Asynchronous Optimisation for Event-based Visual Odometry

Daqi Liu[†], Alvaro Parra[†], Yasir Latif[†], Bo Chen[†], Tat-Jun Chin[†,‡], Ian Reid[†]

Abstract—Event cameras open up new possibilities for robotic perception due to their low latency and high dynamic range. On the other hand, developing effective event-based vision algorithms that fully exploit the beneficial properties of event cameras remains work in progress. In this paper, we focus on event-based visual odometry (VO). While existing event-driven VO pipelines have adopted continuous-time representations to asynchronously process event data, they either assume a known map, restrict the camera to planar trajectories, or integrate other sensors into the system. Towards map-free event-only monocular VO in $SE(3)$, we propose an asynchronous structure-from-motion optimisation back-end. Our formulation is underpinned by a principled joint optimisation problem involving *non-parametric* Gaussian Process motion modelling and incremental *maximum a posteriori* inference. A high-performance incremental computation engine is employed to reason about the camera trajectory with every incoming event. We demonstrate the robustness of our asynchronous back-end in comparison to frame-based methods which depend on accurate temporal accumulation of measurements.

I. INTRODUCTION

Many robotic perception capabilities are developed for regular cameras, which are affected by blurring or lack of contrast due to high-speed manoeuvres or low-light settings. By asynchronously detecting intensity changes, event cameras have low latency and high dynamic range, which can help mitigate the above issues. However, the very different sensing principle calls for novel processing methods [1].
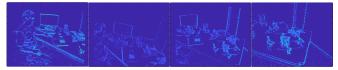
In this paper, we focus on event-based monocular visual odometry (VO). An event camera outputs a time-continuous stream of events $\mathcal{E} = \{e_1, e_2, \dots\}$, where each event

$$e_k = (z_k, t_k, p_k) \tag{1}$$

is an observation of a change in intensity at pixel $z_k \in \mathbb{R}^2$ at time $t_k \in \mathbb{R}_+$ with polarity $p_k \in \{-1, +1\}$. The goal of VO is to recover the motion of the camera that gave rise to $\mathcal{E}$.

A baseline approach is to temporally accumulate events to form event images (a.k.a. *batching*) then apply conventional frame-based VO methods, e.g., [3], [4]. However, correct batching is critically important for frame-based methods. The optimal batching parameter (e.g., batch size or duration) is a time-varying quantity that depends on the instantaneous motion and scene complexity [5], [6]. Arguably, finding the optimal batching parameter (in real-time) is a non-trivial problem itself [5], [6]. Fig. 1b shows an inaccurate VO result by a frame-based method due to suboptimal batching.

Fundamentally, batching of events leads to a loss of temporal and spatial information, thus lowering signal-to-noise ratio. An asynchronous approach for VO that

[†]School of Computer Science, The University of Adelaide.
[‡]SmartSat CRC Professorial Chair of Sentient Satellites.

(a) Event images to visualise the event data used [2].

(b) Frame-based method with suboptimal batching.

(c) Asynchronous optimisation that does not require batching.

Fig. 1: (a) Input data for event-based VO. (b) The frame-based method is accurate due to suboptimal batching. Note that we are not claiming it is difficult to set the batching parameter in this specific event data; rather, we are showing the negative effects on frame-based VO *given* suboptimal batching. (c) By asynchronously processing the data using our CT SfM formulation (no batching required), our event-driven approach can robustly track the camera in $SE(3)$.

- (P1) Fits the trajectory using all *individual* events e_k; and
- (P2) Updates the motion at each newly *incoming* event

will help alleviate the problems due to batching. Fig. 2 conceptually contrasts batching and asynchronous processing.

A promising idea to realise asynchronous event-driven VO is to estimate a continuous-time (CT) camera trajectory from $\mathcal{E}$. This implicitly allows the camera pose to be temporally interpolated and extrapolated, which in turn provides a basis to perform P1 and P2. Along the lines above, there have been efforts on event-driven VO [7], [8], [9], [10]. However, previous works either assume a known map [7], [8], restrict the camera to planar motions [9], or incorporate an IMU with preintegration [11] thereby achieving event-inertial VO [10].

Towards *event-only* and *map-free* monocular VO in $SE(3)$, we propose an asynchronous structure-from-motion (SfM) back-end optimisation that satisfies P1 and P2 above. Our formulation conducts incremental *maximum a posteriori* (MAP) optimisation [12], [13] to achieve CT trajectory estimation and mapping [14]. An $SE(3)$ trajectory is modelled using a Gaussian Process (GP), which allows the camera

(a) Batching for frame-based optimisation   (b) Event-driven optimisation   (c) Event-driven and tracking opt.
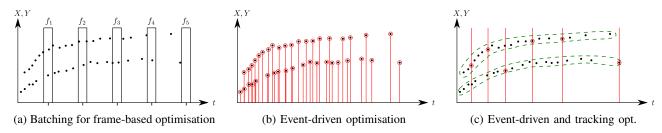
Fig. 2: Contrasting batching and asynchronous processing for event-only VO. Black dots indicate an event stream unevenly distributed in the image plane XY and time t. (a) Batching accumulates events in time windows, i.e., batches (represented by rectangular columns), to form event images $f_1, f_2, \ldots$, which are then subject to frame-based optimisation for VO. For correct operation, the batching parameters (e.g., location, duration and/or size of batches) must correctly reflect the underlying motion and scene structure. (b) Ideal event-driven optimisation integrates temporal and spatial information from each asynchronous event; the diagram stresses the event-driven sampling with a red decorator. (c) Event-driven optimisation with asynchronous event tracks (green dashed lines) increases efficiency by reducing optimisation instances (red lines).

poses at all $e_1, e_2, \ldots$ and the pertinent scene structure to be robustly reasoned from noisy event data. Moreover, the estimated CT trajectory can be extrapolated to allow motion updates at each new event. Fig. 1 illustrates the benefits of our approach, while Sec. VI will present further empirical results. Our work also represents one of the first applications of [12], [13], [14] to event-based VO.

II. RELATED WORK

A. Frame-based optimisation with batching

Early event-based VO methods conduct batching to generate event images which are then fed into frame-based pipelines [15], [16], [17], [18], [19]. Images are usually used to represent the event information for batching techniques, which can further be divided into two groups. The first group typically produces *motion-compensated images* [15], [16], [20], [21], [22] by assuming some motion model [22] or relying on IMU measurements [15], [16], while the second group produces *time-surface maps* [23], [18], [19] by the temporal component of the event. Most of the above methods use a fixed temporal resolution [17] or spatial resolution [15], [16], [21] to conduct batching, specified respectively by the time duration of a batch and the number of events in a batch.

Batching based on fixed temporal or spatial resolutions are a source of failure in frame-based VO. The example data stream in Fig. 2 corresponds to a decelerating camera in a static scene, which causes more events at the beginning of the stream and fewer events at the end. In this scenario, batching using a constant temporal resolution (the approach depicted in Fig. 2a) will eventually lead to images with insufficient information (e.g., no events in f_5) for motion estimation. On the other hand, batching based on constant spatial resolution in the decelerating camera scenario above will lead to noisy ("blurry") event images, since a batch during the slow motion period is no longer a good representation of the instantaneous motion. Moreover, the optimal spatial resolution also depends on the scene complexity in the camera FOV [24], which can change rapidly during the VO process.

In principle, a dynamic batching strategy [25] can alleviate the issues above. However, accurately estimating batching parameters is difficult; perhaps as difficult as performing VO itself since the optimal batching parameters depend on knowledge of the instantaneous camera motion.

B. Event-driven optimisation

Event-driven optimisation for VO (see Fig. 2b) should alleviate the weaknesses of batching. Fundamentally, however, a single event does not provide enough information to compute/update the camera motion, hence a motion model needs to be maintained by the algorithm [8] to facilitate asynchronous processing. Previous methods have mainly utilised non-parametric motion models [7], [8], [9], [10].

Mueggler et al. [7] pioneered a CT framework for event-based VO with known map and later extended with IMU fusion [8]. Their method models the $SE(3)$ camera trajectory by using cubic splines. However, their approach will not work in unknown environments or when IMU is not available. The event-inertial approach of Le Gentil et al. [10] jointly optimises scene features (3D lines in [10]) and the camera motions, based on GP motion model. However, their paradigm based on IMU preintegration [11] (regressing IMU pose and velocity from IMU readings) is fundamentally different from CT trajectory estimation. In [10], the role of GP is on preintegrating IMU and not on trajectory interpolation, and the resulting trajectory is a discrete set of camera poses and velocities. Closest in spirit to ours is [9] which is based on volumetric contrast maximisation to perform CT trajectory estimation (using splines) and jointly solving for camera tracking and (dense) mapping for event-only VO. However, only planar trajectories were considered.

As alluded to in Sec. I, our core contribution is to formulate an asynchronous back-end optimisation that can support map-free event-based VO in $SE(3)$. Our back-end is based on incremental MAP optimisation for CT trajectory estimation, and mapping [12], [13], [14], where a GP is formulated into CT $SE(3)$ trajectory and GTSAM [26] is further incorporated for incremental optimisation. Key to our formulation is a tight coupling between GP trajectory modelling and MAP inference to solve data association and outlier (spurious event) removal. The incremental MAP

estimation supports per-event updates (Fig. 2b) as well as delayed updates (Fig. 2c) to improve efficiency if an event-driven feature tracker (e.g., [27]) is available, without batching.

III. EVENT-DRIVEN VO PIPELINE

Our event-driven VO pipeline is shown in Fig. 3. The input to the pipeline is an event stream $\mathcal{E}$, and the output of the pipeline is a CT camera trajectory in $SE(3)$ and a local map which is a by-product. The camera trajectory is represented by a GP motion model. At the core of the pipeline—outlined by the blue box in the diagram, which is also the focus of our paper—is a MAP incremental SfM optimisation routine.

In the rest of the section, we will justify the design of the pipeline. Details of the core optimisation routine will be provided in Sec. IV, while some details of our implementation of the pipeline will be given in Sec. V.

A. GP representation for CT camera trajectory

Let $q(t) \in SE(3)$ be a camera pose at time t. Our VO method employs the state representation

$$x(t) := (q(t), \dot{q}(t)) \tag{2}$$

and models the event camera trajectory as a GP

$$x(t) \sim \mathcal{GP}(\mu(t), \mathcal{K}(t, t')) \tag{3}$$

with mean $\mu(t)$ and covariance (kernel function) $\mathcal{K}(t, t')$. We would like to fit (3) onto $\mathcal{E}$; in practice, this is equivalent to being able to compute the maximum of the posterior

$$p(x(\tau) \mid \{x(t_i)\}_{i=1}^{M}, \mathcal{E}), \tag{4}$$

for any desired time τ, where

$$\{x(t_i)\}_{i=1}^{M} \tag{5}$$

is a set of M *control states*; these can be the states associated to a subset of the events (the red circles in Fig. 2c), or the states of all $\mathcal{E}$ (the scenario depicted in Fig. 2b). Maximising (4) is called *GP regression*. Secs. IV and V will describe GP regression and control state selection respectively.

Important characteristics of the GP representation are:

a) Compactness: The appropriate number of control states depends on the complexity of the motion. In segments of the trajectory with uniform motions, the control states can be significantly fewer than event data. Arguably a discrete-time trajectory representation (e.g., camera poses defined over frames obtained from batching) is also more "compact" than the raw event data, however, it does not facilitate asynchronous processing, as explained next.

b) Asynchronous processing: The ability to perform GP regression at any τ permits to seamlessly incorporate event data into the estimation of the control points, i.e., optimising over asynchronous measurements. On the other hand, a discrete-time trajectory representation, which does not allow smooth interpolation, will require a minimum amount of co-visibility (visual overlapping of the observations) between adjacent states to prevent a "break" in the trajectory.

c) Constant-time regression: The benefits above arguably exist in other CT trajectory representations for event-based VO (e.g., splines [7], [8]). However, a fundamental benefit of GP is constant-time regression, i.e., the cost of finding the maximum of (4) is independent of the number of control states M. This is enabled by the Markovian structure of the state representation (2) (assuming a white-noise-on-acceleration prior [14]), which leads to a class of GPs with exactly sparse inverse kernel matrix [14].

B. Event-based MAP incremental SfM

The goal of the back-end is to perform the computations to fit the GP model (3) onto the event stream $\mathcal{E}$ by estimating a set of control states. The problem translates to MAP inference of the control states from $\mathcal{E}$. Imperative for a VO system is to extend the trajectory in an online fashion, and to do so efficiently. For our GP formulation, this translates into incorporating new control states into the trajectory without solving the MAP problem from scratch. We call this problem MAP *incremental SfM* (details in Sec. IV).

MAP incremental SfM has been explored previously, but *not* for event-driven VO. To address planar trajectories from wheel and laser odometry, Yan et al. [28] presented an *incremental* version of the approach in [14]. The key component for incremental MAP was using a Bayes tree [29] (as available in GTSAM [26]) to perform incremental variable reordering and just-in-time relinearisation operations during incremental MAP optimisation. Other works [12] have followed the same strategy: Anderson et al. [30] solves GP trajectory regression in $SE(3)$ and Dong et al. [12] extended the approach to general matrix Lie groups. We based our $SE(3)$ implementation in the code of [12].

C. Asynchronous back-end integration

Our MAP incremental SfM back-end enables the desired properties of an event-driven VO (P1 and P2 in Sec. I) to be achieved. There are also other appealing properties that support event-only VO pipeline (Fig. 3):

- The estimated CT trajectory can facilitate tracking events in the front-end. For example, a GP trajectory extrapolation can guide local search for event data association.
- The outlier removal component can remove inconsistent observations assisted by GP constant-time interpolation.
- Under the unified probabilistic framework of GP and MAP SfM, the back-end estimates the distribution of the trajectory. Thus, front-end components can introduce risk factors in their strategies.

IV. CONTINUOUS-TIME SfM FORMULATION FOR EVENT-ONLY VO

In this section, we provide the details of the core optimisation routine in the proposed event-drive VO pipeline (Fig. 3).

A. Measurements

Let $\mathcal{E} = \{e_k\}_{k=1}^{N}$ be the event data observed thus far. Let z_k to be the pixel coordinates of an event at time t_k. Under
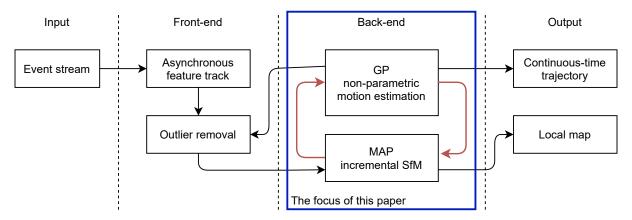
Fig. 3: Event-driven VO pipeline. The front-end receives a stream of asynchronous observations. The feature track component of the front-end solves data association. Then an outlier removal component filters out inconsistent measurements. The asynchronous back-end uses all spatial and temporal event information. The GP and MAP components collaborate to produce a continuous-time trajectory that can be updated from a single event. The back-end produces a local map as a sub-product.

a probabilistic Gaussian framework,

$$z_k = h_k(x(t_k)) + \epsilon_k, \ \forall k = 1, \dots, N \quad (6)$$

where h_k is the pinhole projection function (our measurement model) of the observed scene point and $\epsilon_k \sim \mathcal{N}(0, \mathcal{R}_k)$ is zero-mean Gaussian noise with covariance $\mathcal{R}_k$.

We group all observed events into the measurement vector

$$Z := \begin{bmatrix} z_1^T & \cdots & z_N^T \end{bmatrix}^T \sim \mathcal{N}(0, \mathcal{R}), \quad (7)$$

where $\mathcal{R}$ accommodates all covariances $\mathcal{R}_k$, $k = 1, \dots, N$.

B. Variables

Our GP formulation estimates a set of control states for the trajectory and a set of 3D landmarks. Define

$$\mathcal{X} := \{(x_i, s_i)\}_{i=1}^M \quad (8)$$

as a set of M control states $x_j := x(s_j)$ in correspondence with a subset of event timestamps $\{s_j\}_{j=1}^M \subseteq \{t_k\}_{k=1}^N$ (usually $M \ll N$). Fundamentally, events correspond to brightness changes at 3D scene points; here, we assume that the control states have been associated to a set of 3D landmarks $\begin{bmatrix} \ell_1^T & \cdots & \ell_L^T \end{bmatrix}^T$ (details of data association will be given in Sec. V). Under GP modelling, the (unknown) landmarks are random variables

$$\begin{bmatrix} \ell_1^T & \cdots & \ell_L^T \end{bmatrix}^T \sim \mathcal{N}(\lambda, \mathcal{L}) \quad (9)$$

drawn from a multivariate Gaussian. We group all variables in the vector

$$X := \begin{bmatrix} x_1^T & \cdots & x_M^T & \ell_1^T & \cdots & \ell_L^T \end{bmatrix}^T \sim \mathcal{N}(\eta, \mathcal{P}), \quad (10)$$

where

$$\eta := \begin{bmatrix} \mu(s_1)^T & \cdots & \mu(s_M)^T & \lambda^T \end{bmatrix}^T \quad (11)$$

and

$$\mathcal{P} := \begin{bmatrix} \mathcal{K} & \\ & \mathcal{L} \end{bmatrix}. \quad (12)$$

In the following, for simplicity we use the notation x_i for the i-th element of X; the reader should take note that x_i as an item of X could refer to a landmark variable.

C. MAP inference

The Gaussian distribution $\mathcal{N}(\eta, \mathcal{P})$ defines the prior distribution in the MAP formulation. Our aim is find the X that maximises the posterior density given the measurements Z:

$$\hat{X} = \operatorname*{argmax}_X p(X|Z) \quad (13a)$$

$$\propto \operatorname*{argmax}_X p(Z|X) \, p(X). \quad (13b)$$

Under Gaussian noise assumption, the likelihood

$$p(Z|X) \propto \prod_k \exp \left\{ -\frac{1}{2} \|h_k(x(t_k)) - z_k\|_{\mathcal{R}_k}^2 \right\} \quad (14)$$

and the prior distribution

$$p(X) \propto \prod_i \exp \left\{ -\frac{1}{2} \|x_i - \eta_i\|_{\mathcal{P}_i}^2 \right\} \quad (15)$$

are well characterised. However, to conduct MAP optimisation, we need to write the likelihood (14) in terms of the variables $\{x_i\}$. By using the GP representation for a Markovian trajectory, $x(t_k)$ can be written in terms of variables x_l, x_r, such that $(x_l, s_l), (x_r, s_r) \in \mathcal{X}$, and $s_l < t_k < s_r$ are the tightest variable timestamps to the event observation. We stress that the likelihood (14) incorporates all N events into the MAP estimator (13b) of the combined variables.

D. Trajectory Interpolation

The GPs for Markovian trajectories resulting from LTV-SDE produce the $O(1)$ interpolation [14]

$$x(t_k) = \mu(t_k) + \Lambda(t_k)(x_l - \mu(s_l)) + \Psi(t_k)(x_r - \mu(s_r)), \quad (16)$$

where

$$\mu(t_k) := \Phi(t_k, s_l) \, \mu(s_l) \quad (17)$$

and the matrix operators Λ, Ψ and Φ are as defined in [14]. As a linear combination of two trajectory states (x_l, x_r), GP interpolation is the key to efficiently introduce the observations into the CT SfM formulation.

However, (16) assumes trajectory states as elements of a vector space, which is invalid for trajectories in $SE(3)$. Dong et al. [12] addressed this limitation by extending this special class of GPs to matrix Lie groups [31]. The state $x(t)$ of Markovian trajectory (2) in $SE(3)$, becomes

$$\gamma(t; t_i) := (\xi(t), \dot{\xi}(t)), \tag{18}$$

in its Lie algebra $\mathfrak{se}(3)$, where $[\xi(t)]_\times$ is an element of $\mathfrak{se}(3)$ coinciding with the local tangent space of $SE(3)$. Precisely, for a transformation (a pose) $T(t) \in SE(3)$ around a linearisation point $T_i \in SE(3)$,

$$[\xi(t)]_\times := T_i^{-1} T(t). \tag{19}$$

Thus, GP interpolation is also $O(1)$ in the tangent space

$$\gamma(t_k; s_l) = \Lambda(t_k)\gamma(s_l; s_l) + \Psi(t_k)\gamma(s_r; s_l). \tag{20}$$

See [12], [13] for more details.

E. Solving the CT MAP problem

Gauss-Newton (GN) method is typically used to solve MAP optimisation problems with non-linear measurement models. In the context of (13b), GN is applied to compute the locally optimal displacement δX for the state update

$$X + \delta X \rightarrow X \tag{21}$$

by solving

$$\min_{\delta X} \frac{1}{2}\|J\delta X - \Delta Z\|_{\mathcal{R}}^2 + \frac{1}{2}\|\delta X - \Delta X\|_{\mathcal{P}}^2, \tag{22}$$

where

$$J := \frac{\partial h}{\partial X} \tag{23}$$

is the Jacobian matrix with $h := [h_1^T, \cdots, h_N^T]^T$,

$$\Delta X := X - \eta \tag{24}$$

is the *deviation from the mean*, and

$$\Delta Z := Z - h \tag{25}$$

is the *prediction error*. For brevity, we define the update (21) as a vector addition; however, this update is really carried out over a Lie manifold through the exponential map [31]. The least-squares problem (22) has the normal form

$$(\mathcal{P}^{-1} + J^T\mathcal{R}^{-1}J)\delta X^* = \mathcal{P}^{-1}\Delta X + J^T\mathcal{R}^{-1}\Delta Z, \tag{26}$$

which allows δX^* to be obtained using linear solvers.

F. Sparsity

Crucial to efficiently find the update δX^* from (26) is the sparsity of the *information matrix*

$$A := \mathcal{P}^{-1} + J^T\mathcal{R}^{-1}J. \tag{27}$$

Since A is symmetric, a recurrent strategy is to find the Cholesky factorisation $A = B^T B$ (B is lower triangular) and then solve for δX^* through back-substitution. The complexity of solving the Cholesky factorisation is affected by the sparsity of $A \in \mathbb{R}^{n \times n}$: from $O(n^3)$ for dense matrices to $O(n^1)$ for narrowly banded matrices [32].

The $J^T\mathcal{R}^{-1}J$ component in A is the information from the (event) measurements. This matrix reflects the sparse structure of the SLAM problems [33].

$$\mathcal{P}^{-1} = \begin{bmatrix} \mathcal{K}^{-1} & \\ & \mathcal{L}^{-1} \end{bmatrix} \tag{28}$$

encodes the prior information. It is also sparse as $\mathcal{K}^{-1}$ is sparse for Markovian trajectories [14] and $\mathcal{L}$ is diagonal.

V. IMPLEMENTATION

Algorithm 1 presents the main steps of our implementation of the event-driven VO pipeline introduced in Sec. III. An important problem is solving for data association. Our implementation uses the asynchronous method of [27] (Lines 5 and 18) to track event features (Lines 4 and 16) obtained from a simple heuristic based on histogramming events.

Algorithm 1 Asynchronous VO

1: Set a list of control states $\mathcal{X} = \{\}$.
2: Listen for an initial event $e_0 = (z_0, t_0, p_0)$.
3: Add a new control state (x_0, t_0) to $\mathcal{X}$, where x_0 encodes the initial pose and velocity at time t_0.
4: Listen for more events to track the selected features.
5: Track the selected features; for example, with [27].
6: **while** Tracking **do**
7:   **if** there is only one state in $\mathcal{X}$ **then**
8:     Add (x_1, t_c) to $\mathcal{X}$ such that x_1 encodes the rel. pose between matching keypoints at t_0 and current t_c.
9:     Estimate the initial map coordinates by triangulating matching keypoints.
10:     Refine the estimates by solving the CT SfM to reason over all events within the event tracks.
11:   **end if**
12:   Add a new (x_{new}, t_c) to $\mathcal{X}$ from GP extrapolation at current time t_c.
13:   Remove outlying tracks as those with average reprojection error (over all its events) above some threshold.
14:   Optimise variables with our asynchronous back-end.
15:   **if** number of event tracks is less than a threshold **then**
16:     Find more feature tracks as in Line 4.
17:   **end if**
18:   Continue tracking the selected features.
19: **end while**

VI. RESULTS
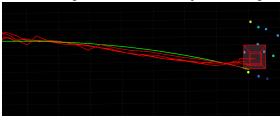
A. Evaluation of core VO optimisation

First, we focus on evaluating the core optimisation routine (Sec. IV) using data captured under controlled settings. We recorded a 30-second event sequence (containing $\approx$10M events) from a simple scene with several dots with a Prophesee event camera mounted in an UR5 robot arm; see Fig. 4a for the setup. The simple scene enabled the usage of STR [34] (essentially point cloud registration) to conduct

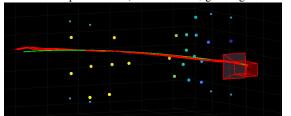| Method	Asynchronous	Bundle Adjustment
RPE [m]	**0.0124**	0.0136
ATE [m]	**0.0954**	0.1066

TABLE I: Relative Pose Error (RPE) and Absolute Trajectory Error (ATE) (see [35] for the definitions) for our asynchronous backend and frame-based bundle adjustment.

data association. The preprocessed data is then subjected to the MAP incremental SfM routine. For comparisons, we generated event images with 100 ms batches, then subjected the frames (with data association performed by associating the average coordinates of feature locations in frames) to bundle adjustment. Qualitative and quantitative results are depicted in Fig. 4 and Table I; the latter is by computing the relative pose error (RPE) and absolute trajectory error (ATE) (see [35] for the definitions) between the estimated trajectory and ground truth trajectory of the robot arm. The results show that, if data association is satisfactory, both asynchronous and frame-based back-ends perform equally well; of course, the latter also presumes accurate batching, which may not be possible in all cases (Sec. I). In addition, the smoother trajectory in the asynchronous back-end is due the GP prior. Using our unoptimised implementation of MAP incremental SfM, the result in Fig. 4c was generated in ≈1 minute (excluding preprocessing).

(a) Recording of event data and sample event image.

(b) Frame-based optimisation (red: estimated, green: ground truth).

(c) Asynchronous optimisation (red: estimated, green: ground truth).

Fig. 4: (a) Recording of event data with a Prophesee camera undergoing repetitive motion with a robot arm. (b)(c) Estimated trajectories from the frame-based back-end (bundle adjustment) and asynchronous back-end (MAP incremental SfM with GP model). Dots are the estimated scene points.

B. Evaluation of VO pipeline

To evaluate Algorithm 1, we employed publicly available event data, specifically the `Dynamic_6dof` sequence from the Event-Camera Dataset [2] (see [2] for the detailed description of the dataset). At this stage, our implementation of Algorithm 1 is rudimentary, thus our program is not able to perform VO continuously throughout the sequence. However, our program is able to work successfully on subsequences of 5 s-duration (≈5 M events in each sequence) from `Dynamic_6dof`; see Fig. 5. Currently, our implementation is also not optimised; the total runtime of the back-end was ≈10 minutes to execute on each subsequence. Nonetheless, we believe our results indicate the promise of our asynchronous back-end for event-only VO.
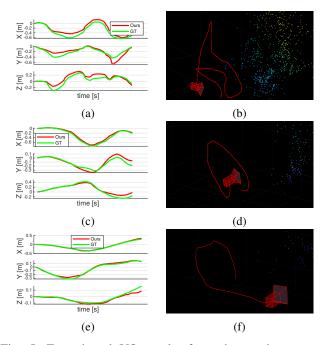

Fig. 5: Event-based VO results from three subsequences (rows 1 to 3) of `Dynamic_6dof`. (a)(c)(e) Coordinates (X, Y, Z) of the trajectory estimated by Algo. 1 compared against ground truth. (b)(d)(f) Estimated camera trajectory.

VII. CONCLUSIONS AND FUTURE WORK

We presented an event-driven back-end for event-only VO which integrates event observations in a principled way with a continuous-time SFM formulation. We believe our proposed back-end provides theoretical justification on how to achieve a practical event-only VO system. To this goal, we believe significant effort should be devoted to designing ad hoc mechanisms for front and back-end collaboration that leverage a CT trajectory representation. This integration could benefit typical sub-processes required in VO pipelines such as new point creation, data association and outlier removal.

REFERENCES

[1] "Event-based vision: A survey," *IEEE TPAMI*, 2020.

[2] 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," *The International Journal of Robotics Research*, vol. 36, no. 2, pp. 142–149, 2017.

[3] C. Forster, M. Pizzoli, and D. Scaramuzza, "SVO: Fast semi-direct monocular visual odometry," in *ICRA*, 2014.

[4] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale, "Keyframe-based visual–inertial odometry using nonlinear optimization," *IJRR*, 2015.

[5] A. Zihao Zhu, N. Atanasov, and K. Daniilidis, "Event-based visual inertial odometry," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[6] E. Mueggler, C. Forster, N. Baumli, G. Gallego, and D. Scaramuzza, "Lifetime estimation of events from dynamic vision sensors," in *2015 IEEE international conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 4874–4881.

[7] E. Mueggler, G. Gallego, and D. Scaramuzza, "Continuous-time trajectory estimation for event-based vision sensors," in *RSS*, 2015.

[8] E. Mueggler, G. Gallego, H. Rebecq, and D. Scaramuzza, "Continuous-time visual-inertial odometry for event cameras," *IEEE Transactions on Robotics*, 2018.

[9] Y. Wang, J. Yang, X. Peng, P. Wu, L. Gao, K. Huang, J. Chen, and L. Kneip, "Visual odometry with an event camera using continuous ray warping and volumetric contrast maximization," *arXiv preprint arXiv:2107.03011*, 2021.

[10] C. Le Gentil, F. Tschopp, I. Alzugaray, T. Vidal-Calleja, R. Siegwart, and J. Nieto, "IDOL: A framework for IMU-DVS odometry using lines," in *IEEE/RSJ IROS*, 2020.

[11] C. Le Gentil, T. Vidal-Calleja, and S. Huang, "Gaussian process preintegration for inertial-aided state estimation," *IEEE RAL*, 2020.

[12] J. Dong, B. Boots, and F. Dellaert, "Sparse Gaussian processes for continuous-time trajectory estimation on matrix Lie groups," *arXiv preprint arXiv:1705.06020*, 2017.

[13] J. Dong, M. Mukadam, B. Boots, and F. Dellaert, "Sparse Gaussian processes on matrix Lie groups: A unified framework for optimizing continuous-time trajectories," in *IEEE ICRA*, 2018.

[14] T. D. Barfoot, C. H. Tong, and S. Särkkä, "Batch continuous-time trajectory estimation as exactly sparse Gaussian process regression." in *RSS*, 2014.

[15] 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 RAL*, 2016.

[16] H. Rebecq, T. Horstschaefer, and D. Scaramuzza, "Real-time visual-inertial odometry for event cameras using keyframe-based nonlinear optimization," in *BMVC*, 2017.

[17] A. Zihao Zhu, N. Atanasov, and K. Daniilidis, "Event-based visual inertial odometry," in *CVPR*, 2017.

[18] Y. Zhou, G. Gallego, H. Rebecq, L. Kneip, H. Li, and D. Scaramuzza, "Semi-dense 3D reconstruction with a stereo event camera," in *ECCV*, 2018.

[19] Y. Zhou, G. Gallego, and S. Shen, "Event-based stereo visual odometry," *IEEE Transactions on Robotics*, 2021.

[20] 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 *CVPR*, 2018.

[21] A. R. Vidal, H. Rebecq, T. Horstschaefer, and D. Scaramuzza, "Ultimate SLAM? Combining events, images, and IMU for robust visual SLAM in HDR and high-speed scenarios," *IEEE RAL*, 2018.

[22] D. Liu, Á. Parra, and T.-J. Chin, "Globally optimal contrast maximisation for event-based motion estimation," in *CVPR*, 2020.

[23] X. Lagorce, G. Orchard, F. Galluppi, B. E. Shi, and R. B. Benosman, "HOTS: A hierarchy of event-based time-surfaces for pattern recognition," *IEEE TPAMI*, 2017.

[24] N. Khan and M. G. Martini, "Bandwidth modeling of silicon retinas for next generation visual sensor networks," *Sensors*, vol. 19, no. 8, p. 1751, 2019.

[25] S. A. Mohamed, M.-H. Haghbayan, J. Heikkonen, H. Tenhunen, and J. Plosila, "Towards real-time edge detection for event cameras based on lifetime and dynamic slicing," in *AICV*. Springer, 2020.

[26] Gtsam. [Online]. Available: https://gtsam.org/

[27] I. Alzugaray Lopez and M. Chli, "Haste: multi-hypothesis asynchronous speeded-up tracking of events," in *31st British Machine Vision Virtual Conference (BMVC 2020)*. ETH Zurich, Institute of Robotics and Intelligent Systems, 2020, p. 744.

[28] X. Yan, V. Indelman, and B. Boots, "Incremental sparse GP regression for continuous-time trajectory estimation & mapping," in *ISRR*, 2015.

[29] M. Kaess, V. Ila, R. Roberts, and F. Dellaert, "The Bayes tree: An algorithmic foundation for probabilistic robot mapping," in *Algorithmic Foundations of Robotics IX*. Springer, 2010.

[30] S. Anderson and T. D. Barfoot, "Full STEAM ahead: Exactly sparse Gaussian process regression for batch continuous-time trajectory estimation on SE (3)," in *IEEE/RSJ IROS*, 2015.

[31] P.-A. Absil, R. Mahony, and R. Sepulchre, *Optimization algorithms on matrix manifolds*. Princeton University Press, 2009.

[32] X. Yan, V. Indelman, and B. Boots, "Incremental sparse GP regression for continuous-time trajectory estimation and mapping," *Robotics and Autonomous Systems*, vol. 87, pp. 120–132, 2017.

[33] F. Dellaert and M. Kaess, "Square root SAM: Simultaneous localization and mapping via square root information smoothing," *IJRR*, 2006.

[34] D. Liu, A. Parra, and T.-J. Chin, "Spatiotemporal registration for event-based visual odometry," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 4937–4946.

[35] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of rgb-d slam systems," in *2012 IEEE/RSJ international conference on intelligent robots and systems*. IEEE, 2012, pp. 573–580.