Ultimate SLAM? Combining Events, Images, and IMU for Robust Visual SLAM in HDR and High-Speed Scenarios

Antoni Rosinol Vidal, Henri Rebecq, Timo Horstschaefer, and Davide Scaramuzza

Abstract—**Event cameras are bioinspired vision sensors that output pixel-level brightness changes instead of standard intensity frames. These cameras do not suffer from motion blur and have a very high dynamic range, which enables them to provide reliable visual information during high-speed motions or in scenes characterized by high dynamic range. However, event cameras output only little information when the amount of motion is limited, such as in the case of almost still motion. Conversely, standard cameras provide instant and rich information about the environment most of the time (in low-speed and good lighting scenarios), but they fail severely in case of fast motions, or difficult lighting such as high dynamic range or low light scenes. In this letter, we present the first state estimation pipeline that leverages the complementary advantages of these two sensors by fusing in a tightly coupled manner events, standard frames, and inertial measurements. We show on the publicly available Event Camera Dataset that our *hybrid* pipeline leads to an accuracy improvement of 130% over *event-only* pipelines, and 85% over *standard-frames-only* visual-inertial systems, while still being computationally tractable. Furthermore, we use our pipeline to demonstrate—to the best of our knowledge—the first autonomous quadrotor flight using an event camera for state estimation, unlocking flight scenarios that were not reachable with traditional visual-inertial odometry, such as low-light environments and high dynamic range scenes. Videos of the experiments: http://rpg.ifi.uzh.ch/ultimateslam.html**

Index Terms—**SLAM, visual-based navigation, aerial systems: perception and autonomy.**

I. Introduction

THE TASK of estimating a sensor's ego-motion has important applications in various fields, such as augmented/virtual reality or autonomous robot control. In recent years, great progress has been achieved using visual and inertial information ([1]–[3]). However, due to some well-known limitations of traditional cameras (motion blur and low dynamic-range), these Visual Inertial Odometry (VIO) pipelines still struggle to cope with a number of situations, such as high-speed motions or high-dynamic range scenarios.

Novel types of sensors, called event cameras, offer great potential to overcome these issues. Unlike standard cameras, which transmit intensity frames at a fixed framerate, event cameras, such as the Dynamic Vision Sensor (DVS) [4], only transmit *changes of intensity*. Specifically, they transmit per-pixel intensity changes at the time they occur, in the form of a set of asynchronous *events*, where each event carries the space-time coordinates of the brightness change, and its sign.

Event cameras have numerous advantages over standard cameras: a latency in the order of microseconds and a very high dynamic range (140 dB compared to 60 dB of standard cameras). Most importantly, since all the pixels capture light independently, such sensors do not suffer from motion blur.

Event cameras transmit, in principle, all the information needed to reconstruct a full video stream [5]–[7], and one could argue that an event camera alone is sufficient to perform state estimation. In fact, this has been shown recently in [8] and [9]. However, to overcome the lack of intensity information, these approaches need to reconstruct, in parallel, a consistent representation of the environment (a semi-dense depth map in [8] or a dense depth map with intensity values in [9]), by combining—in one way or another—information from a large number of events to recover most gradients in the scene.

Conveniently, standard cameras provide direct access to intensity values, but do not work in low-light conditions, suffer from motion blur during fast motions (due to the synchronous exposure on the whole sensor), and have a limited dynamic range (60 dB), resulting in frequent over- or under-exposed areas in the frame.

Observing this complementarity, in this letter we propose a pipeline that leverages the advantages of both sensing modalities in combination with an inertial measurement unit (IMU) to yield a robust, yet accurate, state estimation pipeline.

While there is a considerable body of literature investigating the use of standard cameras with an IMU to perform state estimation, as well as recent work using an event camera with an IMU, combining all three sensing modalities is yet an open problem. Additionally, in the core application that we envision—flying autonomously a quadrotor with an event camera—there is no specific literature, although attempts to use an event camera for

Manuscript received September 10, 2017; accepted December 4, 2017. Date of publication January 15, 2018; date of current version February 1, 2018. This letter was recommended for publication by Associate C. Pradalier and Editor F. Chaumette upon evaluation of the reviewers' comments. This work was supported in part by the DARPA FLA program, in part by the Swiss National Center of Competence Research Robotics, through the Swiss National Science Foundation, and in part by the SNSF-ERC starting grant. *(Antoni Rosinol Vidal and Henri Rebecq contributed equally to this work.) (Corresponding author: Henri Rebecq.)*

The authors are with the Robotics and Perception Group, Department of Neuroinformatics, ETH Zurich and University of Zurich, Zurich CH-8057 Switzerland, and also with the Department of Informatics, University of Zurich, Zurich CH-8006, Switzerland (e-mail: antonir@student.ethz.ch; rebecq@ifi.uzh.ch; mail@timo-horstschaefer.de; davide.scaramuzza@ieee.org).

Digital Object Identifier 10.1109/LRA.2018.2793357

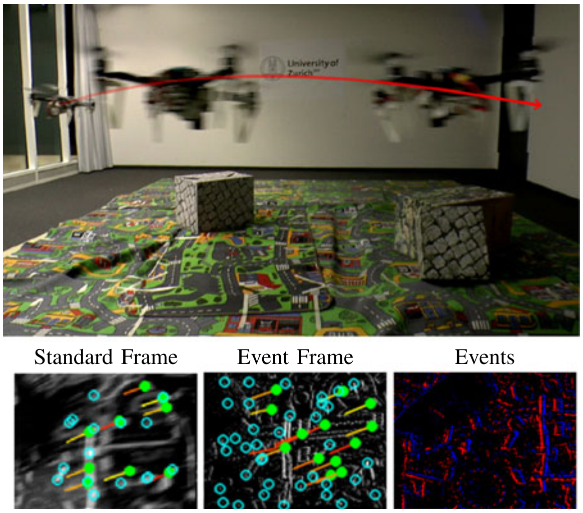

Fig. 1. Our state estimation pipeline combines events, standard frames, and inertial measurements to provide robust state estimation, and can run onboard an autonomous quadrotor with limited computational power. Bottom Left: Standard frame, Bottom Middle: Virtual event frame, Bottom Right: Events only (blue: positive events, red: negative events).

quadrotor flight can be traced to a single paper [10], which is currently limited to vertical landing maneuvers.

In this work, we propose—to the best of our knowledge—the first state estimation pipeline that fuses all three sensors, and we build on top of it to propose the first quadrotor system that can advantageously exploit this hybrid sensor combination to fly in difficult scenarios, using only onboard sensing and computing (see Fig. 1).

A. *Contributions*

A frontal comparison with state-of-the-art, commercial visual-inertial pipelines (like for example the ones used for the Snapdragon flight [11] or Google Tango [12]) is not our goal in this work. Indeed, such solutions typically use one or more high quality cameras with a much higher resolution than the sensor we used, and are carefully engineered to work well in the most common consumer situations. Instead, in this work, we focus on difficult scenarios, and show, for the first time, that (i) it is possible to run state estimation with an event camera onboard a computationally limited platform, and (ii) we show that it can unlock, in a set of difficult scenarios, the possibility for autonomous flight where even commercial systems would struggle.

Specifically, our contributions in this letter are three-fold:

- We introduce the first state estimation pipeline that fuses events, standard frames, and inertial measurements to provide robust and accurate state estimation. While our pipeline is based on [13], we extend it to include standard frames as an additional sensing modality, and propose several improvements to make it usable for real-time applications, with a focus on mobile robots.
- We evaluate quantitatively the proposed approach and show that using standard frames as an additional modality improves the accuracy of state estimation while keeping the computational load tractable.
- We show that our method can be applied for state estimation onboard an autonomous quadrotor, and demonstrate in a set of experiments that the proposed system is able to fly reliably in challenging situations, such as low-light scenes or fast motions.

Our work aims at highlighting the potential that event cameras have for robust state estimation, and we hope that our results will inspire other researchers and industries to push this work forward, towards the wide adoption of event cameras on mobile robots.

The rest of the paper is organized as follows: Section II reviews related literature on event-based ego-motion estimation methods, particularly those involving event cameras. In Section III, we present our hybrid state estimation pipeline that fuses events, standard frames and inertial measurements in a tightly-coupled fashion, and evaluate it quantitatively on the publicly available Event Camera Dataset [14]. Section IV describes how the proposed approached can be used to fly a quadrotor autonomously, and demonstrate in a set of real-life experiments that it unlocks challenging scenarios difficult to address with traditional sensing IV-B. Finally, we draw conclusions in Section V.

II. Related Work

Using visual and inertial sensors for state estimation has been extensively studied over the past decades. While the vast majority of these works use standard cameras together with an IMU, a recent parallel thread of research that uses event cameras in place of standard cameras has recently flourished.

A. *Visual-Inertial Odometry With Standard Cameras*

The related work on visual-inertial odometry (VIO) can be roughly segmented into three different classes, depending on the number of camera poses that are used for the estimation. While full smoothers (or batch nonlinear least-squares algorithms) estimate the complete history of poses, fixed-lag smoothers (or sliding window estimators) consider a window of the latest poses, and filtering approaches only estimate the latest state. Both fixed-lag smoothers and filters marginalize older states and absorb the corresponding information in a Gaussian prior. More specifically:

- Filtering algorithms enable efficient estimation by restricting the inference process to the latest state of the system. A example approach of a filter-based visual-inertial odometry system is [15].
- Fixed-lag smoothers estimate the states that fall within a given time window, while marginalizing out older states, as for example, [2].
- Full smoothing methods estimate the entire history of the states (camera trajectory and 3D landmarks), by solving a large nonlinear optimization problem. A recent approach in this category was proposed by [3].

B. *Visual-Inertial Odometry With Event Cameras*

Since the introduction of the first commercial event camera in 2008 [4], event cameras have been considered for state estimation by many different authors. While early works focused on addressing restricted and easier instances of the problem, like rotational motion estimation ([5], [16], [17], [18]), or Simultaneous Localization and Mapping (SLAM) in planar scenes only [19], it has been shown recently that 6-DOF pose estimation using only an event camera is possible ([8], [9]).

In parallel, other authors have explored the use of complementary sensing modalities, such as a depth sensor [20], or a standard camera ([21], [22]). However, (i) none of these image-based pipelines make use of inertial measurements, and (ii) both of them use the intensity of the frames as a template, to which they align the events. Therefore, these approaches work only when the standard frames are of good quality (sharp and correctly exposed); they will fail in those particular cases where the event camera has an advantage over a standard camera (high-speed motions, and HDR scenes).

Using an event camera and an IMU has only been explored very recently. [23] showed how to fuse events and inertial measurements into a continuous time framework, but their approach is not suited for real-time usage because of the expensive optimization required to update the spline parameters upon receiving every event. [24] proposed to track a set of features in the event stream using an iterative Expectation-Maximization scheme that jointly refines each feature's appearance and optical flow, and then fuse these tracks using an Extended Kalman Filter to yield an event-based visual-inertial odometry pipeline. Unfortunately, due to the expensive nature of their feature tracker, the authors of [24] reported that their pipeline cannot run in real-time in most scenarios.

In [13], we proposed an accurate event-based visual inertial odometry pipeline that can run in real-time, even on computationally limited platforms, such as smartphone processors. The key of this approach was to estimate the optical flow generated by the camera's rigid body motion by exploiting the current camera pose, scene structure, and inertial measurements. We then efficiently generated virtual, motion-compensated event frames using the computed flow, and further tracked visual features across multiple frames. Those feature tracks were finally fused with inertial information using keyframe-based nonlinear optimization, in the style of [2] and [3]. While our proposed state estimation approach is strongly inspired by this work (i.e., [13]), we extend it by allowing it to additionally work with frames from a standard camera, and propose several changes to the pipeline to adapt it to run onboard a flying robot.

C. Quadrotor Control With an Event Camera

Although the research on robot control with event cameras is still in its infancy, previous work has demonstrated possible interesting applications. [25] mounted a DVS sensor on a quadrotor and showed that it can be used to track the 6-DOF motion of a quadrotor performing a high speed flip maneuver, although the tracker only worked for an artificial scene containing a known black square painted over a white wall. Also, the state estimation was performed offline, and therefore not used for closed-loop control of the quadrotor. More recently, [10] showed closed-loop take-off and landing of a quadrotor using an event camera. Their system, however, relied on computing optical flow and assumed the flow field to be divergent, thus it cannot be used for general 6-DOF control of a quadrotor, unlike our approach.

III. Hybrid State Estimation Pipeline

Our proposed state estimation pipeline is largely based on [13]. However, while [13] used only an event camera combined with an IMU, we propose to allow for an additional sensing modality: a standard camera, providing intensity frames at a fixed framerate. For this reason, we focus below on describing the differences between our approach and [13] in order to also consider standard frames. Finally, we evaluate the improved pipeline on the Event Camera Dataset [14] and show evidence that incorporating standard frames in the pipeline leads to an accuracy boost of 130% over a pipeline that uses only events plus IMU, and 85% over a pipeline that uses only standard frames plus IMU.

A. Overview

[13] can be briefly summarized as follows. The main idea is to synthesize virtual frames (*event frames*) from spatio-temporal windows of events, and then perform feature detection and tracking using classical computer vision methods, namely the FAST corner detector [26] and the Lucas-Kanade tracker [27]. Feature tracks are used to triangulate the 3D locations of the corresponding landmarks whenever it can be done reliably. Finally, the camera trajectory and the positions of the 3D landmarks are periodically refined by minimizing a cost function involving visual terms (reprojection error) and inertial terms, thus effectively fusing visual and inertial information.

In this letter, we propose to not only maintain feature tracks from virtual event frames, but to also maintain, in parallel, feature tracks from standard frames as well. We then feed the feature tracks coming from these two heterogeneous sources (virtual event frames and standard frames) to the optimization module, thus effectively refining the camera poses using the events, the standard frames, and the IMU.

1) 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 hybrid sensor composed of an event camera, a standard camera, and an IMU rigidly mounted together. The sensor body is represented relative to an inertial world frame W. Within the sensor body, we distinguish the event camera frame C_0, the standard camera frame C_1 and the IMU-sensor frame S. To obtain $\mathbf{T}_{SC_0}$ and $\mathbf{T}_{SC_1}$, an extrinsic calibration of the $\{$event camera + standard camera + IMU system$\}$ must be performed.

2) Spatio-Temporal Windows of Events: We synchronize the spatio-temporal windows of events on the timestamps of the standard frames. Upon reception of each standard frame at time t_k, a new spatio-temporal window of events W_k is created (see Fig. 2). The kth window is defined as the set of events $W_k = \left\{e_{j(t_k)-N+1}, ..., e_{j(t_k)}\right\}$, where $j(t_k)$ is the index of the

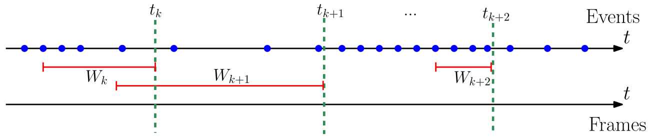

Fig. 2. Upon receiving a new frame from the standard camera at time t_k, we select a spatio-temporal window of events W_k, containing a fixed number of events ($N = 4$ in this example). Note that the temporal size of each window is automatically adapted to the event rate. Blue dots correspond to events, and the dashed green lines correspond to the times at which standard frames are received. The bounds of the spatio-temporal windows of events considered are marked in red.

first event whose timestamp $t_j < t_k$, and N is the window size parameter. Note that the duration of each window is inversely proportional to the event rate.

3) Synthesis of Motion-Compensated Event Frames: As in [13], we then collapse every spatio-temporal window of events to a synthetic event frame I_k by drawing each event on the image plane, after correcting for the motion of each event according to its individual timestamp.

Let $I_k(\mathbf{x}) = \sum_{e_i \in W_k} \delta(\mathbf{x} - \mathbf{x}'_i)$, where function $\delta(\mathbf{x})$ is the Kronecker delta, $\mathbf{x}'_i$ is the *corrected* event position, obtained by transferring event e_i to the reference event camera frame C_{0k}:

$$\mathbf{x}'_i = \pi_0(T_{t_k,t_i}(Z(\mathbf{x}_i)\pi_0^{-1}(\mathbf{x}_i))), \tag{1}$$

where $\mathbf{x}_i$ is the pixel location of event e_i, $\pi_0\,(.)$ the event camera projection model, obtained from prior intrinsic calibration, and T_{t_l,t_m} the incremental transformation between the camera poses at times t_l and t_m, obtained through integration of the inertial measurements (we refer the reader to [13] for details). $Z(\mathbf{x}_i)$ is the scene depth at time t_i and pixel $\mathbf{x}_i$, which we can estimate using 2D linear interpolation (on the image plane) of the landmarks reprojected on the current camera frame C_{0i}. In practice, as in [13], we observed that using the median depth of the landmarks in the field of view instead of linearly interpolating the depth gives satisfactory results at a lower computational cost. The quality of the motion compensation depends on the quality of the 3D landmarks available, therefore the quality of the event frames improves when also using the landmarks from the standard frames.

The number of events N in each spatio-temporal window is a parameter that needs to be adjusted depending on the amount of texture in the scene. As an example, for the quadrotor experiments presented in Section IV, we used $N = 20\,000$ events per frame.

4) Feature Tracking: We use the FAST corner detector to extract features [26], both on the virtual event frames, and the standard camera frames. Those features are then tracked independently across standard frames and event frames using the KLT tracker [27] (see Fig. 1). This yields two sets of independent features tracks $\left\{\mathbf{z}^{0,j,k}\right\}$, $\left\{\mathbf{z}^{1,j,k}\right\}$ (where j is the feature track index, and k is the frame index). For each sensor, each feature is treated as a *candidate* feature, and tracked over multiple frames. Once a feature can be triangulated reliably, the corresponding 3D landmark is triangulated through linear triangulation [28], and converted to a *persistent* feature which will be further tracked across the next frames. We re-detect features on each sensor as soon as the number of tracked features falls below a threshold. We used the same detection and tracking parameters for the motion-compensated event frames and for the standard frames. The FAST threshold we used was 50. We used a a pyramidal implementation of KLT with 2 pyramid levels, and a patch size of 24 × 24 pixels. Additionally, we used a bucketing grid (where each grid cell has size 32 × 32 pixels) to ensure that features are evenly distributed in each sensor's image plane.

5) Visual-Inertial Fusion Through Nonlinear Optimization: The visual-inertial localization and mapping problem is formulated as the joint optimization of a cost function that contains three terms: two weighted reprojection errors corresponding respectively to the observations from the event camera and the standard camera, plus an inertial error term $\mathbf{e}_s$:

$$J = \sum_{i=0}^{1}\sum_{k=1}^{K}\sum_{j\in\mathcal{J}(i,k)} {\mathbf{e}^{i,j,k}}^T \mathbf{W}_r^{i,j,k}\mathbf{e}^{i,j,k} + \sum_{k=1}^{K-1} {\mathbf{e}_s^k}^T \mathbf{W}_s^k \mathbf{e}_s^k$$

where i denotes the sensor index, k denotes the frame index, and j denotes the landmark index. The set $\mathcal{J}(i,k)$ contains the indices of landmarks maintained in the kth frame by sensor i. Additionally, $W_r^{i,j,k}$ is the information matrix of the landmark measurement $\mathbf{l}_{i,j}$, and W_s^k that of the kth IMU error. The reprojection error is:

$$\mathbf{e}_r^{i,j,k} = \mathbf{z}^{i,j,k} - \pi_i\left(\mathbf{T}_{C_i S}^k \mathbf{T}_{SW}^k \mathbf{l}^{i,j}\right)$$

where $\mathbf{z}^{i,j,k}$ is the measured image coordinate of the jth landmark on the ith sensor at the kth frame. We use standard IMU kinematics and biases model (see [3] for example) to predict the current state based on the previous state. Then, the IMU error terms are 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. For details, we refer the reader to [2].

The optimization is carried out not on all the frames observed but on a bounded set of frames composed of M keyframes (we use the same keyframe selection criterion as [13]), and a sliding window containing the last K frames. In between frames, the prediction for the sensor state is propagated using the IMU measurements. We employ the Google Ceres [29] optimizer to carry out the optimization.

Notice that with this formulation we avoid an explicit switching policy between standard and event camera: the optimization naturally uses the best sensing modalities available.

6) Additional Implementation Details:

a) Initialization: We assume that the sensor remains static during the initialization phase of the pipeline, during one or two seconds. We collect a set of inertial measurements and use them to estimate the initial attitude (pitch and roll) of the sensor, as well as to initialize the gyroscope and accelerometer biases.

b) No-motion prior for almost-still motions: When the sensor is still, no events are generated (except noise events). To handle this case in our pipeline, we add a strong zero velocity prior to the optimization problem whenever the event rate falls below a threshold, thus forcing the sensor to be still. We used a threshold

TABLE I
ACCURACY OF THE PROPOSED APPROACH USING FRAMES (FR), EVENTS (E) AND IMU (I), AGAINST USING EVENTS AND IMU, AND USING FRAMES AND IMU

Sequence	**Proposed (Fr + E + I)**		**E + I**		**Fr + I**	
	Mean Position Error (%)	Mean Yaw Error (deg/m)	Mean Position Error (%)	Mean Yaw Error (deg/m)	Mean Position Error (%)	Mean Yaw Error (deg/m)
boxes_6dof	**0.30**	**0.04**	0.44	0.05	0.30	0.06
boxes_translation	0.27	**0.02**	0.76	0.05	**0.17**	0.03
dynamic_6dof	**0.19**	0.10	0.38	**0.06**	0.62	0.10
dynamic_translation	**0.18**	**0.15**	0.59	0.16	0.67	0.26
hdr_boxes	**0.37**	**0.03**	0.67	0.09	0.78	0.17
hdr_poster	0.31	0.05	0.49	**0.04**	**0.28**	0.08
poster_6dof	**0.28**	**0.07**	0.30	0.08	0.59	0.11
poster_translation	**0.12**	0.04	0.15	**0.04**	0.23	0.08
shapes_6dof	**0.10**	**0.04**	0.48	0.06	0.17	0.05
shapes_translation	**0.26**	0.06	0.41	**0.04**	0.29	0.11

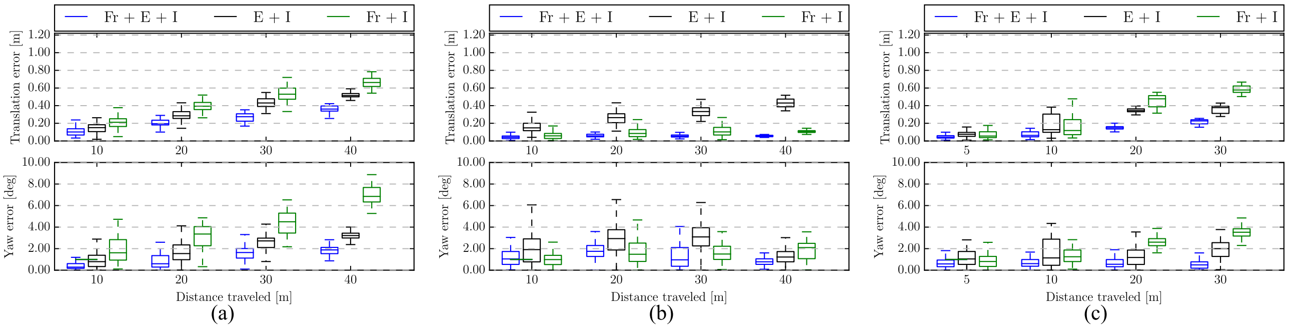

Fig. 3. Comparison of the proposed approach, using frames (Fr), events (E), and IMU (I), on three datasets from the Event Camera Dataset [14]. The graphs show the relative errors measured over different segments of the trajectory as proposed in [32]. Additional plots for all the datasets are provided in the supplementary material. (a) `hdr_boxes`, (b) `shapes_6dof`, (c) `dynamic_6dof`.

in the order of 10^3 events/s in our experiments, and measured the event rate using windows of 20 ms.

B. Evaluation

We evaluate the proposed pipeline quantitatively on the Event Camera Dataset [14], which features various scenes with ground truth tracking information. In particular, it contains extremely fast motions and scenes with very high dynamic range, recorded with the DAVIS[1] [30] sensor. As in [24], we only use the datasets from the Event Camera Dataset that are relevant for Visual-Inertial Odometry. Specifically, we exclude the rotational only datasets, as well as the datasets without inertial measurements.

The DAVIS sensor embeds a 240 × 180 pixels event camera with a 1 kHz IMU and also delivers standard frames at 24 Hz. Events, standard frames, and IMU measurements are synchronized on hardware. The IMU is delayed by a constant time offset in the order of 2.5 ms compared to the events and standard frames (because of the low-pass filter of the IMU). We estimated this delay using Kalibr [31].

To evaluate the results, the estimated and ground truth trajectories are aligned with a 6-DOF transformation in SE3, using 5 seconds of the trajectory (starting at second 3 and ending at second 8). Then, we compute the mean position error (Euclidean distance) and the yaw error as percentages of the total traveled distance. Due to the observability of the gravity direction, the error in pitch and roll is constant and comparable for each pipeline. Thus we omit them for compactness.

Table I shows the results obtained when running the pipeline in its proposed mode, using standard frames (Fr), events (E), and IMU (I). To further quantify the accuracy gained by using events and frames (plus IMU), compared to using only events or only frames (plus IMU), we run our proposed pipeline using the three different combinations, and report the results in Table I. Additionally, in Fig. 3, we use the relative error metrics proposed in [32], which evaluate the relative error by averaging the drift over trajectories of different lengths. Using jointly standard frames, events and IMU leads to an average position accuracy improvement of 85% compared to using frames and IMU only, and 130% against using events and IMU only. Notice that the Event Camera Dataset was made to showcase the situations where an event camera would be more useful. Nevertheless, datasets like boxes_translation and shapes_6dof show that using standard frames might still be advantageous compared to using only events, as can be seen in Table I and the detailed analysis in the supplementary material.

Table II provides a comparison between our approach and the state-of-the-art [13].

[1] https://inilabs.com/products

TABLE II
ACCURACY OF THE PROPOSED APPROACH USING FRAMES (FR), EVENTS (E) AND IMU (I), AGAINST [13], WHICH USES EVENTS AND IMU

Sequence	**Proposed (Fr + E + I)**		**State-of-the-art (E + I) [13]**	
	Mean Position Error (%)	Mean Yaw Error (deg/m)	Mean Position Error (%)	Mean Yaw Error (deg/m)
boxes_6dof	**0.30**	**0.04**	0.36	0.11
boxes_translation	**0.27**	**0.02**	0.31	0.08
dynamic_6dof	**0.19**	**0.10**	0.56	0.41
dynamic_translation	**0.18**	0.15	0.39	**0.06**
hdr_boxes	**0.37**	**0.03**	0.59	0.20
hdr_poster	**0.31**	**0.05**	0.33	0.19
poster_6dof	**0.28**	**0.07**	0.40	0.16
poster_translation	**0.12**	**0.04**	0.46	0.10
shapes_6dof	**0.10**	**0.04**	0.42	0.18
shapes_translation	**0.26**	**0.06**	0.50	0.13

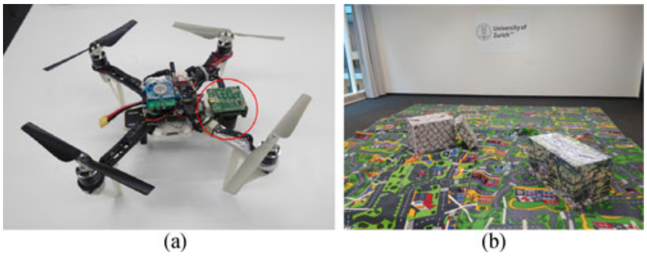

Fig. 4. Quadrotor platform used for the flight experiments, and preview of the flying room. (a) Quadrotor platform used for the flight experiments. (b) Preview of the room in which we conducted the flight experiments.

The events plus IMU pipeline in Table I is not the same as [13] in Table II; the former generates event frames at a fixed rate, while the latter generates them at a rate that depends on the event rate, we refer the reader to [13] for further details. Moreover, the parameters both pipelines share do not necessarily have the same values. These reasons account for the different results in Table I (E + I) and Table II (state-of-the-art E + I).

To the best of our knowledge, we are the first to report results on the Event Camera Dataset using all three sensor modalities. It can be seen that our approach, that uses frames and events, is better in terms of accuracy on almost all the datasets.

IV. QUADROTOR FLIGHT WITH AN EVENT CAMERA

In order to show the potential of our hybrid, frame-and-event–based pipeline in a real scenario, we ran our approach onboard an autonomous quadrotor and used it to fly autonomously in challenging conditions. We first start by describing in detail the quadrotor platform we built (hardware and software) in Section IV-A before turning to the specific in-flight experiments (Section IV-B).

A. Aerial Platform

1) Platform: We built our quadrotor from selected off-the-shelf components and custom 3D printed parts [see Fig. 4(a)]. Our quadrotor relies on a DJI frame, with RCTimer motors and AR drone propellers. The electronic parts of our quadrotor comprise a PX4FMU autopilot [33]. In addition, our quadrotor is equipped with an Odroid XU4 computer, which contains a 2.0 GHz quad-core processor running Ubuntu 14.04 and ROS [34]. Finally, a DAVIS 240 C sensor, equipped with a 70° field-of-view lens, is mounted on the front of the quadrotor, looking downwards. The sensor is connected to the Odroid computer via an USB 2.0 cable, and transmits events, standard frames, and inertial measurements, which we use to compute the state estimate on the Odroid using our proposed pipeline. Since the available ROS driver for the DAVIS did not come with an auto-exposure for the standard camera, we implemented an auto-exposure algorithm and made it available open-source for the community to use.[2] It is based on a simple proportional controller that controls the mean image intensity to a desired value (we used a value of 70 in our experiments).

2) Control: To follow reference trajectories and stabilize the quadrotor, we use the cascaded controllers presented in [35]. The high-level controller running on the Odroid includes a position controller and an attitude controller, while the low-level controller on the PX4 contains a body rate controller. The high-level controller takes a reference trajectory as input and computes desired body rates that are sent to the low-level controller. The low-level controller, in turn, computes the desired rotor thrusts using a feedback linearizing control scheme with the closed-loop dynamics of a first-order system. Details of the controllers can be found in [35].

B. Flight Experiments

We present three flight experiments that demonstrate that our system is able to fly a quadrotor in challenging conditions: (i) flying indoors while switching on and off the light (which is challenging because of the abrupt large change of illumination caused by the switching of the light and the very low light present in the room after the artificial light is turned off); (ii) while performing fast circles in a low-lit room; (iii) hovering at the same position (which is challenging for the event camera because close to no motion). In the first case, when the light is off, the standard frames are completely black. In the second one, the speed of the quadrotor induces severe motion blur on the standard frames. Nevertheless, in both cases, the events are left unaffected and our pipeline is able to successfully exploit them to provide robust state estimation. In the third case, instead, there is almost no motion, which makes it difficult for the event camera to track reliable features. Nevertheless, the frames are left unaffected and our pipeline is able to successfully exploit them to provide robust state estimation.

These three experiments are best appreciated in video attachment.[3]

1) Switching the Light Off and On, in Flight: In this experiment, we pushed our pipeline to the limit by outright switching the room light off while autonomously flying in circles. The only remaining light was residual light coming from the windows (very little light, but still enough for the event camera to work). The standard frames become completely black when the light goes off [top frame in Fig. 5(a)], making them useless for

[2] Available in the DAVIS ROS driver: https://github.com/uzh-rpg/rpg_dvs_ros

[3] http://rpg.ifi.uzh.ch/ultimateslam.html

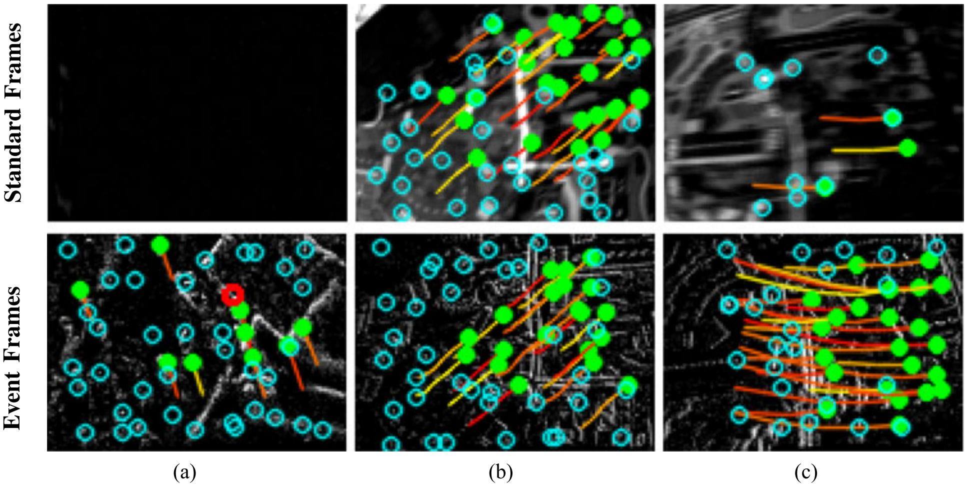

Fig. 5. Example feature tracks in various conditions, on the standard frames (top row) and the virtual event frames (bottom row). Every column corresponds to the same timestamp, a frame from the top row has a corresponding event frame on the bottom row. The green solid dots are persistent features, and the blue dots correspond to candidate features. The tracks are shown as colored lines. (a) Low-light. (b) Good lighting, moderate speed. (c) Motion blur.

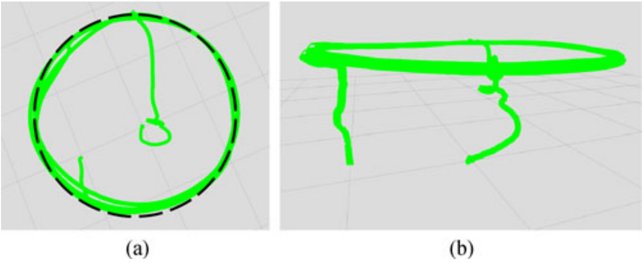

Fig. 6. Experiment 1: Switching the light off and on. The trajectory estimated by our pipeline is the green line. The commanded trajectory is the superimposed black dashed line. (a) Top view. (b) Perspective view.

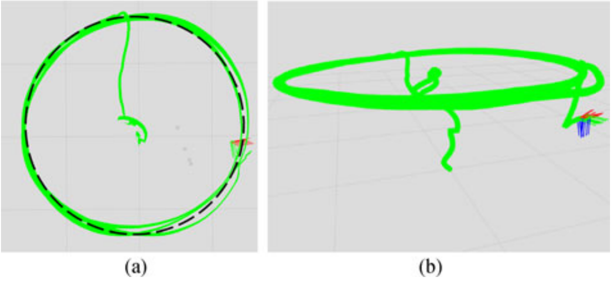

Fig. 7. Experiment 2: Fast circles in a low-lit room. The trajectory estimated by our pipeline is the green line. The commanded trajectory is the superimposed black dashed line. (a) Top view. (b) Perspective view.

state estimation. By contrast, the events still carry enough information (albeit noisier) to allow reasonable feature tracks [bottom frame Fig. 5(a)]. Switching the light off effectively forces the pipeline to rely only on events and inertial measurements. Note that the abrupt illumination change caused by switching the lights on and off makes almost every pixel fire events. Although we do not explicitly handle this particular case, in practice we observed no substantial decrease in accuracy when this occurs as features are quickly re-initialized.

The trajectory flown by the quadrotor is shown in Fig. 6.

2) Fast Circles in a Low-Lit Room: In this experiment, the quadrotor autonomously flies a circular trajectory with increasing speed in a closed room with little light (see Fig. 1); we carried this experiment during the night and set a low lighting in the room. The circular trajectory commanded to the quadrotor is parametrized by its radius and the desired angular velocity. We set the angular velocity to 1.4 rad/s on a circle of 1.2 m radius, corresponding to a top linear velocity of 1.68 m/s. The circle height was 1.0 m. At this speed and height, the optical flow generated on the image plane amounts to approximately 340 pixels/s.

While the speed remains moderate at the beginning of the trajectory (below 1.2 m/s), standard frames do not suffer from motion blur and our pipeline indeed tracks features in both the standard frames and the event frames (cf. top and bottom frames in Fig. 5(b), respectively). Nevertheless, as soon as the speed increases, the standard frames start to suffer from severe motion blur, as shown in the top frame of Fig. 5(c), and the number of features tracked in the standard frames significantly decreases. Conversely, the events allow synthesizing motion-free virtual event frames, which, in turn, allow keeping reliable feature tracks [bottom frame in Fig. 5(c)].

In Fig. 7, both the desired and estimated trajectories are shown for comparison. Interestingly, the right side of the trajectory is slightly noisier than the left side. This turns out to match well with the light configuration in the room: the left side of the room was indeed more illuminated than the right side (visible in Fig. 1). This is coherent with the quantitative experiments presented in Section III-B: the increase of the quality of the standard frames on the room side with more light correlates directly to an increase of accuracy of the pipeline.

3) Hovering: We also provide qualitative experiments to show how our pipeline performs close to no-motion conditions, typically encountered when a drone is hovering.

First, we command the drone to hover while using the events-only pipeline and observe that the state estimate drifts. We then command the drone to hover while using both the images and the event frames, and observe that the drone successfully keeps its position with no noticeable drift.

The difference lies in that features are tracked successfully on the standard frames, while they are lost on the event frames. This is because, unlike standard cameras, the appearance of the features tracked by the event camera may change drastically with the direction of the motion, which is exactly what happens when hovering: vibrations induce frequent changes of motion direction, which reduce the length of the feature tracks from the event streams, leading to increased drift.

V. Conclusion

We introduced the first hybrid pipeline that fuses events, standard frames, and inertial measurements to yield robust and accurate state estimation. We also reported results using these three sensing modalities on the Event Camera Dataset [14] and demonstrated an accuracy boost of 130% compared to using only events plus IMU, and a boost of 85% compared to using only standard frames plus IMU. Furthermore, we successfully integrated the proposed pipeline for state estimation onboard a computationally-constrained quadrotor and used it to realize, to the best of our knowledge, the first closed-loop flight of a quadrotor using an event camera. Finally, in a set of specific experiments, we showed that our hybrid pipeline is able to leverage the properties of the standard camera and the event camera to provide robust tracking when flying in multiple conditions, such as hovering, flying in fast circles or flying in a low-lit room.

References

[1] A. I. Mourikis and S. I. Roumeliotis, “A multi-state constraint Kalman filter for vision-aided inertial navigation,” in *Proc. IEEE Int. Conf. Robot. Autom.*, Apr. 2007, pp. 3565–3572.

[2] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale, “Keyframe-based visual-inertial SLAM using nonlinear optimization,” *Int. J. Robot. Res.*, vol. 34, pp. 314–334, 2015.

[3] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, “On-manifold preintegration for real-time visual-inertial odometry,” *IEEE Trans. Robot.*, vol. 33, no. 1, pp. 1–21, Feb. 2017.

[4] 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, Feb. 2008.

[5] M. Cook, L. Gugelmann, F. Jug, C. Krautz, and A. Steger, “Interacting maps for fast visual interpretation,” in *Proc. Int. Joint Conf. Neural Netw.*, 2011, pp. 770–776.

[6] P. Bardow, A. J. Davison, and S. Leutenegger, “Simultaneous optical flow and intensity estimation from an event camera,” in *Proc. IEEE Int. Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 884–892.

[7] C. Reinbacher, G. Graber, and T. Pock, “Real-time intensity-image reconstruction for event cameras using manifold regularisation,” in *Proc. Brit. Mach. Vis. Conf.*, 2016.

[8] 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, Apr. 2017.

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

[10] B. J. P. Hordijk, K. Y. W. Scheper, and G. C. H. E. de Croon, “Vertical landing for micro air vehicles using event-based optical flow,” arXiv:abs/1702.00061, 2017.

[11] “Snapdragon flight,” Jan. 23, 2018. [Online]. Available: https://developers.google.com/tango/

[12] Google, Mountain View, CA, USA, “Project tango,” Jan. 23, 2018. [Online]. Available: https://www.google.com/atap/projecttango/

[13] H. Rebecq, T. Horstschäfer, and D. Scaramuzza, “Real-time visual-inertial odometry for event cameras using keyframe-based nonlinear optimization,” in *Proc. Brit. Mach. Vis. Conf.*, Sep. 2017.

[14] 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. Res.*, vol. 36, pp. 142–149, 2017.

[15] E. Jones and S. Soatto, “Visual-inertial navigation, mapping and localization: A scalable real-time causal approach,” *Int. J. Robot. Res.*, vol. 30, no. 4, pp. 407-430, Apr. 2011.

[16] H. Kim, A. Handa, R. Benosman, S.-H. Ieng, and A. J. Davison, “Simultaneous mosaicing and tracking with an event camera,” in *Proc. Brit. Mach. Vis. Conf.*, 2014.

[17] G. Gallego and D. Scaramuzza, “Accurate angular velocity estimation with an event camera,” *IEEE Robot. Autom. Lett.*, vol. 2, pp. 632–639, Apr. 2017.

[18] C. Reinbacher, G. Munda, and T. Pock, “Real-time panoramic tracking for event cameras,” in *Proc. IEEE Int. Conf. Comput. Photography*, 2017, pp. 1–9.

[19] D. Weikersdorfer, R. Hoffmann, and J. Conradt, “Simultaneous localization and mapping for event-based vision systems,” in *Proc. Int. Conf. Comput. Vis. Syst.*, 2013, pp. 133–142.

[20] D. Weikersdorfer, D. B. Adrian, D. Cremers, and J. Conradt, “Event-based 3D SLAM with a depth-augmented dynamic vision sensor,” in *Proc. IEEE Int. Conf. Robot. Autom.*, Jun. 2014, pp. 359–364.

[21] A. Censi and D. Scaramuzza, “Low-latency event-based visual odometry,” in *Proc. IEEE Int. Conf. Robot. Autom.*, 2014, pp. 703–710.

[22] B. Kueng, E. Mueggler, G. Gallego, and D. Scaramuzza, “Low-latency visual odometry using event-based feature tracks,” in *Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, Daejeon, South Korea, Oct. 2016, pp. 16–23.

[23] E. Mueggler, G. Gallego, H. Rebecq, and D. Scaramuzza, “Continuous-time visual-inertial trajectory estimation with event cameras,” arXiv:1702.07389, 2017.

[24] A. Zhu, N. Atanasov, and K. Daniilidis, “Event-based visual inertial odometry,” in *Proc. IEEE Int. Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 5816–5824.

[25] E. Mueggler, B. Huber, and D. Scaramuzza, “Event-based, 6-DOF pose tracking for high-speed maneuvers,” in *Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, 2014, pp. 2761–2768.

[26] E. Rosten and T. Drummond, “Machine learning for high-speed corner detection,” in *Proc. Eur. Conf. Comput. Vis.*, 2006, pp. 430–443.

[27] B. D. Lucas and T. Kanade, “An iterative image registration technique with an application to stereo vision,” in *Proc. Int. Joint Conf. Artif. Intell.*, 1981, pp. 121–130.

[28] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge, U.K.: Cambridge Univ. Press, 2003.

[29] S. Agarwal *et al.*, “Ceres solver,” Jan. 23, 2018. [Online]. Available: http://ceres-solver.org

[30] C. Brandli, R. Berner, M. Yang, S.-C. Liu, and T. Delbruck, “A 240 × 180 130 dB 3us latency global shutter spatiotemporal vision sensor,” *IEEE J. Solid-State Circuits*, vol. 49, no. 10, pp. 2333–2341, Oct. 2014.

[31] P. Furgale, J. Rehder, and R. Siegwart, “Unified temporal and spatial calibration for multi-sensor systems,” in *Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, 2013, pp. 1280–1286.

[32] 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 Recognit.*, 2012, pp. 3354–3361.

[33] L. Meier, P. Tanskanen, L. Heng, G. H. Lee, F. Fraundorfer, and M. Pollefeys, “PIXHAWK: A micro aerial vehicle design for autonomous flight using onboard computer vision,” *Auton. Robots*, vol. 33, no. 1/2, pp. 21–39, 2012.

[34] M. Quigley *et al.*, “ROS: An open-source robot operating system,” in *Proc. ICRA Workshop Open Source Softw.*, vol. 3, 2009.

[35] M. Faessler, F. Fontana, C. Forster, E. Mueggler, M. Pizzoli, and D. Scaramuzza, “Autonomous, vision-based flight and live dense 3D mapping with a quadrotor MAV,” *J. Field Robot.*, vol. 33, no. 4, pp. 431–450, 2016.