a candidate viewpoint, specified by pose T (see Fig. 2). The motion field $\mathbf{v}$ in (5) is purely geometric, given in terms of the candidate camera pose T, its linear and angular velocities $\dot{T} \doteq (\mathbf{V}^\top, \boldsymbol{\omega}^\top)^\top$, and the depth of the scene $Z \equiv Z(\mathbf{u})$ with respect to the camera, according to [15], [16]:

$$\mathbf{v}(\mathbf{u}) = J(\mathbf{u})\dot{T}, \qquad (6)$$

with the 2×6 feature sensitivity matrix

$$J(\mathbf{u}) = \begin{bmatrix} -1/Z & 0 & x/Z & xy & -(1+x^2) & y \\ 0 & -1/Z & y/Z & 1+y^2 & -xy & -x \end{bmatrix}. \qquad (7)$$

Substituting (6) in (5) gives the intensity change caused by the camera motion in 3D space,

$$\Delta\hat{L}(\mathbf{u}) \approx -\nabla L(\mathbf{u}) \cdot J(\mathbf{u})\dot{T}\Delta t. \tag{8}$$

Observe that the pose T and its velocity $\dot{T}$ are global quantities shared by all pixels $\mathbf{u}$ of the image plane; these are the unknowns of the problem. The rest, i.e., the intensity I and depth $Z(\mathbf{u})$ (given by the photometric map $\mathcal{M}$) and the time span Δt (given by the event timestamps) are all known. The depth $Z(\mathbf{u})$ and the intensity gradient ∇L depend on the projection of the map from the candidate pose T, although this is omitted in the notation for readability.

D. Maximum Likelihood Optimization Framework

As anticipated at the beginning of the section and illustrated at the bottom of Fig. 2 (where both branches meet), we propose to use the difference between two images: the observed intensity changes ΔL from the events (2) and the predicted ones $\Delta\hat{L}$ from the photometric map (8), to estimate the camera motion (pose and velocity).

More specifically, assuming that the difference $\Delta L - \Delta\hat{L}$ follows a zero-mean additive Gaussian distribution with variance σ^2, the likelihood of a group of events $\mathcal{E}$ being generated by a camera moving with respect to a given map $\mathcal{M}$ is

$$p(\mathcal{E}|T,\dot{T};\mathcal{M}) \doteq \frac{1}{\sqrt{2\pi\sigma^2}} \int_{\Omega} e^{-\frac{1}{2\sigma^2}\left(\Delta L(\mathbf{u})-\Delta\hat{L}(\mathbf{u};T,\dot{T})\right)^2} d\mathbf{u}. \tag{9}$$

Our goal is to find the camera pose T and velocity $\dot{T}$ that maximize the event likelihood (9), i.e., that best explain the events with the proposed generative model.

Maximizing (9) with respect to the motion parameters T and $\dot{T}$ (since $\mathcal{M}$, i.e., ∇L, is known) is equivalent to the minimization of the $\mathcal{L}^2$ norm of the intensity-change residual,

$$\min_{T,\dot{T}} \|\Delta L(\mathbf{u}) - \Delta\hat{L}(\mathbf{u};T,\dot{T})\|^2_{\mathcal{L}^2(\Omega)}, \tag{10}$$

where $\|f(\mathbf{u})\|^2_{\mathcal{L}^2(\Omega)} \doteq \int_{\Omega} f^2(\mathbf{u})d\mathbf{u}$. However, (10) depends on the contrast sensitivity C (via (2)), which is unknown in practice. Inspired by [17], we instead propose to minimize the difference between normalized (unit-norm) images:

$$\min_{T,\dot{T}} \left\| \frac{\Delta L(\mathbf{u})}{\|\Delta L(\mathbf{u})\|_{\mathcal{L}^2(\Omega)}} - \frac{\Delta\hat{L}(\mathbf{u};T,\dot{T})}{\|\Delta\hat{L}(\mathbf{u};T,\dot{T})\|_{\mathcal{L}^2(\Omega)}} \right\|^2_{\mathcal{L}^2(\Omega)}, \tag{11}$$

which cancels the terms in C and Δt, and only depends on the direction of the velocity vector $\dot{T}$. The image registration implied by the objective function (11) was implemented, as is standard, using a multiresolution approach (i.e., image-pyramid) and was minimized using the nonlinear least squares solver in the Ceres library [18]. To gain resilience to outliers, we replaced the simple $\mathcal{L}^2$ norm by robust kernels (Huber norm).

Fig. 3 shows an example of a 2D slice of the objective function (11) for an experiment with real data, color coded from blue (small error) to red (large error). By parametrizing the camera's pose using local exponential coordinates (i.e. with respect to an offset rotation [19], [20]) and using the straightforward representations for position and velocity, we obtain a twelve-dimensional parameter space (of which we plot 2D slices). As it is observed, the landscape of the objective function is complex, but it has a clear minimum, reachable provided the solver is initialized in its basin of attraction. As in [3], we assume the initial camera pose is given, and initialize the camera velocity randomly, with about $1\,\mathrm{m/s}$ magnitude. Upon convergence, the pose and velocities are used to initialize the parameters of the optimization corresponding to the next group of events, i.e., next intensity-change image.

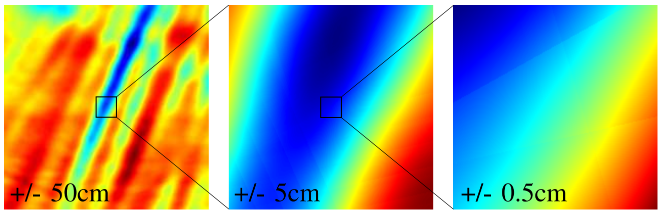


Fig. 3. Objective function (11) (pseudo-colored from blue (low) to red (high)) for variations in the x and y translational components of the pose T of the event camera, shown for $\pm 50\mathrm{cm}$, $\pm 5\mathrm{cm}$ and $\pm 5\mathrm{mm}$.

E. Discussion of the Approach

As mentioned in Section II, our approach does not need to manage a history of past camera poses. This is prevented by the linearization in the event generation model (4) (and (5)). However, for this to be valid, we require to compute the intensity change due to several events, as opposed to the per-event generation model of previous approaches.

An interesting characteristic of our approach is that it is based on the generative event model (8), which not only takes into account that events are triggered at the intensity edges of the projected map, ∇L, but also that the triggering condition depends on the appearance of such edges with respect to the direction of the camera velocity (the appearance of $\Delta\hat{L}$ significantly varies with $\dot{T}$). Our method not only estimates the event camera pose, but also its velocity, and consequently, the optic flow. This dependency was not explicitly modeled in previous works, such as filtering approaches [3], [4], [12].

The number of events N_e used to build an intensity-change images (2) plays an important role in the objective function. A small N_e does not yield sufficient signal-to-noise ratio or evidence of existing edge motion to produce good pose estimates. A large N_e increases latency, produces motion blur in the intensity-change images and breaks the assumption about events being triggered by the camera at a "single" location T (the poses for the first and last event become further apart), which is also harmful for estimation. Hence, there is a trade-off in the selection of N_e, as illustrated in Fig. 4. (a reference value is $0.10\,\mathrm{events/pixel}$). As shown in [14] for feature tracking, it is possible to dynamically set N_e based on the amount of texture present in the scene. In the experiments, we use $N_e/N_p = 0.20\,\mathrm{events/pixel}$, which gives about $N_e \approx 10\,000\,\mathrm{events}$ per intensity-change image.

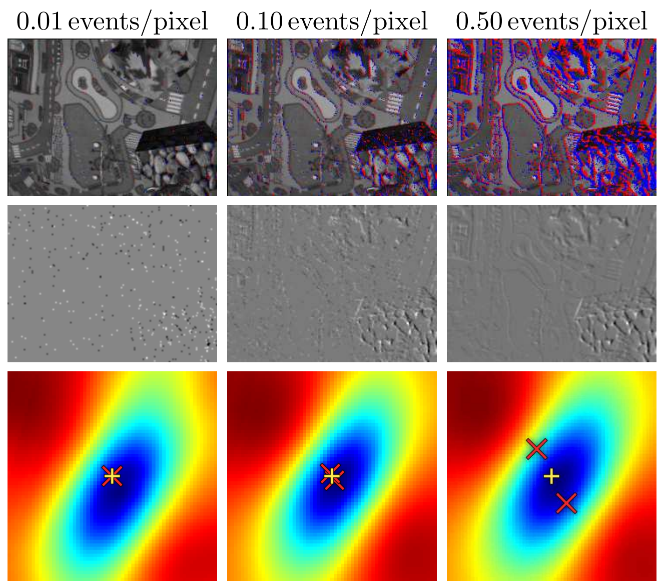


Fig. 4. Effect of varying the integration time Δt, or, equivalently, the number of accumulated events N_e. Columns are sorted according to the increasing value of N_e/N_p, where N_p is the number of pixels of the image. Top row: intensity image $I(\mathbf{u})$ with overlaid events (in red and blue, colored according to polarity). Middle row: intensity-change images $\Delta L(\mathbf{u})$ in (2), obtained by pixel-wise accumulation of event polarities. Bottom row: 2D slice of the 12-D objective function across the X and Y translation axes (± 3 cm). A yellow plus ($+$) indicates the ground truth pose at the center of the event integration window Δt; red crosses ($\times$) indicate the ground truth poses at the beginning and end of the integration window.

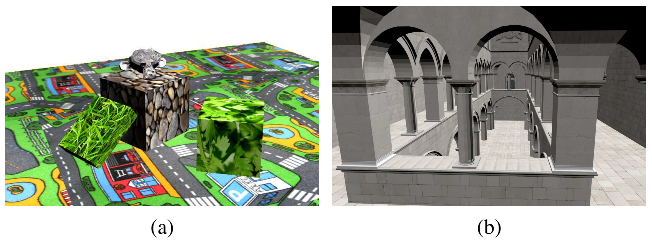


Fig. 5. Two of the synthetic scenes: (a) `toy_room` depicts some textured boxes on a carpet; (b) `atrium` consists of a virtual model of the Atrium Sponza Palace in Dubrovnik.

TABLE I

MEDIAN ACCURACY OF TRACKING OF ALL EXPERIMENTS.

		Pose		Velocity Direction [°]	
Scene	Ave. Depth [m]	Pos. [cm]	Orient. [°]	Linear	Angular
`carpet`	2.11	0.73	0.16	23.56	17.76
`carpet` (D)	2.11	1.12	0.21	20.24	16.17
`toy_room`	1.99	0.45	0.20	17.54	62.69
`toy_room` (D)	1.99	0.89	0.28	17.53	61.17
`atrium`	10.60	6.53	0.43	9.58	5.47
`atrium` (D)	10.60	6.90	0.52	11.17	6.04
`room` Traj. 1	3.20	9.95	3.08	n/a	n/a
`room` Traj. 2	3.41	8.82	3.84	n/a	n/a
`room` (D)	3.41	9.27	3.63	n/a	n/a

(D): downsampled intensity-change images.

IV. EXPERIMENTS

To assess the accuracy of our method, we first evaluate it on synthetic data, where we are able to control scene parameters (depth, illumination, etc.) and we have perfect knowledge of the event camera trajectory. Then, we test our method on real data acquired in an indoor scene. There, the ground truth camera trajectory is provided by a motion-capture system with sub-millimeter accuracy. We release our datasets to the public at http://rpg.ifi.uzh.ch/direct_event_camera_tracking.

A. Evaluation Metrics

The pose estimation error is measured in both position and orientation. The position error is given by the Euclidean distance between the ground truth and the estimated event camera trajectories (position of the optical center). We also report the relative position error with respect to the mean scene depth, as in [3]. The orientation error is measured using the geodesic distance in the rotation group $SO(3)$ [21], which is the angle of the residual rotation between the estimated pose and the ground truth. Since the velocity of the camera $\dot{T}$ can only be estimated up to a non-zero scale factor, to compare two velocity vectors $\dot{T}$ we use the angle between them, using the dot product.

B. Synthetic Data. Assessing Tracking Accuracy

Using simulated data, we validate our method and assess its tracking accuracy. We used the event camera simulator [22], [23] on several scenes with different types of textures and objects: `carpet`, `toy_room` and `atrium`. The `carpet` scene consists of a flat, textured surface. It is an interesting scenario since pose estimation can be ambiguous for planar scenes (rotational and translational motions have similar motion fields), which is more severe if the field-of-view (FOV) of the camera is limited. The `toy_room` scene consists of a couple of boxes with natural textures (e.g., plant leaves) on the same flat surface as `carpet` (see Fig. 5a). The non-flatness of the scene prevents the appearance of pose ambiguities. Finally, the `atrium` scene is a famous computer graphics virtual model of the Atrium Sponza Palace in Dubrovnik. It presents rock-like textures, with strong intensity edges mostly due to boundary of objects (columns, windows, doors, etc.), as shown in Fig. 5b.

In the synthetic scenes, we generated smooth camera trajectories by fitting splines to randomly generated poses. Events were triggered as the camera moved using the simulator [23], with a 240×180 pixel resolution, thus simulating the event camera within the DAVIS240C device [24]. Then, we run our pose tracking algorithm, feeding as photometric 3D map the virtual model of the scene (i.e., a colored mesh comprising the surface of all objects in the scene). The resulting pose tracking errors of our method are summarized in the top half of Table I.

As can be observed in Table I, errors across all simulated scenes are remarkably small: less than $0.65\,\%$ in relative position error (with respect to the mean depth of the scene) and less than $0.52°$ in orientation error. Moreover, tracking in the planar scene (`carpet`) does not produce motion ambiguities, e.g., x-translation is properly estimated, disambiguated from a y-rotation, in spite of the limited FOV

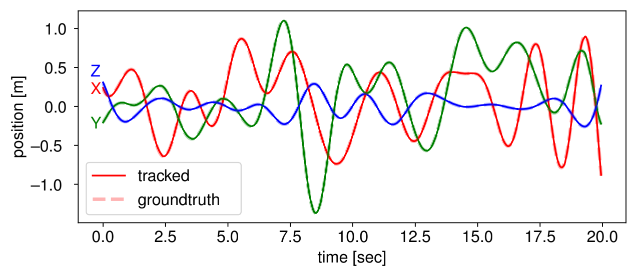


Fig. 6. Pose tracking results over time for a smooth random motion over a planar texture (`carpet` scene). Tracking does not suffer from ambiguities. The median position error is 0.73 cm, which corresponds to a relative error of 0.34 % with respect to the mean scene depth. The median orientation error is 0.16° (see Table I).

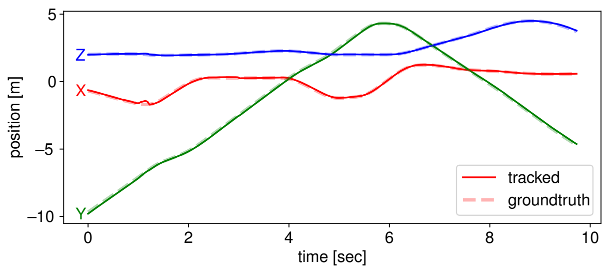


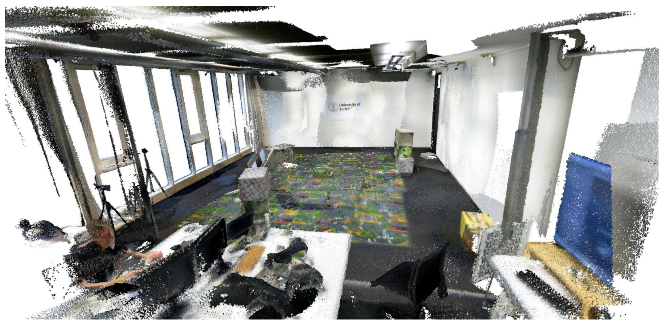
Fig. 7. Pose tracking results for a simulated flight trough the `atrium` scene in Fig. 5b. The median position error is 6.53 cm, which amounts to a relative error of 0.62 % with respect to the mean scene depth (10.60 m). The median orientation error is 0.43°.

of the camera (60° horizontally and 50° vertically). Fig. 6 displays the translation of the camera in this planar scene. As it is observed, the estimated trajectory (solid lines) and ground truth one (dashed lines) are almost indistinguishable compared to the excursion of the motion.

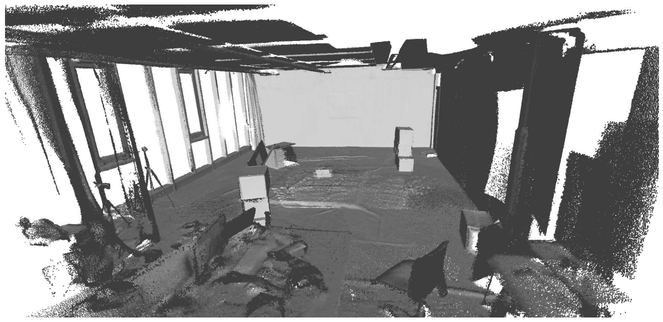
The `atrium` experiments present the largest absolute errors since the scene is considerably larger than the previous two. However, in terms of relative error with respect to the mean scene depth, errors are of the same order of magnitude as those of the experiments in the `carpet` and `toy_room` scenes. Fig. 7 shows both the estimated trajectory of the event camera moving through the scene (solid line) and the ground truth trajectory (dashed line). The curves are almost on top of each other since errors are small compared to the amount of motion.

Supplementary Material: We encourage the reader to watch the accompanying video, which shows the experiments here presented in a better form than still images can convey.

Using Downsampled Intensity-Change Images: We also tested the pose tracking algorithm using lower resolution intensity-change images. As mentioned in Section III, the method is implemented using a multiresolution approach. In these experiments, we registered intensity-change images ΔL and $\Delta \hat{L}$ at the second highest-resolution level of the image pyramid (e.g., 120×90 pixels for the DAVIS240C). Such experiments are marked as "(D)" in Table I. This modification had a minimal effect on accuracy loss (Table I) while it significantly sped up computation time, as reported in Table II.



(a) Photometric information (color)



(b) Geometric information (shape)

Fig. 8. `room` dataset. Photometric map used, generated with ElasticFusion [25]. Some artifacts in texture-less areas are present due to auto-exposure and white-balance of the consumer-grade RGB-D camera used.

C. Real Data

Experiments were also conducted on real scenes. The event camera was moved hand-held in a room comprising texture as well as large white walls (i.e., texture-less). For this experiment, we used the DAVIS346 (346×260 pixels) from iniVation. Ground truth poses were provided by a motion-capture system. The room was pre-mapped using a consumer RGB-D camera (ASUS Xtion Pro) and running open-source software ElasticFusion [25]. For improved accuracy of the 3D reconstructed map, we provided to ElasticFusion poses from the motion capture system. The resulting color and shape of the 3D-reconstructed photometric map are shown in Figs. 8a and 8b, respectively. Auto-exposure and white-balance were enabled during recording with the RGB-D camera, leading to some unseemly intensity edges in the map in what are actually uniform white walls (see Fig. 8a).

The results of our pose tracking method on a couple of trajectories of the event camera under 6-DOF motion in this room environment are reported in the bottom half of Table I.

In this scenario tracking is more challenging than in previous scenarios for a number of reasons: (*i*) the presence of uniformly white walls and a floor with high-frequency texture, (*ii*) the presence of noise in both inputs: the events and the photometric map, (*iii*) the presence of inaccuracies in the calibration of the event camera and the hand-eye calibration with respect to the motion-capture system. In spite of these source of errors, our method is still able to track gracefully, with relative position errors of 2.58 % to 3.10 % and orientation errors of 3.08° to 3.84°. A visualization of the pose tracking results (events aligned with the reprojected

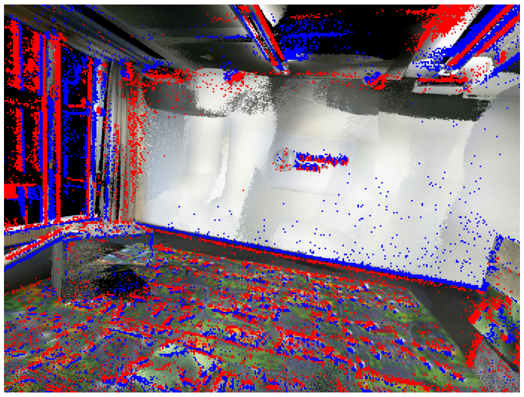


Fig. 9. Pose tracking visualization on one of the sequences of the `room` dataset (Fig. 8). Events overlaid on the projected map from the viewpoint of the estimated camera pose. Events are colored in red and blue, according to polarity (positive in blue, negative in red). The alignment of the majority of the events with respect to the intensity edges of the projected map is a good indicator of the accuracy of the estimated camera pose.

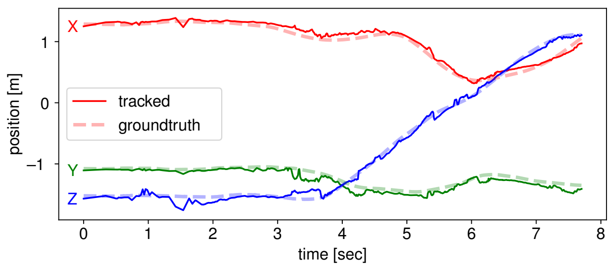


Fig. 10. Pose tracking results on the `room` scene (Traj. 1) using the photometric map in Fig. 8. The median position error is 9.95 cm with respect to a mean scene depth of 3.20 m, that is, a 3.10 % relative error. The median orientation error is 3.08°.

photometric map) is given in Fig. 9. The estimated and ground truth trajectories of one of the sequences are displayed in Fig. 10. Compared to noise-free synthetic data, we observe the decrease in tracking accuracy in the presence of the above-mentioned noise sources.

D. Computational Load

We also report the computational effort required by our non-optimized C++ implementation of the proposed approach. Timing results are presented in Table II for both simulated and real data. The current implementation, built on top of Ceres [18], is one to three orders of magnitudes slower than real-time, depending on the settings. Making the approach real-time capable is of course essential for any practical application. As mentioned in Section IV-B, using lower resolution intensity-change images is an option to speed up the algorithm with a minor impact on accuracy loss. To further reduce pixel count one could heuristically filter pixels and only keep those that contribute significantly to the error. On the implementation side there is room for optimization using parallelization: the pixels are independent, only the last step of solving (11) is inherently sequential. Our method would thus be ideally suited for a more distributed platform, such as a GPU (currently, all computation is done on the CPU, except for the rendering of the map, where the GPU is mainly a convenience), which was used in other works [12], [26], [27].

TABLE II
COMPUTATION TIME REQUIRED FOR POSE TRACKING.

		Processing	Full Tracking Step		
Scene	Length [s]	Events/sec	Iter./step	ms/iter.	total [ms]
`carpet`	20	2857	24.66	120.06	3024.70
`carpet` (D)	20	12 789	17.01	38.52	675.70
`toy_room`	20	3160	25.94	132.26	3418.50
`toy_room` (D)	20	6433	27.59	59.59	1679.30
`atrium`	10	4130	12.23	166.07	2098.60
`atrium` (D)	10	11 532	17.34	41.01	751.60
`room` Traj. 1	8.32	995	64.63	359.54	22 704.00
`room` Traj. 2	43.24	1016	90.31	315.08	22 139.20

(D): downsampled intensity-change images.
The upper half are simulated environments.
These results were obtained on a single Intel i7-870 core.

TABLE III
COMPARISON OF ROOT-MEAN-SQUARE (RMS) POSE ERRORS WITH RESPECT TO STATE-OF-THE-ART.

		Gallego et al. [3]			This work		
	Length	Position		Orientation	Position		Orientation
	[s]	[cm]	[%]	[°]	[cm]	[%]	[°]
`boxes 1`	23.3	5.08	2.69	2.51	**4.74**	**2.52**	**1.86**
`boxes 2`	26.7	**4.04**	**2.15**	2.18	4.46	2.38	**2.10**
`boxes 3`	33.7	5.47	2.90	2.82	**5.05**	**2.68**	**2.39**
`pipe 1`	29.8	10.96	4.04	2.90	**10.23**	**3.77**	**2.13**
`pipe 2`	22.2	15.26	5.34	4.68	**11.29**	**3.95**	**4.02**

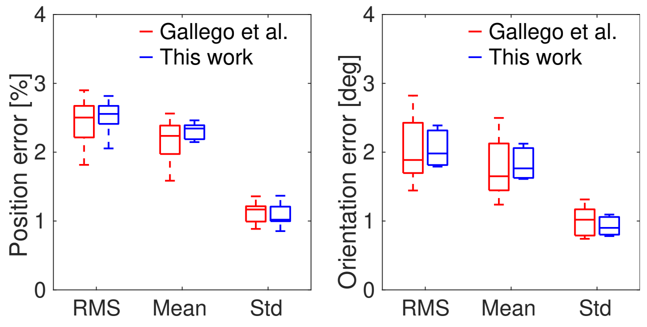


Fig. 11. Comparison with [3, Fig. 13]. Error in position (relative to a mean scene depth of 1.90 m) and orientation (in degrees) of the camera poses recovered by the method in [3] (in red) and by our method (in blue) for the `boxes` sequences from [3] (ground truth is given by a motion-capture system). We provide box plots of the root-mean-square (RMS) errors, the mean errors and the standard deviation (Std) of the errors.

E. Comparison with State of the Art

We compared our method against the state-of-the-art [3]. To this end, we run our pose tracking algorithm on the same datasets used in [3], and compared the resulting tracking errors. These are reported in Table III. As it can be observed, our method offers improved accuracy over [3], specially in orientation estimation. Smallest errors per row, per error type are highlighted in bold. Orientation error improves, on average, by 15 %, and position error, by 7 %. Differences are most notorious in the outdoor (`pipe`) sequences. Both errors are related, as it is well known that errors in orientation at some time t produce worse translation estimates at later times, and vice-versa. Additional comparison on more `boxes` sequences of [3] is provided in Fig. 11. Our method presents more concentrated error statistics than [3].

F. Discussion

The previous experiments demonstrate that the proposed method accurately tracks the pose of the event camera in both simulated and real-world environments. In simulation, position errors are smaller than $0.65\,\%$, and orientation errors smaller than half a degree. These errors are very small and validate our method with respect to noise-free event data. The resulting pose tracking errors are imputable to modeling errors (such as the linearization (4), the approximation of intensity gradients using finite differences, as well as non-explicitly modeled effects, e.g., occlusions and disocclusions). While poses are estimated very accurately, camera velocities are not as accurately estimated, which is due to velocities being very sensitive to error sources, such as modeling errors, as can be seen with perfect data (noise-free events and map).

The above tracking errors increase to about $3\,\%$ and $4°$, respectively, in real-world sequences. The accuracy gap between simulation and real world experiments is due to additional error sources: (*i*) noise in the photometric map reconstructed from a noisy RGB-D camera, (*ii*) noise and non-idealities of event cameras [1], (*iii*) small delays among the sensors, and (*iv*) inaccuracies in internal and external calibration of all sensors involved to produce the "ground truth" data used for evaluation. In spite of all these factors, pose tracking still performs well in real-world conditions.

V. Conclusion

We have presented a method to track the 6-DOF pose of an event camera with respect to a photometric 3D map, which is a paramount scenario in robotics and AR/VR scenarios. Our method leverages a principled event generation model within a maximum-likelihood framework to jointly estimate the camera poses and velocities, through nonlinear optimization, fully exploiting the strength of the intensity gradients in the scene. A thorough evaluation on both synthetic and real data proves that our method provides compelling accuracy, and it improves over the state-of-the-art filtering approach [3] while being simpler to formulate. A key characteristic of our method is that it uses the events, directly, without intermediate hand-crafted features, thus exploiting as much of the information contained in the events as possible. Additionally, we release all our datasets with ground truth poses, in an effort to foster reproducibility and research in this topic.

References

[1] P. Lichtsteiner, C. Posch, and T. Delbruck, "A 128×128 120 dB 15 μs latency asynchronous temporal contrast vision sensor," *IEEE J. Solid-State Circuits*, vol. 43, no. 2, pp. 566–576, 2008.

[2] B. Son, Y. Suh, S. Kim, H. Jung, J.-S. Kim, C. Shin, K. Park, K. Lee, J. Park, J. Woo, Y. Roh, H. Lee, Y. Wang, I. Ovsiannikov, and H. Ryu, "A 640x480 dynamic vision sensor with a 9um pixel and 300Meps address-event representation," in *IEEE Intl. Solid-State Circuits Conf. (ISSCC)*, 2017.

[3] G. Gallego, J. E. A. Lund, E. Mueggler, H. Rebecq, T. Delbruck, and D. Scaramuzza, "Event-based, 6-DOF camera tracking from photometric depth maps," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 10, pp. 2402–2412, Oct. 2018.

[4] H. Kim, A. Handa, R. Benosman, S.-H. Ieng, and A. J. Davison, "Simultaneous mosaicing and tracking with an event camera," in *British Mach. Vis. Conf. (BMVC)*, 2014.

[5] A. Censi and D. Scaramuzza, "Low-latency event-based visual odometry," in *IEEE Int. Conf. Robot. Autom. (ICRA)*, 2014, pp. 703–710.

[6] C. Reinbacher, G. Munda, and T. Pock, "Real-time panoramic tracking for event cameras," in *IEEE Int. Conf. Comput. Photo. (ICCP)*, 2017.

[7] D. Weikersdorfer, D. B. Adrian, D. Cremers, and J. Conradt, "Event-based 3D SLAM with a depth-augmented dynamic vision sensor," in *IEEE Int. Conf. Robot. Autom. (ICRA)*, 2014, pp. 359–364.

[8] B. Kueng, E. Mueggler, G. Gallego, and D. Scaramuzza, "Low-latency visual odometry using event-based feature tracks," in *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*, 2016, pp. 16–23.

[9] E. Mueggler, G. Gallego, and D. Scaramuzza, "Continuous-time trajectory estimation for event-based vision sensors," in *Robotics: Science and Systems (RSS)*, 2015.

[10] H. Rebecq, T. Horstschäfer, G. Gallego, and D. Scaramuzza, "EVO: A geometric approach to event-based 6-DOF parallel tracking and mapping in real-time," *IEEE Robot. Autom. Lett.*, vol. 2, no. 2, pp. 593–600, 2017.

[11] H. Rebecq, G. Gallego, E. Mueggler, and D. Scaramuzza, "EMVS: Event-based multi-view stereo—3D reconstruction with an event camera in real-time," *Int. J. Comput. Vis.*, vol. 126, no. 12, pp. 1394–1414, Dec. 2018.

[12] H. Kim, S. Leutenegger, and A. J. Davison, "Real-time 3D reconstruction and 6-DoF tracking with an event camera," in *Eur. Conf. Comput. Vis. (ECCV)*, 2016, pp. 349–364.

[13] G. Gallego, H. Rebecq, and D. Scaramuzza, "A unifying contrast maximization framework for event cameras, with applications to motion, depth, and optical flow estimation," in *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, 2018, pp. 3867–3876.

[14] D. Gehrig, H. Rebecq, G. Gallego, and D. Scaramuzza, "Asynchronous, photometric feature tracking using events and frames," in *Eur. Conf. Comput. Vis. (ECCV)*, 2018.

[15] G. Gallego, C. Forster, E. Mueggler, and D. Scaramuzza, "Event-based camera pose tracking using a generative event model," 2015, arXiv:1510.01972.

[16] P. Corke, *Robotics, Vision and Control: Fundamental Algorithms in MATLAB*, ser. Springer Tracts in Advanced Robotics. Springer, 2017.

[17] G. D. Evangelidis and E. Z. Psarakis, "Parametric image alignment using enhanced correlation coefficient maximization," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 10, pp. 1858–1865, Oct. 2008.

[18] S. Agarwal, K. Mierle, and Others, "Ceres solver," http://ceres-solver.org.

[19] Z. Zhang, G. Gallego, and D. Scaramuzza, "On the comparison of gauge freedom handling in optimization-based visual-inertial state estimation," *IEEE Robot. Autom. Lett.*, vol. 3, no. 3, pp. 2710–2717, Jul. 2018.

[20] G. Gallego and A. Yezzi, "A compact formula for the derivative of a 3-D rotation in exponential coordinates," *J. Math. Imaging Vis.*, vol. 51, no. 3, pp. 378–384, Mar. 2015.

[21] D. Q. Huynh, "Metrics for 3D rotations: Comparison and analysis," *J. Math. Imaging Vis.*, vol. 35, no. 2, pp. 155–164, 2009.

[22] E. Mueggler, H. Rebecq, G. Gallego, T. Delbruck, and D. Scaramuzza, "The event-camera dataset and simulator: Event-based data for pose estimation, visual odometry, and SLAM," *Int. J. Robot. Research*, vol. 36, no. 2, pp. 142–149, 2017.

[23] H. Rebecq, D. Gehrig, and D. Scaramuzza, "ESIM: an open event camera simulator," in *Conf. on Robotics Learning (CoRL)*, 2018.

[24] C. Brandli, R. Berner, M. Yang, S.-C. Liu, and T. Delbruck, "A 240x180 130dB 3us latency global shutter spatiotemporal vision sensor," *IEEE J. Solid-State Circuits*, vol. 49, no. 10, pp. 2333–2341, 2014.

[25] T. Whelan, S. Leutenegger, R. F. Salas-Moreno, B. Glocker, and A. J. Davison, "ElasticFusion: Dense SLAM without a pose graph," in *Robotics: Science and Systems (RSS)*, 2015.

[26] P. Bardow, A. J. Davison, and S. Leutenegger, "Simultaneous optical flow and intensity estimation from an event camera," in *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, 2016, pp. 884–892.

[27] A. Z. Zhu, Y. Chen, and K. Daniilidis, "Realtime time synchronized event-based stereo," in *Eur. Conf. Comput. Vis. (ECCV)*, 2018.