A Tightly-Coupled Event-Inertial Odometry using Exponential Decay and Linear Preintegrated Measurements

Benny Dai, Cedric Le Gentil, and Teresa Vidal-Calleja

***Abstract*— In this paper, we introduce an event-based visual odometry and mapping framework that relies on decaying event-based corners. Event cameras, unlike conventional cameras, can provide sensor data during high-speed motions or in scenes with high dynamic ranges. Rather than providing intensity information at a global shutter rate, events are triggered asynchronously depending on whether there is a change in brightness at the pixel location. This novel sensing paradigm calls for unconventional ego-motion estimation techniques to address these new challenges. The key aspect of our framework is the use of a continuous representation of inertial measurements to characterise the system's motion which accommodates the asynchronous nature of the event data while estimating a discrete state in an optimisation-based approach. The proposed method relies on corners extracted from events-only data and associates them with a spatio-temporal locality scheme based on exponential decay. Event tracks are then tightly coupled with temporally accurate preintegrated inertial measurements, allowing for the estimation of ego-motion and a sparse map. The proposed method is evaluated on the Event Camera Dataset showing performance against the state-of-art in event-based visual-inertial odometry.**

I. INTRODUCTION

The estimation of a sensor's egomotion has been widely explored and utilised in many fields and applications including robotics and computer vision. Whilst there have been major developments in the space of Visual Inertial Odometry (VIO) ([1], [2], [3], [4]), these pipelines are limited by technical deficits of the conventional camera which include motion blur and low dynamic range that can occur in environments where there are high-speed motions or low lighting.

Event-based cameras such as the DAVIS 240 [5] sensor offer a different paradigm of sensing whereby changes in brightness in the environment at the pixel level are asynchronously transmitted. This addresses the shortcomings of conventional cameras as they can operate in the orders of microseconds and have a high dynamic range. However, they present a unique challenge in the processing of visual information; whereby these events are consequently triggered through egomotion or motion in the environment. In other words, events are triggered if there is motion either from the camera and/or the scene.

Rather than providing intensity information like conventional cameras: event streams contain accurate timestamps and detail the pixel location of whether illumination has increased/decreased. To leverage the perceptive capabilities of events, the use of aggregation or grouping has been popularised in the implementations of event-based VIO; shifting paradigms in the techniques and formulations to do so.

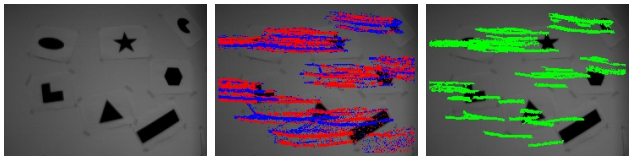

Fig. 1. Visualisation of the tracked events with the proposed approach at the beginning of $shapes\_translation$ from the Event Camera Dataset [6]. Left: Raw Image frame (not used). Middle: Corners from Arc* Corner Event Detector [7]. Right: Event corner tracks from the Exponential Decay Association Scheme.

Many works in event-based VIO still rely on conventional image techniques such as the FAST corner detector [8] and the KLT tracker [9]. This is achieved through the aggregation of events to yield a conventional-image-like appearance, which potentially prevents the full utilisation of the capabilities of events in certain situations. Specifically, such processing of the group of events could for example overlook information that could be needed for ego-motion estimation in aggressive motions.

In this paper, we investigate the use of event-based corners in an event-by-event fashion for egomotion estimation. We adopt the use of Arc* Corner Detector [7] to detect event corners and form tightly-coupled corner tracks (see Fig. 1) with a continuous form of preintegrated measurements known as Linear Preintegrated Measurement (LPMs) [10], which omits motion assumptions and allows for the query of inertial observations at any point in time. Our work proposes the use of tracks formed from pure events which are associated using spatial and temporal locality, without any assumptions made about motion. We present the full odometry and mapping pipeline that tightly integrates asynchronous events and inertial measurements in a batch-optimisation framework as shown in Fig. 2.

The rest of the paper is organised as follows: Section II-A reviews related literature on event-based odometry; especially those that are coupled with inertial measurements. Section III presents our proposed pipeline that incorporates events and inertial measurements to estimate ego-motion and mapping. Section IV details the front-end whilst Section V details the back-end. Section VI presents our quantitative evaluation of the Event Camera Dataset [6]. We then con-

All authors are with the UTS: Robotics Institute, Faculty of Engineering and IT, University of Technology Sydney, Australia. This work was partially supported by the Australian Research Council Discovery Project under Grant DP210101336. benny.dai@student.uts.edu.au, cedric.lengetil@uts.edu.au, teresa.vidalcalleja.uts.edu.au

clude our paper with remarks in Section VII.

II. RELATED WORK

A. Event-Based Visual Inertial Odometry

Estimating ego-motion with event cameras has garnered much attention in the robotics and computer vision literature, with [11] initially demonstrating the feasibility to estimate planar motion on planar scenes. Later on, [12] demonstrated that by using conventional camera frames in conjunction with events, 6-DoF motion estimation could be achieved.

More recently, [13] presented a VIO framework to address the incomplete estimation of scale that occurs in the case of visual odometry without the IMU. The authors proposed an iterative Expectation Maximisation scheme [14] to estimate optical flow - allowing them to achieve soft implicit data association of events through the use of cues from the IMU. They rely on FAST corners [8] detected on the underlying edge map on a temporal window of events.

The work in [15] proposed a continuous time framework that fuses both the event streams and inertial data; maintaining a smooth camera trajectory through a cubic spline. Since both events and inertial measurements can poll at a high rate; the estimation of the spline provided a mechanism to allow for fusing the asynchronous nature of events with high-rate inertial measurements.

Following the continuous time formulation, [2] proposed a framework that relies on the tracking and refinement of lines in scenes - as opposed to known priors in point and line primitives as in [15]. Ref. [2] employs Gaussian Preintegrated Measurements (GPMs) [16] which allow for the system's pose and velocities to be queried at any given time; even though it is not maintained in the state estimates. However, [13], [15], [16] are more suited to offline applications as they rely upon complex front-ends; which have a large computational burden.

Our approach is most similar to [17] as it relies on a spatio-temporal window of events to maintain a discrete number of states, which is refined through a non-linear optimisation framework through the use of virtual event frames. Motion-compensation is used to create virtual event frames which resemble an edge map of conventional intensity frames. The approach relies on the interpolation of preintegrated measurements [18] to align the events and relies on FAST corners [8] to recover tracks. With a computationally inexpensive front-end and with the back-end being OKVIS [19]; this approach provides an event-based VIO system with real-time performance. In contrast, our approach does not rely on conventional image processing techniques; and instead, adopts event-by-event based methods.

The work in [17] was further extended through the addition of image frames which incorporated the same feature processing in [20]. Using three modalities, they demonstrate the efficacy of events in conventional VIO being able to operate in obscure lighting and high dynamic range scenes. One of the key differences in the work done in [20] vs [17] is that the former synchronises the motion-compensated images to the same rate as conventional frames, whilst the latter operates asynchronously much like our proposed work. Furthermore, our work much like [17], only relies on two sensor modalities - events and inertial measurements.

Most event-based VIO systems have adopted the use of the conventional corner detection such as [13], [17], [20], in particular the FAST corner detection applied on some form of a spatiotemporal slice of events. To track the conventional corners over a period of time, [17], [20] adopt the use of the KLT tracker [9]. The use of corners as features for odometry and tracks has been highly utilised in many other works outside of event-based VIO, as they provide high precision and are distinct enough for recognition over time. However, to detect and track in this manner forces the odometry's front-end to be motion-variant; which limits the true potential of the event-based sensor.

In particular, event-based corners provide a means to continually operate at high speed and highly dynamic conditions versus conventional image corners such as FAST corners. Except for IDOL proposed by [2], there has been little work in exploring event-by-event based processing for odometry; in particular event-based corners. An extensive review on event-based corner detection can be found in [21] indicating that the Arc* Corner Event detector proposed by [7] is quite suitable for precision, whilst staying computationally efficient. Thus in our work, the Arc* Corner Event detector is used within our front-end.

B. Feature Tracking

Typically to track features or to achieve data association, most works can be categorised into either employing conventional tracking methods with features ([22], [7], [17], [20], [23]) or adopting alignment strategies for tracking events in spatio-temporal space ([14], [24], [25], [23], [26]). Typically alignment strategies are much more complex and computationally demanding but can yield more accurate tracking results through longer feature lifetimes. For example, [26] aligns events to Bezier curves to estimate motion by maximising contrast, rather than aligning to straight lines [24] or computing new feature locations through stacking events within short periods of time in [14].

In addition, [22] relies on using the constant velocity model to track corners in an event-by-event fashion, using optical flow orientation in the local neighbourhood [27] to achieve the data association across frames. As mentioned above for [17], [20]; this causes the features tracked to be motion-variant which could fail in aggressive motions. To address the model-based approaches, both ([25], [23]) propose to track patches rather than features. Ref. [23] relied upon conventional images to detect features and track along with events; whereby data association is achieved similarly like [28] and [14] where events are implicitly associated through compared patches. Whilst these methods do yield good feature track lifetimes, the initialisation and the quality of patches need to be sustained rigorously - creating a more complex system.

Ref. [7] relies on proximity and management on a tree graph whereby events are associated based on locality. Our

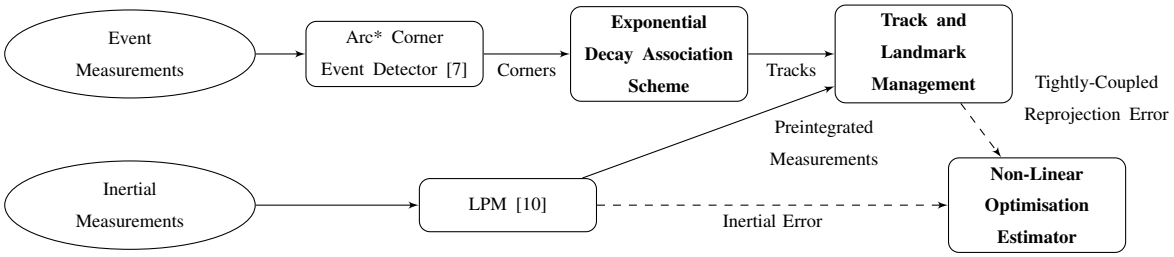

Fig. 2. Proposed structure of the Event-based Visual Inertial Odometry Pipeline. Nodes in bold are our proposed work.

tracking is most similar to [7], without the heuristics and management of the tree graph, and we exploit the sparsity of corners provided by Arc* Corner Event Detector [7]. Much like in ([7], [25], [2]); we process the event stream as they arrive.

III. METHOD OVERVIEW

The proposed method aims to recover the ego-motion of an event-based VIO system and the mapping of sparse landmarks. To achieve so, corners are detected in the event stream and associated with tracks. These tracks are then used to estimate the 3D locations of the corresponding landmarks and the system's trajectory, orientation and velocity. An illustration of the pipeline is shown in Fig. 2, indicating how our residual terms are processed from both event and inertial measurements. Due to the asynchronous nature of events, preintegrated inertial measurements are calculated for each event to accurately determine the system's state; tightly-coupling events and inertial information to form the reprojection error. The reprojection error of both the event corners and the projected preintegrated inertial measurements is then minimised; to estimate a discrete set of M states containing the system's poses, velocities and environment landmarks.

A. Problem Statement

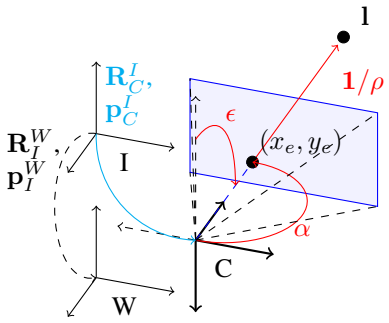

Fig. 3. Illustration of the event-based and IMU camera system. W represents the world reference frame, C represents the camera frame and I is the inertial frame. $\mathbf{R}_I^W$ and $\mathbf{p}_I^W$ of $\mathfrak{X}^m$ is indicated as continuous I in W frame. The landmark $\mathbf{l}$ observed from the camera is projected into the image plane with coordinates (x_e, y_e). The bearings (ϵ and α), and ρ the inverse depth are used in our Inverse Depth Parameterisation for landmarks; and are indicated in red. The cyan line represents the prior known transformation, which is obtained through extrinsic calibration of I and C which are $\mathbf{R}_C^I$ and $\mathbf{p}_C^I$

Let us consider a sensor system composed of an event camera and a 6-DoF IMU rigidly mounted together. Let us define the world frame as W, the camera frame as C and I as the IMU frame. To obtain $\mathbf{R}_C^I$ and $\mathbf{p}_C^I$, prior extrinsic calibration is performed.

In general, the rotation between Frame A and Frame B is denoted as $\mathbf{R}_A^B \in SO(3)$, where the translation vector is denoted as $\mathbf{p}_A^B$. A 3D point $\mathbf{p}^A$ from frame A can be projected in frame B as

$$\mathbf{p}^B = \mathbf{R}_A^B \mathbf{p}^A + \mathbf{p}_A^B. \tag{1}$$

Formally, we define the part of the system's state $\mathfrak{X}^m$ to correspond to the state of the inertial frame I with respect to W at time τ_m:

$$\mathfrak{X}^m := [\mathbf{p}_I^{W\tau_m}; \mathbf{R}_I^{W\tau_m}; \mathbf{v}_I^{W\tau_m}; \mathbf{b}_f^{\tau_m}; \mathbf{b}_\omega^{\tau_m}] \in \mathbb{R}^3 \times SO^3 \times \mathbb{R}^9 \tag{2}$$

where $\mathbf{b}_f^{\tau_m}$ and $\mathbf{b}_f^{\tau_m}$ are the IMU's accelerometer and gyroscope biases. Note that we assume the slow-varying inertial biases to be constant in between two state timestamps. Landmarks $\mathbf{l}^j$ (with $j = 1, ..., Q$) are also estimated and maintained in conjunction with the system trajectory. We denote the overall estimated state with $\mathcal{X} = \{\mathfrak{X}^1, ..., \mathfrak{X}^M, \mathbf{l}^1, ..., \mathbf{l}^Q\}$. An illustration of the reference frames, pose of $\mathfrak{X}$ and landmark $\mathbf{l}$ are shown in Fig. 3. Landmark parameterisation and treatment are discussed more in detail in Section V-A.

The odometry and mapping problem is then formulated as a Maximum Likelihood Estimation (MLE)

$$\mathcal{X}^* = \arg\min_{\mathcal{X}} \; -\log(p(\mathrm{Z}|\mathcal{X})) = \arg\min_{\mathcal{X}} \; J(\mathcal{X}), \tag{3}$$

where Z are a form of the available sensor measurements and J is the cost function to be optimised. More details on the event-based visual-inertial cost function J is detailed in Section V-B. The events and inertial measurements are processed as described by the front-end, while the state estimation is performed via the approach depicted in the back-end.

IV. FRONT-END

A. Corner-Event Detection and State Instantiation

To extract the distinctive features from a stream of events, we rely on the use of Arc* Corner Event Detector [7]. This provides a purely event-driven and computationally efficient method to filter out many events that do not carry distinguishable information, therefore allowing us to apply a

relatively simple data association scheme. Like events, corner data is still collected as an asynchronous stream

$$\mathbf{c} = \{x_e, y_e, t_e\}, \tag{4}$$

where x_e, y_e are the pixel coordinates of the event corner, and t_e is the arrival time of the event corner. Note that in our work, we disregard the polarity - although Arc* Corner Event Detector does provide polarity information. We neglect polarity in our work to ensure that tracks can capture corner events in back and forth motions; where polarity would typically change.

The creation of a new discrete state $\mathcal{X}^m$ is aligned against windows of N corners. The duration of each window is inversely proportional to the rate of corners in the data stream. The state's timestamp τ_m is defined as the timestamp of the $(N+1)^{th}$ corner that satisfies $t_e \geq \tau_{m-1}$ as illustrated in Fig. 4.

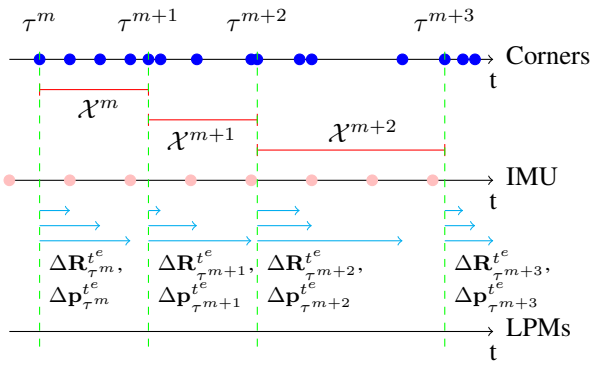

Fig. 4. Window of corners denoting the management of $\mathcal{X}^m$ and τ^m and how LPMs ($\Delta\mathbf{R}^{t^e}_{\tau^m}$, $\Delta\mathbf{p}^{t^e}_{\tau^m}$) [10] are calculated for each N corners at t^e. In this example, we have blue circles representing N=4 corners, and each state window is fixed according to N corners. Pink circles represent IMU measurements, and the cyan arrows indicate how LPMs are calculated for each N corners. We fix τ^m on the first corner time of each window of corners.

B. Exponential Decay Event Association Scheme

To achieve data association between corner events, we first maintain a history of corners through the use of the Surface of Active Events (time surfaces) [27]

$$S : (x_e, y_e) \in \mathbb{R}^2 \mapsto \mathfrak{t}_e \in \mathbb{R} \tag{5}$$

that associates each pixel location (x_e, y_e) with the timestamp $\mathfrak{t}_e$ of the last corner event occurring at that location.

We then convolve the Surface of Active Events with the Exponential Decay Kernel [29],

$$\boldsymbol{\lambda}(x_e, y_e) = \begin{bmatrix} \delta_e(1,-1) & \delta_e(1,0) & \delta_e(1,1) \\ \delta_e(0,-1) & 0 & \delta_e(0,1) \\ \delta_e(-1,-1) & \delta_e(-1,0) & \delta_e(-1,1) \end{bmatrix}, \tag{6}$$

with

$$\delta_e(i,j) = \exp(-\frac{S(x_e, y_e) - S(x_e + i, y_e + j)}{\zeta}), \tag{7}$$

to establish data association. The parameter ζ is the exponential decay constant that dictates the degree of sensitisation of recent events. Eqn. (6) returns λ, a map that quantifies the time difference between the event and its neighbours. More recent events would correspond to values in the λ map approaching 1, while older events would have values in the λ map that tend to 0.

The maximum value and position in

$$\begin{aligned} n_{\lambda^*} &= \sum_{i,j} thr(\lambda_{ij}, \lambda^*) \\ \text{with} \quad thr(\lambda_{ij}, \lambda^*) &= \begin{cases} 1 & \text{if} \quad \lambda_{ij} > \lambda^*, \\ 0 & \text{otherwise.} \end{cases} \end{aligned} \tag{8}$$

is then used for the data association of event corners into a track, assuming it is greater than a decided threshold which we call λ^*. We initialise an arriving corner to be a new track if the size of n_{λ^*} is 0. This means that there are no recent corners in the neighbourhood of the corner. If the size of n_{λ^*} is greater than 0, we then associate the arriving corner with the neighbouring corner event corresponding to the maximum value of n_{λ^*}. In other words, the corner is now associated with the most recent corner in the kernel operation which assumes that it is the same corner in a recent position.

Since corner events are temporally and spatially sparse, we can assume that corners that were triggered locally must be of the same landmark. We note that the utilisation of the exponential decay kernel on event streams has been adopted in [29]; but not to associate events for tracking. For our purposes, the application of the exponential decay kernel ensures that relevant context is retained when associating events into tracks; keeping the front-end computationally efficient.

C. Event Track Pruning

Since our data associations on events are based on the locality of recent events, the effect of event clutter can cause tracks that resemble branching or bleeding. An example of this is when events in opposing directions and heading are associated. Thus, we close a track for tracking depending on the cumulative sum of pixel distances

$$d_p = \sum_{1}^{N} \sqrt{(x_e - \mathbf{x_e}^{-1:-N})^2 - (y_e - \mathbf{y}_e^{-1:-N})^2)} \tag{9}$$

on an N number of recent corners which we refer to as N_d. We consider a threshold of pixel distance d_p^* and if d_p exceeds it, then we close the track. $\mathbf{x}_e^{-1:-N}$ and $\mathbf{y}_e^{-1:-N}$ are the recent corner positions in the track.

D. Inertial Linear Preintegrated Measurements

To accurately account for the system's motion, we associate each corner event from each track with rotational and positional preintegrated measurements denoted $\Delta\mathbf{R}^{\mathfrak{t}_e}_{\tau_m}$ and $\Delta\mathbf{p}^{\mathfrak{t}_e}_{\tau_m}$ respectively. We use the LPMs [10] for efficient and accurate preintegration of the raw IMU measurements. The LPMs first upsample the raw gyroscope measurements via linear interpolation and perform the standard preintegration [18] at a high frequency to obtain $\Delta\mathbf{R}^{\mathfrak{t}_e}_{\tau_m}$ and project the accelerometer measurements into the first IMU frame.

Then, $\Delta\mathbf{p}_{\tau_m}^{\mathfrak{t}_e}$ is computed continuously by double integrating the projected acceleration with a piece-wise linear model (equivalent to the trapezoidal rule for numerical integration). The system's pose at time $\mathfrak{t}_e$ is then computed as

$$\mathbf{R}_I^{W\,\mathfrak{t}_e} = \mathbf{R}_I^{W\,\tau_m}\Delta\mathbf{R}_{\tau_m}^{\mathfrak{t}_e}, \tag{10}$$

$$\mathbf{p}_I^{W\,\mathfrak{t}_e} = \mathbf{p}_I^{W\,\tau_m} + \mathbf{v}_I^{W\,\tau_m}\Delta t + \frac{\mathbf{g}^W\Delta t^2}{2} + \mathbf{R}_I^{W\,\tau_m}\Delta\mathbf{p}_{\tau_m}^{\mathfrak{t}_e}, \tag{11}$$

with $\mathbf{g}^W$ being the known gravity vector in the world frame, $\Delta t = \mathfrak{t}_e - \tau_m$ and τ_m the highest state timestamp that satisfies $\mathfrak{t}_e > \tau_m$.

As part of the IMU residuals (c.f. Section V-D) we also leverage the velocity part of the LPMs $\Delta\mathbf{v}_{\tau_m}^{\tau_{m+1}}$ that relates to the system's velocity as

$$\mathbf{v}_I^{W\,\tau_{m+1}} = \mathbf{v}_I^{W\,\tau_m} + \mathbf{g}^W\Delta t + \mathbf{R}_I^{W\,\tau_m}\Delta\mathbf{v}_{\tau_m}^{\tau_{m+1}}\,. \tag{12}$$

Note that we also use the postintegration bias correction mechanism of the LPMs to accurately account for the slow-varying additive biases present in gyroscope and accelerometer data (not detailed here due to space limitation).

V. BACK-END

A. Inverse Depth Parameterisation

To maintain and parameterise the landmark locations that correspond to tracked events, we rely on Inverse Depth Parameterisation (IDP) ([30], [31]). The landmarks are encoded as

$$\mathbf{l}_{IDP} = \begin{bmatrix}(\epsilon,\alpha)\\ \rho\end{bmatrix} \in \mathbb{R}^3\,, \tag{13}$$

where ϵ and α are the elevation and azimuth angles respectively; and ρ is the inverse depth which is $1/d$, where d is the distance from the camera position to the landmark. This is illustrated in Fig. 3.

Expressing the IDP in Euclidean form, we get

$$\mathbf{l}_{EP} =_A \mathbf{p}_C^W + \mathbf{m}^*(\epsilon,\alpha)/\rho\,, \tag{14}$$

where ${}_A\mathbf{p}_C^W$ is the camera position where the feature was first initialised, which can be shown in Eqn. (22) using Eqn. (11) later on with prior extrinsic calibration between the camera and IMU, and

$$\mathbf{m}^*(\epsilon,\alpha) = \begin{bmatrix}\cos(\epsilon)\cos(\alpha)\\ \cos(\epsilon)\sin(\alpha)\\ \sin(\epsilon)\end{bmatrix} \tag{15}$$

is the unit vector in direction of (ϵ,α).

We first consider

$$\mathbf{u} = [x_e, y_e, 1] \tag{16}$$

containing the pixel coordinate of the event.

The initialisation of the landmark is a function of the measurement and the state $\mathcal{X}$

$$\mathbf{l}_{IDP} = \begin{bmatrix}\boldsymbol{\mu}^*(\mathbf{R}_I^{W\,t_1}\Delta\mathbf{R}_{t_1}^{t_2}\mathbf{R}_C^I\mathbf{K}^{-1}\mathbf{u})\\ \rho^*\end{bmatrix} \tag{17}$$

where $\mathbf{K}^{-1}$ is the inverse of the intrinsic camera matrix that back projects pixel measurements into landmarks, and $\boldsymbol{\mu}^*(\mathbf{m})$ is the direction vector that gives us the elevation and azimuth angles

$$\boldsymbol{\mu}^*(m_x,m_y,m_z) = \begin{bmatrix}arctan(m_z/\sqrt{m_x^2+m_y^2}\\ arctan(m_y/m_x)\end{bmatrix} = \begin{bmatrix}\epsilon\\ \alpha\end{bmatrix} \tag{18}$$

where ρ^* is given as a prior.

B. Visual-Inertial Integration

Following (3), the visual-inertial odometry and mapping problem is formulated as a joint optimisation with the cost function defined as

$$J(\mathcal{X}) := \sum_{m=1}^{M}\sum_{j\in\mathcal{J}(m)} \mathbf{e}_r^{j,m\intercal}\mathbf{W}_r^{j,k}\mathbf{e}_r^{j,m} + \sum_{m=1}^{M-1}\mathbf{e}_I^{m\intercal}\mathbf{W}_I^m\mathbf{e}_I^m\,, \tag{19}$$

where the weighted reprojection error and inertial error are $\mathbf{e}_r$ and $\mathbf{e}_I$ respectively, with m denoting the frame/state index and j the landmarks index. Additionally, $\mathbf{W}_r$ denotes the information matrix of the residual corresponding to the event measurements and $\mathbf{W}_I$ denotes the information matrix for IMU residuals. We note that our proposed VIO backend differs from traditional backends as we leverage LPMs to provide fine-temporal granularity in the calculation of the reprojection error. This provides us with the ability to represent the trajectory in a continuous manner while maintaining discrete states.

C. Tightly-Coupled Reprojection Error

Our state variables $\mathfrak{X}^m$ are initiated at the same timestamp of the first corner from a window of N corners like shown in Fig. 4. To determine the system's state in a continuous manner at the corner event arrival time $\mathfrak{t}_e$, we rely on the use of tightly-coupling LPMs [10]. Thus, using the camera-IMU extrinsic calibration and Eqn. (10) and (11), the camera position at any event timestamps is computed as

$$\mathbf{p}_C^{W\,\mathfrak{t}_e} = \mathbf{p}_I^{W\,\mathfrak{t}_e} + \mathbf{R}_I^{W\,\mathfrak{t}_e}\mathbf{p}_C^I \tag{20}$$

It allows for the computation of a residual error term for each event, for every subsequent track.

Let us denote ${}_A\mathbf{p}_C^W$ the position of the camera at the timestamp of the first event in a track. This position is used as the anchor of the landmark's IDP associated with the track.

Given an IDP and $\mathfrak{X}^m$, it is possible to project the landmark in the camera plane at any event timestamp $\mathfrak{t}_k$ as

$$\hat{\mathbf{u}}^{j,k} = \mathbf{K}((\mathbf{R}_I^{W\,\mathfrak{t}_k}\mathbf{R}_C^I)^T(\mathbf{m}^*(\epsilon,\alpha) - (\mathbf{p}_C^{W\,\mathfrak{t}_k} -_A \mathbf{p}_C^W)\rho)) \tag{21}$$

where $\mathbf{K}$ is the intrinsic camera matrix that projects landmarks into the image following the pinhole camera model.

Then, the reprojection error simply corresponds to the difference between the predicted event coordinates $\hat{\mathbf{u}}$ and the actual event measurement $\mathbf{u}$

$$\mathbf{e}_r^{j,k} = \mathbf{u}^{j,k} - \hat{\mathbf{u}}^{j,k}\,. \tag{22}$$

D. Inertial and Bias Error

The proposed method uses the LPMs [10] based on Eqn. (10), (11), and (12) to constrain the system's trajectory from state to state. We constitute inertial residuals as

$$\mathbf{e}_I^k(\mathfrak{X}^k, \mathfrak{X}^{k+1}) = \begin{bmatrix} \hat{\mathbf{p}}_I^{W^{\tau_{k+1}}} - \mathbf{p}_I^{W^{\tau_{k+1}}} \\ Log(\hat{\mathbf{R}}_I^{W^{\tau_{k+1}}{}^T} \mathbf{R}_I^{W^{\tau_{k+1}}}) \\ \hat{\mathbf{v}}_I^{W^{\tau_{k+1}}} - \mathbf{v}_I^{W^{\tau_{k+1}}} \\ \mathbf{b}_f^k - \mathbf{b}_f^{k+1} \\ \mathbf{b}_\omega^k - \mathbf{b}_\omega^{k+1} \end{bmatrix} \in \mathbb{R}^{15} \quad (23)$$

with $Log(.)$ the logarithmic map from $SO(3)$ (rotation matrix) to $\mathfrak{so}(3)$ (axis-angle), and $\hat{\bullet}_I^{W^{\tau_{k+1}}}$ the system's state predicted at τ_{k+1} given the preintegrated measurements and the current estimate of $\mathfrak{X}^k$. As mentioned in Section III-A, the IMU measurements are subject to slow-varying additive biases. The last two lines of (23) enforce the slow-varying nature of the biases with a Brownian motion model.

VI. EXPERIMENTS

Our implementation uses Ceres [32] to solve the optimisation problem in Eqn. (19); and relies on the Robot Operating System (ROS) [33] for handling the sensor data streams. To initialise the orientation in the beginning, we align a small window of accelerometer measurements with the gravity vector in the world frame assuming the coordinate acceleration to be small with respect to the gravity-induced acceleration.

The front-end processes corner events asynchronously as it arrives, associating them into tracks that are later used to estimate the trajectory. In its current stage, the front-end does not possess any explicit mechanism for outlier rejection nor insuring that corner tracks contain enough information to allow for trajectory estimation. Thereupon, we only consider the TQ longest tracks when creating a new state (with TQ between 30 and 50 as shown in Table I), and apply a Cauchy loss function to the reprojection residuals.

In Fig. 5, we show the soundness of our method over simulated data of a planar scene generated with the ESIM simulator [34]. As expected, the estimated landmarks lie in a plane in front of the camera and one can recognise the corners of the shapes observed by the camera. We compare quantitatively on the Event Camera Dataset [6] with the state-of-the-art [17] which relies on FAST corners extracted from motion-compensated images with a conventional VIO framework.

A. Datasets and Evaluation

We evaluate the proposed pipeline on the Event Camera Datasets [6] which contains sequences captured in various indoor environments using the DAVIS 240 [5]. Each dataset used in our evaluation contains conventional frames, inertial data, and a stream of events. Note that our method does not use the conventional greyscale frames in the estimation process. These are only used for visualisation purposes as in Fig. 1 and 6. In most of the sequences, the aggressiveness of the motion is progressive and ends with very challenging jerk-like movements. In the *hdr_poster* and *hdr_boxes* sequences, the sensor observes scenes with a high dynamic range.

TABLE I

PARAMETERS USED IN OUR PROPOSED METHOD IN THE EVALUATION OF THE EVENT CAMERA DATASET [6]. WC IS DENOTED AS THE NUMBER OF EVENTS USED FOR THE WINDOW OF CORNERS AND TQ ARE THE NUMBER OF TRACKS THAT WERE QUERIED AT EACH WINDOW OF CORNERS J

Sequence	WC	TQ	ρ^*	ζ	λ^*	d_p^*	N_D
boxes_6dof	20,000	50	0.8	0.1	0.9	30	3
boxes_translation	5,000						
dynamic_6dof		30					
dynamic_translation							
hdr_boxes	10,000	50	1.0				
hdr_poster		30					
poster_6dof							
poster_translation							
shapes_6dof	20,000	50					
shapes_translation							

TABLE II

ACCURACY OF THE PROPOSED APPROACH AGAINST [17], THE STATE-OF-THE-ART

Sequence	**Proposed**		**State-of-the-art [17]**	
	Mean Position Error (%)	Mean Yaw Error (deg/m)	Mean Position Error (%)	Mean Yaw Error (deg/m)
boxes_6dof	0.015	**1.103**	**0.014**	2.291
boxes_translation	0.010	0.870	**0.010**	**0.510**
dynamic_6dof	0.015	0.620	**0.012**	**0.398**
dynamic_translation	0.009	**2.218**	**0.008**	2.714
hdr_boxes	0.018	0.450	**0.012**	**0.406**
hdr_poster	0.028	0.695	**0.004**	**0.070**
poster_6dof	**0.012**	**0.406**	0.016	1.730
poster_translation	0.019	**0.207**	**0.007**	2.676
shapes_6dof	**0.005**	**0.326**	0.011	0.585
shapes_translation	**0.004**	**0.234**	0.012	2.380

We evaluate the proposed method using all the datasets that contain 6-DoF ground truth omitting rotation-only variants, and sequences that do not provide inertial data. We use the Kalibr toolbox [35] to estimate the extrinsic calibration $\mathbf{R}_I^C$ and $\mathbf{p}_I^C$, the intrinsic camera matrix K and also the constant time offset between event and inertial measurements.

To quantitatively evaluate our method, we align the ground-truth and estimated trajectories using [36]. We use both the RMSE error over the full trajectory, as well as average relative position and yaw errors. The relative metrics are computed over chunks of trajectory that correspond to $\{10, 20, 30, 40, 50\}\%$ of the full trajectory length.

The proposed method relies on a small set of parameters that are dependent on the density of events in different scenes. While default parameters work sufficiently to obtain decent results; they can be slightly tuned to obtain optimal results. We denote all the parameters used in our experiments in Table I.

TABLE III

OVERALL RMSE TRANSLATION ESTIMATION (M) OF THE PROPOSED APPROACH AGAINST [17], THE STATE-OF-THE-ART

	boxes		dynamic		hdr		poster		shapes	
	6dof	translation	6dof	translation	boxes	poster	6dof	translation	6dof	translation
Proposed	1.302	0.700	0.647	0.287	1.036	1.754	**0.789**	0.997	**0.295**	**0.250**
State-of-the-art [17]	**1.057**	**0.686**	**0.487**	**0.279**	**0.727**	**0.257**	1.055	**0.378**	0.558	0.732

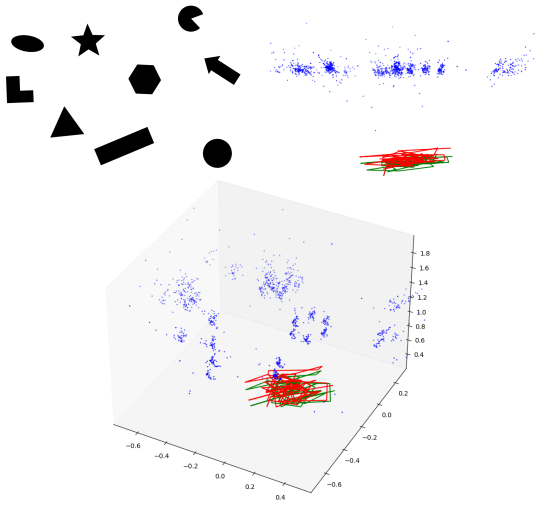

Fig. 5. The estimated landmarks and trajectories (bottom and top right) that were used on the ESIM [34] using the *shapes* texture (top left). The Red line is the Ground Truth and the Green line is the estimated trajectory of our pipeline. Blue dots are landmarks.

B. Results

Table II shows the average relative errors for both our method and [17]. Overall, the two methods perform similarly with a significant advantage for our method on the $shapes$ datasets. However, on hdr_poster and $poster\_translation$, [17] provides more accurate trajectory estimates. We believe the drop in accuracy of our method on these sequences is correlated with the degree of texture in the scene. Highly textured scenes contain a lower number of "clear" corners making the corner extraction nosier. The authors of corner detector [7] used in this work also report that corners can "jump" in highly textured environments. Therefore, it leads to cluttered feature tracks that do not convey much information about the scene or the system's trajectory. This is illustrated in Fig. 6 where some tracks on the right correspond to the camera motion while the majority of tracks are curled up. This highlights the need for future work on outlier rejection mechanisms at the front-end level.

The RMSE errors reported in Table III show the same trends as the relative metrics discussed above. While both frameworks perform in the same order of magnitude, note that [17] estimates the system's pose in real-time, unlike the proposed method. For example, the trajectory estimation of the 60-second $shapes\_6dof$ sequence runs in about 30 minutes. In its current stage, the minimisation Eqn. (3) is solved as a full-batch optimisation every time a new pose is added to the state. This simple strategy inherently prevents the proposed method to be scaled to large datasets. However, we believe that future work on integrating sliding window estimation techniques alongside some software optimisation and multi-threading of front-end operation would allow for real-time computation.

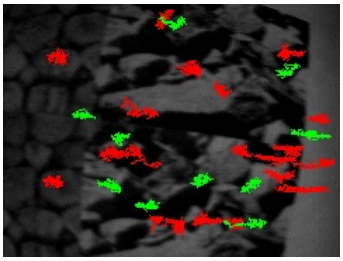

Fig. 6. Example of poor performance of our proposed front-end in the $poster\_translation$ in the Event Camera Dataset [17]. Events in red are events that were initiated in previous state windows, whilst events in green are initiated within the state window.

VII. CONCLUSIONS

In this paper, we introduce a pipeline that tightly couples event-based corner tracks with inertial measurements for VIO. In contrast with most works in event-based VIO, we do not rely on the use of image-based corner detection based on a spatio-temporal window of events but use asynchronously detected and tracked corner events. Our "unconventional" approach uses LPMs to characterise the system's motion in a continuous fashion between estimated discrete states and leverages "per-event" residuals in an optimisation-based probabilistic formulation. Through real-world experiments, we show that the proposed method can perform odometry in challenging environments; with relatively similar results to the state-of-the-art [17]. Hence, we demonstrate that accurate event-based odometry does not need to rely on traditional frame-based VO techniques over the image-like aggregation of events.

While we present an unrefined pipeline that lacks real-time capabilities, the proposed methods have great potential to operate in highly dynamic environments with very few assumptions about motion and the scene. We believe that little work on software optimisation and the use of sliding window estimation will make our framework computationally efficient without sacrificing performance.

Future work also includes further developments of the front-end to be more robust whilst staying computationally efficient. To be specific, improvements in feature lifetimes and their consistency is needed, allowing them to operate robustly in a larger variety of environments.

REFERENCES

[1] M. Bloesch, S. Omari, M. Hutter, and R. Siegwart, "Robust visual inertial odometry using a direct EKF-based approach," *IEEE International Conference on Intelligent Robots and Systems*, vol. 2015-Decem, pp. 298–304, 2015.

[2] C. Le Gentil, F. Tschopp, I. Alzugaray, T. Vidal-Calleja, R. Siegwart, and J. Nieto, "IDOL: A framework for IMU-DVS odometry using lines," *IEEE International Conference on Intelligent Robots and Systems*, pp. 5863–5870, 2020.

[3] T. Qin, P. Li, and S. Shen, "VINS-Mono: A Robust and Versatile Monocular Visual-Inertial State Estimator," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 1004–1020, aug 2017. [Online]. Available: http://arxiv.org/abs/1708.03852 http://dx.doi.org/10.1109/TRO.2018.2853729

[4] J. Delmerico and D. Scaramuzza, "A Benchmark Comparison of Monocular Visual-Inertial Odometry Algorithms for Flying Robots," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 2502–2509, 2018.

[5] C. Brandli, R. Berner, Minhao Yang, Shih-Chii Liu, and T. Delbruck, "A 240 × 180 130 dB 3 µs Latency Global Shutter Spatiotemporal Vision Sensor," *IEEE Journal of Solid-State Circuits*, vol. 49, no. 10, pp. 2333–2341, oct 2014. [Online]. Available: https://ieeexplore.ieee.org/document/6889103

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

[7] I. Alzugaray and M. Chli, "Asynchronous Corner Detection and Tracking for Event Cameras in Real Time," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3177–3184, 2018.

[8] E. Rosten, R. Porter, and T. Drummond, "Faster and better: A machine learning approach to corner detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 1, pp. 105–119, oct 2010. [Online]. Available: http://arxiv.org/abs/0810.2434 http://dx.doi.org/10.1109/TPAMI.2008.275

[9] B. D. Lucas and T. Kanade, "An Iterative Image Registration Technique with an Application to Stereo Vision," in *Proceedings of the 7th International Joint Conference on Artificial Intelligence - Volume 2*, ser. IJCAI'81. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1981, pp. 674–679.

[10] C. Le Gentil and T. Vidal-Calleja, "Continuous Integration over SO(3) for IMU Preintegration," in *Robotics: Science and Systems XVII*, no. 3. Robotics: Science and Systems Foundation, jul 2021. [Online]. Available: http://www.roboticsproceedings.org/rss17/p078.pdf

[11] D. Weikersdorfer, R. Hoffmann, and J. Conradt, "Simultaneous localization and mapping for event-based vision systems," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 7963 LNCS, pp. 133–142, 2013.

[12] H. Kim, S. Leutenegger, and A. J. Davison, "Real-time 3D reconstruction and 6-DoF tracking with an event camera," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 9910 LNCS, pp. 349–364, 2016.

[13] A. Z. Zhu, N. Atanasov, and K. Daniilidis, "Event-based visual inertial odometry," *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, vol. 2017-Janua, pp. 5816–5824, 2017.

[14] ——, "Event-based feature tracking with probabilistic data association," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 4465–4470, 2017.

[15] E. Mueggler, G. Gallego, H. Rebecq, and D. Scaramuzza, "Continuous-Time Visual-Inertial Odometry for Event Cameras," *IEEE Transactions on Robotics*, vol. 34, no. 6, pp. 1425–1440, 2018.

[16] C. Le Gentil, T. Vidal-Calleja, and S. Huang, "Gaussian Process Preintegration for Inertial-Aided State Estimation," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2108–2114, 2020.

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

[18] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, "On-Manifold Preintegration for Real-Time Visual-Inertial Odometry," *IEEE Transactions on Robotics*, vol. 33, no. 1, pp. 1–21, 2017.

[19] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale, "Keyframe-based visual–inertial odometry using nonlinear optimization," *The International Journal of Robotics Research*, vol. 34, no. 3, pp. 314–334, mar 2015. [Online]. Available: http://journals.sagepub.com/doi/10.1177/0278364914554813

[20] 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," *arXiv*, vol. 3, no. 2, 2017.

[21] Ö. Yılmaz, C. Simon-Chane, and A. Histace, "Evaluation of Event-Based Corner Detectors," *Journal of Imaging*, vol. 7, no. 2, p. 25, 2021.

[22] X. Clady, S. H. Ieng, and R. Benosman, "Asynchronous event-based corner detection and matching," *Neural Networks*, vol. 66, pp. 91–106, 2015. [Online]. Available: http://dx.doi.org/10.1016/j.neunet.2015.02.013

[23] D. Gehrig, H. Rebecq, G. Gallego, and D. Scaramuzza, "EKLT: Asynchronous Photometric Feature Tracking Using Events and Frames," *International Journal of Computer Vision*, vol. 128, no. 3, pp. 601–618, 2020. [Online]. Available: https://doi.org/10.1007/s11263-019-01209-w

[24] G. Gallego, H. Rebecq, and D. Scaramuzza, "A Unifying Contrast Maximization Framework for Event Cameras, with Applications to Motion, Depth, and Optical Flow Estimation," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 3867–3876, 2018.

[25] I. Alzugaray and M. Chli, "Asynchronous Multi-Hypothesis Tracking of Features with Event Cameras," *Proceedings - 2019 International Conference on 3D Vision, 3DV 2019*, pp. 269–278, 2019.

[26] H. Seok and J. Lim, "Robust feature tracking in DVS event stream using bézier mapping," *Proceedings - 2020 IEEE Winter Conference on Applications of Computer Vision, WACV 2020*, pp. 1647–1656, 2020.

[27] R. Benosman, C. Clercq, X. Lagorce, S. H. Ieng, and C. Bartolozzi, "Event-based visual flow," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 2, pp. 407–417, 2014.

[28] B. Kueng, E. Mueggler, G. Gallego, and D. Scaramuzza, "Low-latency visual odometry using event-based feature tracks," *IEEE International Conference on Intelligent Robots and Systems*, vol. 2016-Novem, pp. 16–23, 2016.

[29] 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 Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 7, pp. 1346–1359, 2017.

[30] J. Civera, A. Davison, and J. Montiel, "Inverse Depth Parametrization for Monocular SLAM," *IEEE Transactions on Robotics*, vol. 24, no. 5, pp. 932–945, oct 2008. [Online]. Available: http://ieeexplore.ieee.org/document/4637878/

[31] J. Solà, T. Vidal-Calleja, J. Civera, and J. M. M. Montiel, "Impact of landmark parametrization on monocular EKF-SLAM with points and lines," *International Journal of Computer Vision*, vol. 97, no. 3, pp. 339–368, 2012.

[32] S. Agarwal, K. Mierle, and Others, "Ceres Solver," http://ceres-solver.org, 2020. [Online]. Available: http://ceres-solver.org

[33] Stanford Artificial Intelligence Laboratory et al., "Robotic Operating System." [Online]. Available: https://www.ros.org

[34] H. Rebecq, D. Gehrig, and D. Scaramuzza, "ESIM: an Open Event Camera Simulator," *CoRL*, no. CoRL, pp. 1–14, 2018. [Online]. Available: https://github.com/uzh-rpg/rpg_esim

[35] P. Furgale, J. Rehder, and R. Siegwart, "Unified temporal and spatial calibration for multi-sensor systems," *IEEE International Conference on Intelligent Robots and Systems*, pp. 1280–1286, 2013.

[36] Z. Zhang and D. Scaramuzza, "A Tutorial on Quantitative Trajectory Evaluation for Visual(-Inertial) Odometry," *IEEE International Conference on Intelligent Robots and Systems*, pp. 7244–7251, 2018.