Spatiotemporal Registration for Event-based Visual Odometry

Daqi Liu Álvaro Parra Tat-Jun Chin
School of Computer Science, The University of Adelaide
{daqi.liu, alvaro.parrabustos, tat-jun.chin}@adelaide.edu.au

Abstract

A useful application of event sensing is visual odometry, especially in settings that require high-temporal resolution. The state-of-the-art method of contrast maximisation recovers the motion from a batch of events by maximising the contrast of the image of warped events. However, the cost scales with image resolution and the temporal resolution can be limited by the need for large batch sizes to yield sufficient structure in the contrast image. In this work, we propose spatiotemporal registration as a compelling technique for event-based rotational motion estimation. We theoretically justify the approach and establish its fundamental and practical advantages over contrast maximisation. In particular, spatiotemporal registration also produces feature tracks as a by-product, which directly supports an efficient visual odometry pipeline with graph-based optimisation for motion averaging. The simplicity of our visual odometry pipeline allows it to process more than 1 M events/second. We also contribute a new event dataset for visual odometry, where motion sequences with large velocity variations were acquired using a high-precision robot arm.

1. Introduction

Due to their ability to asynchronously detect intensity changes, event sensors are well suited for conducting visual odometry (VO) in applications that require high temporal resolution [16, 28, 43, 26, 41], e.g., high-agility robotic manipulation, fast manoeuvring aerial vehicles. However, to fully reap the benefits of event sensing for VO, efficient algorithms are required to process event streams with low-latency to accurately recover the experienced motion.

An event sensor produces an event stream $S = \{e\}$, where each $e = (u, t, p)$ is a tuple containing the 2D image coordinates $u$, time stamp $t$ and polarity $p$ associated with a brightness change that exceeded the preset threshold. In scenarios where the event camera (i.e., event sensor plus optics and other components) moves in a static environment, the events are triggered mainly by the camera motion. The goal of VO is to recover the camera motion from $S$.

Many event-based VO methods [47, 55, 20, 34] conduct “batching”, where small subsets of $S$ are processed incrementally. Each batch $E = \{e_i\}_{i=1}^N \subset S$ is acquired over a time window $T = [\alpha, \beta]$, where each $e_i = (u_i, t_i, p_i)$ is associated with a 3D point in the camera FOV that triggered $e_i$ at time $t_i \in T$. The core task is to estimate the relative motion $M$ between $\alpha$ and $\beta$ from $E$. The estimated $M$ is then subject to the broader VO pipeline (more in Sec. 2).

1.1. Contrast maximisation

A state-of-the-art approach to estimate $M$ from $E$ is contrast maximisation (CM) [19]. Parametrising $M$ by a vector $\omega \in \Omega$ and letting $D = \{x_j\}_{j=1}^P$ be the image domain (the set of pixel coordinates) of the event sensor, each candidate $\omega$ yields the image of warped events (IWE)

$$H(x_j; \omega) = \sum_{i=1}^{N} \kappa_{\delta}(x_j - f(u_i, t_i; \omega)), \quad (1)$$

Figure 1. Conceptual difference between contrast maximisation and spatiotemporal registration for event-based motion estimation.

1See supplementary material for demonstration program.

2Dataset: https://github.com/liudaqikk/RobotEvt
where \( f \) warps \( u_t \) to a position in \( H \) by reversing the motion \( \omega \) from \( t_i \) to the start of \( T \). The form of \( f \) depends on the type of motion \( \mathcal{M} \) (see [19] for details). The warped events are aggregated by a kernel \( \kappa_\delta \) with bandwidth \( \delta \), e.g.,
\[
\kappa_\delta(x) = \exp\left(\frac{||x||^2}{2\delta^2}\right).
\]
(2)

The contrast of \( H \) is given by
\[
C(\omega) = \frac{1}{P} \sum_{j=1}^{P} (H(x_j; \omega) - \mu(\omega))^2,
\]
(3)
where \( \mu(\omega) \) is the mean intensity of \( H \) the image. Both \( C \) and \( \mu \) are functions of \( \omega \) since \( H \) is dependent on \( \omega \). CM estimates \( \omega \) by maximising \( C(\omega) \), the intuition being that the correct \( \omega \) will yield a sharp image \( H \); see Fig. 1.

Previous studies found CM effective in a number of event-based VO tasks [19], especially where \( \mathcal{M} \) is a rotation, i.e., \( \Omega = SO(3) \). However, there are a couple of fundamental weaknesses in CM, as described in the following.

**Computational cost** Maximising \( C(\omega) \) can be done using conjugate gradient [19] and branch-and-bound [25]. Note that the cost to compute (3) depends on both
- the number of pixels \( P \); and
- the number of events \( N \) in the batch \( E \).

While \( P \) is a constant of the event sensor, \( N \) depends on the motion speed and scene complexity. A higher \( P \) increases the FOV and hence tends to increase \( N \), however, the cost of \( C(\omega) \) will increase with \( P \) even if \( N \) is constant.

The basic analysis above indicates that the cost of CM (regardless of the algorithm) will also scale with both \( P \) and \( N \). Fig. 2 plots the runtime of CM (using conjugate gradient) on input instances with increasing \( P \) and constant \( N \), which shows a clear uptrend. While early event sensors have low resolutions (e.g., \( 240 \times 180 \) on inviAtion Davis 240C), current sensors can have up to 1 Megapixels (e.g., \( 1280 \times 720 \) on Prophesee 720P CD, \( 1280 \times 800 \) on CeleX-V). Following industry trends, event sensors will likely continue to increase in resolution. To maintain the efficiency of CM on high-resolution sensors, a separate heuristic to reduce the “image resolution” is needed (note that this is different sparsifying \( E \) by reducing the number of events \( N \)).

**Temporal resolution** Intuitively the accuracy of estimating \( \mathcal{M} \) depends on capturing sufficient “structure” in \( E \). For a fixed scene and motion rate, the amount of structure in \( E \) increases with the duration \( |T| = \beta - \alpha \) [27]. Conversely to achieve VO with high temporal resolution, \( |T| \) should be as small as possible to minimise batching effects. The conflicting demands indicate a maximum temporal resolution achievable by an event-based motion estimation technique.

Fig. 3 shows the motion estimation accuracy of CM on batches \( E \) of different durations \( |T| \) from sequence PureRot_Mid_Off of our event dataset (Sec. 3.3). Unlike the experiment in Fig. 2, the number of events \( N \) were varied according to \( |T| \). Note the degradation in accuracy as \( |T| \) decreases (i.e., \( N \) decreases), which indicates a lower temporal resolution of CM (more results in Sec. 3.4).

**1.2. Our contributions**

We propose spatiotemporal registration (STR) as a cogent alternative to CM; see Fig. 1 on the concept of STR. Despite the relative simplicity of STR, it has not been thoroughly investigated for event-based motion estimation. Specifically, we will justify STR by examining the conditions in which it is valid (Sec. 3) and demonstrate that it is generally as accurate for rotational motion estimation but does not suffer from the fundamental weaknesses of CM demonstrated in Sec. 1.1 (more results in Sec. 3.4).

Further, unlike CM, STR produces feature correspondences as a by-product (see Fig. 1). This directly enables a novel event-based VO pipeline (Sec. 4) that conducts feature tracking and motion averaging. To support our experiments, we build a new event dataset for VO using a high-precision robot arm (Sec. 3.3 and our dataset website).

2. Related works

The CM framework has been improved in several directions. Stoffregen et al. [38] adjusted the contrast objec-
tive by introducing “sparsity” to improve the accuracy of motion estimation. They later integrated CM into the segmentation task [37]. Globally optimal CM was proposed in [25, 32], where [25] further accelerated the algorithm by integer quadratic programming relaxation. Seok and Lim [36] dropped the constant velocity assumption and replaced linear interpolation by Bezier curve. In general, the improvements above tend to increase cost, which discourage real-time applications.

Closer to our work is Nunes and Demiris [31] who proposed an entropy minimisation framework (EM) for event-based motion estimation. Their approach maximises the similarity (minimising the entropy) between the feature vector of events. Like our proposed STR method, EM also obviates the need to compute the image of warped events. However, although a truncated kernel was used to accelerate their algorithm, EM is still too expensive for online application, as results in Sec. 3.4 will show.

The techniques surveyed above can be considered “direct methods” since all events are utilised in the computation. Unlike direct methods, “feature-based” methods achieve VO by detecting and tracking simple structures in the event data, such as circles [30] and lines [14]. To handle the more complex scene, traditional frame-based feature detectors are utilised on motion-compensated event images [34, 41], frames [39, 24] and time surfaces (TS) [40, 26, 4] recently. A TS [17] is a reconstructed image that each pixel records the temporal information of the last event as “intensity” of the image. Based on the detected features, Alzugaray et al. [3] propose a descriptor and a tracker that employed the descriptors for event data. Zhu et al. [48] present a feature tracking based on Expectation Maximisation (EM). They later propose visual-inertial odometry (VIO) [51] system by fusing IMU and their feature trackers. On the other hand, [34] utilises Kanade–Lucas–Tomasi feature tracker [6] on the motion-compensated event images with the motion from IMU. Note that all the feature-based methods highly rely on a different heuristic for keypoint detection and tracking, which can quickly lose track without IMU.

Learning-based method has gained more attention recently, and there have been several works that solve the event-based VO with unsupervised learning [50, 44], and spiking network [21]. Both [50] and [44] follow the framework and architecture from SfMLearner [46] and propose some changes for event-based setting. However, Since lack of training data, all learning-based methods cannot be generalised to different environments, which overfitting to the training data. Moreover, [45, 8] show the limitations in pure rotation motion of learning-based method.

3. Spatiotemporal registration

In this section, we describe the proposed event-based relative motion estimation technique, including its fundamental underpinnings and optimisation algorithm.

3.1. Motion model

For a batch of events $\mathcal{E}$ acquired over time duration $\mathcal{T} = [\alpha, \beta]$, we represent using

$$M_t = \begin{bmatrix} R_t & c_t \\ 0 & 1 \end{bmatrix}$$

(4)

the absolute pose of the event camera at time $t \in \mathcal{T}$, where $R_t \in SO(3)$ and $c_t \in \mathbb{R}^3$ are respectively the absolute orientation and position of the camera at the same time $t$. Let $a$ and $b$ be two time instances in $\mathcal{T}$, where

$$\alpha \leq a \leq b \leq \beta.$$

(5)

The relative motion between $a$ and $b$ is given by

$$M_{a,b} = M_b M_{a}^{-1}$$

$$= \begin{bmatrix} R_b R_a^T & -R_b R_a^T c_a + c_b \\ 0 & 1 \end{bmatrix}.$$  

(6)

We follow many previous works [25, 20, 32, 21] to focus on rotational odometry, which is useful for a number of applications, e.g., video stabilisation [19], panorama construction [23], star tracking [13, 5]. This allows to assume pure rotational motion for $M_t$, where $c_t = 0$ for all $t$ and the relative motion (6) reduces to the relative rotation

$$M_{a,b} = \begin{bmatrix} R_b R_a^T & 0 \\ 0 & 1 \end{bmatrix}.$$  

(7)

More succinctly, the relative rotation between $a$ and $b$ is

$$R_{a,b} := R_b R_a^T,$$  

(8)

and setting $a = \alpha$ and $b = \beta$ yields $R_{\alpha,\beta}$, which is the target relative motion $M$ to be estimated from $\mathcal{E}$.

The short duration of $\mathcal{E}$ (e.g., in the ms range) further motivates to assume constant angular velocity in the period $\mathcal{T}$. Specifically, the absolute orientation can be written as

$$R_t = \exp\left([t\omega]_\times\right) \exp\left([\theta_0]_\times\right).$$

(9)

for all $t \in \mathcal{T}$, where $\exp$ is the exponential map. In more detail, vector $\omega \in \mathbb{R}^3$ defines the angular velocity in period $\mathcal{T}$, where the direction $\omega$ of $\omega$ provides the axis of rotation and the length $\|\omega\|_2$ of $r$ specifies the rate of change of the angle of the rotation about $\omega$. The initial orientation at time $\alpha$ is given by $\theta_0$, and $t\omega$ is the rotational increment on $\theta_0$ from time $a$ to time $t$.

Applying the BCH formula [1] on (8) yields

$$R_{a,b} = \exp\left([t\omega]_\times\right) \exp\left([-a\omega]_\times\right)$$

$$= \exp\left([b-a]\omega]_\times\right),$$

(10)

where terms involving $\theta_0$ cancel out, and $[a\omega]_\times$ and $[b\omega]_\times$ commute and hence the Lie bracket $[[a\omega]_\times, [b\omega]_\times] = 0$. The significance of this derivation is encapsulated in the following lemma.
Lemma 1 Assuming that the camera undergoes pure rotational motion with constant angular velocity (9) in the period \(T\), the relative rotation \(R_{a,b}\) between any \(a, b \in T\) with \(a \leq b\) depends only on the difference \(b - a\) and \(\omega\).

A straightforward corollary of Lemma 1 is as follows, which is also illustrated in Fig. 4.

Corollary 1 Under the motion model assumed in Lemma 1, \(R_{a,b} = R_{c,d}\) for all time instances \(a, b, c, d\) in the period \(T\) such that \(a \leq b, c \leq d\) and \(b - a = d - c\).

![Figure 4. Equivalent relative rotations (see Corollary 1) and their spatiotemporally consistent event correspondences (see Definition 2).](image)

3.2. Event-based relative motion estimation

We exploit the insights above to estimate \(M = R_{a,b}\) from event batch \(E\). First, we define the notion of spatiotemporal consistency and event correspondences.

Definition 1 (Spatiotemporal consistency) Under the motion model assumed in Lemma 1, a relative rotation \(R_{a,b}\) with \(a, b \in T\) and \(a \leq b\), and a pair of events \(e = (u, t, p)\) and \(e' = (u', t', p')\), where \(t, t' \in T\) and \(t \leq t'\), are spatiotemporally consistent if

- \(t' - t = b - a\) (temporal consistency); and
- \(u' = R_{a,b} u\) (geometric consistency),

where \(u\) is the backprojected ray (a unit vector)

\[
\hat{u} = \frac{K^{(1:2)} u}{K^{(3)} u}
\]

of the image point \(u\), where \(K = [u^T, 1]^T\) and \(K^{(1:2)}\) and \(K^{(3)}\) are respectively the first-2 rows and 3rd row of the camera intrinsic matrix \(K \in \mathbb{R}^{3 \times 3}\) [22] (similarly for \(u'\)).

Definition 2 (Event correspondence) An event correspondence \(\langle e, e'\rangle\) is a pair of two events \(e\) and \(e'\) that are spatiotemporally consistent with a relative rotation.

Intuitively, an event correspondence is associated with the same 3D scene point that was observed during \(T\). Fig. 4 also shows valid event correspondences. In particular, Fig. 4 depicts event correspondences for the relative rotation \(R_\Delta := R_{a,a+(\beta-\alpha)/2}\), where to simplify notation we also define

\[
\Delta = (\beta - \alpha)/2.
\]

We approach the estimation of \(R_{a,b}\) by recovering \(R_\Delta\) from the noisy event batch \(E = \{e_i\}_{i=1}^N = \{(u_i, t_i, p_i)\}_{i=1}^N\) acquired over period \(T = [\alpha, \beta]\). To this end, we first separate \(E\) into two mutually exclusive subsets

\[
E_\alpha = \{e_i \in E \mid \alpha \leq t_i \leq \alpha + \Delta\},
\]

\[
E_\beta = \{e_i \in E \mid \alpha + \Delta < t_i \leq \beta\}.
\]

Note that since the events in \(E\) are ordered in time by construction, we can write

\[
E_\alpha = \{e_1, e_2, \ldots, e_M\},
\]

\[
E_\beta = \{e_{M+1}, e_{M+2}, \ldots, e_N\},
\]

where \(M\) is the largest index such that \(t_M \leq \alpha + \Delta\). To aid subsequent notations, we define the index sets

\[
I_\alpha = \{1, 2, \ldots, M\},
\]

\[
I_\beta = \{M + 1, M + 2, \ldots, N\}.
\]

Define the temporal neighbours of each \(e_j \in E_\alpha\) as

\[
L_j = \{k \in I_\beta \mid |t_k - t_j - \Delta| \leq \epsilon_T\},
\]

where \(\epsilon_T\) is a user-determined threshold. Intuitively, \(L_j\) is the subset of \(E_\beta\) with a temporal gap of approximately \(\Delta\) with \(e_j\). The tolerance of \(\epsilon_T\) allows for time-stamping noise by the event sensor. Given a candidate \(R_\Delta\), define

\[
r_j(R_\Delta) = \min_{k \in L_j} \|\hat{u}_k - R_\Delta \hat{u}_j\|_2
\]

as the residual of event \(e_j\) from \(E_\alpha\). The quantity

\[
\|\hat{u}_k - R_\Delta \hat{u}_j\|_2 \propto \angle(\hat{u}_k, R_\Delta \hat{u}_j)
\]

measures the geometric misalignment between \(e_j\) and \(e_k\) in \(L_j\). Computing \(r_j(R_\Delta)\) implies searching for the “best” spatiotemporally matching event from \(E_\beta\) for \(e_j\) under \(R_\Delta\).

Our method simultaneously estimates \(R_\Delta\) and event correspondences that are spatiotemporally consistent with \(R_\Delta\) (up to temporal and geometric noise) by solving

\[
\min_{R_\Delta} \sum_{j=1}^K r_j(R_\Delta),
\]
where $K$ is a user-determined integer ($1 \leq K \leq M$), and $r(j)(R_{\Delta})$ is the $j$-th item of the ordered set
\begin{equation}
\{r(1)(R_{\Delta}), r(2)(R_{\Delta}), \ldots, r(M)(R_{\Delta})\},
\end{equation}
i.e., for all $j_1$ and $j_2$ such that $j_1 < j_2$,
\begin{equation}
r(j_1)(R_{\Delta}) \leq r(j_2)(R_{\Delta}).
\end{equation}
Fundamentally, solving (23) finds the maximum likelihood estimate [7] of $R_{\Delta}$ from $E$. The usage of a “trimming” parameter $K$ provides robustness against outliers [12], i.e., events in $E_\alpha$ and $E_\beta$ without valid corresponding events.

Before describing the algorithm to solve (23), let
\begin{equation}
\tilde{R}_{\Delta} = \exp([\tilde{r}]_\times)
\end{equation}
be the solution of (23). Following the motion model (10), we recover the angular velocity $\omega$ as
\begin{equation}
\tilde{\omega} = \frac{2}{(\beta - \alpha)} \tilde{r}.
\end{equation}
Recall that our aim is to recover $M = R_{\alpha, \beta}$ from $E$. Referring to (10) again, we obtain
\begin{equation}
\tilde{R}_{\alpha, \beta} = \exp([2\tilde{r}]_\times).
\end{equation}
Sec. 3.4 will investigate the performance of our approach.

**Method** Algorithm 1 summarises a simple algorithm based on trimmed iterative closest points (TICP) [12, 11] to solve (23) up to local optimality. Given an initial $R_{\Delta} = I$, the algorithm iterates two main steps to refine $R_{\Delta}$:

- Nearest neighbour search (Step 11), which produces a set of tentative event correspondences $\{(e_j, e_{n_j})\}_{j=1}^M$ that are spatiotemporally consistent with the current $R_{\Delta}$.
- Parameter update (Step 15), which solves Wahba’s problem [42] on the $K$-most promising correspondences.

Given the short duration $T$, the angular separation of rays from events in $E_\alpha$ and $E_\beta$ are not significant and such a scheme is sufficient; as we will demonstrate in Sec. 3.4.

To analyse Algorithm 1, we assume for simplicity
\begin{equation}
|I_\alpha| = |I_\beta| = M = \frac{1}{2} N \equiv O(N).
\end{equation}
A major task is to find the temporal neighbours in Step 6. A naive approach is to compare each $t_j$ with $t_k$, which is $O(M^2)$. A more efficient technique is to index the intervals
\begin{equation}
\{(t_k - \Delta, t_k + \Delta)\}_{k \in I_\beta}
\end{equation}
in an interval tree [15], which takes $O(M \log M)$ time, then query the tree with each $t_j$ to find intervals that overlap with it in $O(\log M + m)$, where $m$ is the average size of $L_j$. Assuming events are distributed uniformly in $T$, we can take
\begin{equation}
m \approx \frac{\Delta}{\epsilon_T} M.
\end{equation}

By indexing each $L_j$ in a kd-tree, which takes time $O(m \log m)$, the nearest neighbour search in Step 11 can be accomplished typically in time $O(\log m)$. The remaining major operations are sorting the residuals (Step 14), which can be done in $O(M \log M)$, and solving Wahba’s problem (Step 15), can be accomplished in $O(M)$ using singular value decomposition (SVD), which involves a matrix multiplication of $3 \times M$ and $M \times 3$ and the $3 \times 3$ matrix can be solve constantly.

The total cost of the Algorithm 1 is thus
\begin{equation}
O(M \log M) + O(M \log M + m) + O(m \log m) + \ldots
\end{equation}
\begin{equation}
\text{build interval tree} \quad \text{query interval tree} \quad \text{build \, M \, kd-trees}
\end{equation}
\begin{equation}
T(M \log m) + O(M \log M) + O(M) + cO(M).
\end{equation}

\begin{algorithm}
\caption{Spatiotemporal registration for event-based relative rotation estimation.}
\begin{algorithmic}
\Require Event batch $E = \{e_i\}_{i=1}^N = \{(u_i, t_i, p_i)\}_{i=1}^N$ acquired over period $T = [\alpha, \beta]$; camera intrinsic matrix $K$, temporal threshold $\epsilon_T$, trimming parameter $K$.
\EndPara
1: $\Delta \leftarrow 0.5(\beta - \alpha)$.
2: $M \leftarrow \max_{i \in \{1, \ldots, N\}} t_i$ such that $t_i \leq \alpha + \Delta$.
3: $\mathcal{I}_\alpha \leftarrow \{1, \ldots, M\}$.
4: $\mathcal{I}_\beta \leftarrow \{M + 1, \ldots, N\}$.
5: \For {$j \in \mathcal{I}_\alpha$}
6: \State $L_j \leftarrow \{k \in \mathcal{I}_\beta \mid |t_k - t_j - \Delta| \leq \epsilon_T\}$.
7: \EndFor
8: $R_{\Delta} \leftarrow I$.
9: \While {not converged}
10: \For {$j \in \mathcal{I}_\alpha$}
11: \State $n_j \leftarrow \arg \min_{k \in L_j} \| \hat{u}_k - R_{\Delta} \hat{u}_j \|_2$.
12: \State $r_j \leftarrow \| \hat{u}_{n_j} - R_{\Delta} \hat{u}_j \|_2$.
13: \EndFor
14: \State $\{1, \ldots, (M)\} \leftarrow \text{Index of sorting } \{r_1, \ldots, r_M\}$.
15: $R_{\Delta} \leftarrow \arg \min_{R} \sum_{j=1}^{K} \| \hat{u}_{n(j)} - R_{\Delta} \hat{u}_j \|_2$.
16: \EndWhile
17: \Return $R_{\Delta} = R_{\Delta}$ and $\{(e_{n(j)}, e_{n(j)})\}_{j=1}^{K}$.
\EndAlgorithm
\end{algorithm}

From our experiments, the algorithm typically takes $T = 10$ iterations to converge. Secs. 3.4 and 4.1 will report the runtimes of our method. Note also that the cost of Algorithm 1 does not depend on sensor resolution (number of pixels $P$).

**Parameter setting** The free parameters in Algorithm 1 and their typical values are as follows:
- temporal threshold $\epsilon_T = 0.02(\beta - \alpha)$; and
- trimming parameter $K = \lfloor 0.8M \rfloor$. 

}}}
3.3. RobotEvt dataset

To objectively evaluate STR, we construct an event dataset RobotEvt using an iniVation DAVIS 240C event camera [9] and a UR-5 robot arm [2], see Fig. 5 for our setup. A number of event sequences were collected under different motion models, speeds and brightness from a static scene, specifically

- 4 motion models: PureRot - pure rotation; ParRot - partial rotation; PureTranslate - pure translation and FullMod - full rigid motion model.
- 3 speeds: Fast - maximum speed of the robot arm (1 m/s); Mid - 75% of the maximum speed and Slow - 50% of the maximum speed.
- 2 brightness conditions: On and Off means bright and dark conditions, respectively.

In total, there are $4 \times 3 \times 2 = 24$ sequences, each of 60 s duration and is named as a tuple of motion model, speed and brightness, e.g., PureRot_Fast_On.

Ground truth camera poses were extracted from the joint angles using the robot API in 125 Hz. Radial undistortion for the event camera was also conducted prior to estimation.

![UR5 robot arm with DAVIS 240C event camera, and sample APS image and event image captured with our setup.](image)

Figure 5. UR5 robot arm with DAVIS 240C event camera, and sample APS image and event image captured with our setup.

3.4. Results

We evaluate STR on the sequences with pure rotational motions from RobotEvt and the UZH dataset [29] (specifically poster, boxes, dynamic and shapes). All sequences have 60 s duration with ground truth orientations.

From each sequence, we extract non-overlapping consecutive event batches $E$ of size $N = 10,000$ to 30,000, estimate the relative rotation $R_\Delta$ from each batch and compared it against the ground truth $R^*_\Delta$ using

$$d_\Delta(R_\Delta, R^*_\Delta) = \| \log(R_\Delta R^*_\Delta^T) \|_2.$$  (32)

The angular error (32) is then normalised by dividing with $\Delta$ to yield the angular velocity error (in deg/s).

![Figure 6. RMS error (deg/s) and Runtime over batch size on boxes (top) and PureRot_Mid_On (bottom).](image)

Figure 6. RMS error (deg/s) and Runtime over batch size on boxes (top) and PureRot_Mid_On (bottom).

We compared STR with CM [19] and EM [31]. All methods were implemented in C++ on a standard desktop with 3.0 GHz Intel i5 and 16 GB RAM. However, EM$^3$ was at least two orders of magnitude slower than STR; given the large number of batches to test (e.g., $> 7000$ batches in Dynamic), we leave the comparison with EM to Sec. 4.1.

Figs. 3 and 6 show the RMS angular velocity error versus batch duration $|\mathcal{T}|$ and runtime versus over batch size $N$ for PureRot_Mid_Off, boxes and PureRot_Mid_On (see supplementary material for more plots). Table 1 records the statistics on the remaining sequences of Uth (with the exception of Shapes which we show in the supplementary material due to space constraints) and sequences of RobotEvt (PureRot_Off indicates all pure rotational sequences in dark conditions; similarly for PureRot_On).

The results show that as the batch size (or equivalently batch duration in this experiment) decreases, the accuracy of CM also decreases. This trend was more pronounced in RobotEvt, possibly due to the lower (but still substantial) speeds. In contrast, STR was able to maintain accuracy throughout the event batches, which indicates a higher temporal resolution than CM. Moreover, as demonstrated in Fig. 2, STR will not suffer from increasing resolution.

3.5. Utilising depth information

If depth information is available (e.g., by using stereo event cameras [49]), we show how our method can be extended to estimate full rigid (6 DoF) motion.

---

3 http://github.com/ImperialCollegeLondon/EventEMin
this characteristic to track features across the event stream \( S \) to build a rotational VO pipeline; see Algorithm 2.

Given a fixed batch size \( N \), Algorithm 2 accumulates overlapping event batches with “stride” \( 0.5N \), i.e., if \( E \) and \( E’ \) are overlapping event batches, where

\[
\mathcal{T} = [\alpha, \beta] \quad \text{and} \quad \mathcal{T}' = [\alpha', \beta']
\]

are respectively the corresponding time windows, then \(|\mathcal{E}| = N\), and \( \mathcal{E}' = N \), and the batches have in common the set of events

\[
\mathcal{F} = \mathcal{E} \cap \mathcal{E}'
\]

in the time window \([\alpha', \beta']\), where \(|\mathcal{F}| = 0.5N\).

To conduct tracking, without loss of generality, let \( \mathcal{E} \) be the first batch. Executing Algorithm 1 (STR) on \( \mathcal{E} \), we obtain the relative rotation and event correspondences

\[
\mathbf{R}_{\alpha, \beta} \quad \text{and} \quad \{(e_{j}, e_{n})\}_{j=1}^{K}
\]

Let \( \mathcal{E} \) be the subset

\[
\mathcal{E}' = \{e_{j}, e_{n}\}_{j=1}^{K} \cap \mathcal{F},
\]

i.e., the subset of \( \mathcal{E}' \) that contains only events that make up estimated correspondences that occurred in \([\alpha', \beta]\). Then, STR is performed on the reduced batch

\[
\mathcal{E}' = \mathcal{E} \cup \mathcal{E}' \setminus \mathcal{F}
\]

to estimate \( \mathbf{R}_{\alpha', \beta'} \) and new event correspondences in \( \mathcal{T}' \); see an illustration of the process in the supplementary material. By connecting the event correspondences in \( \mathcal{T} \) and \( \mathcal{T}' \), the process generates a set of \( K \) feature tracks

\[
e_{i} \leftrightarrow e_{j} \leftrightarrow e_{k}
\]

in the time window \([\alpha', \beta']\). By applying the same step on subsequent batches, the tracks can be extended (Step 12 in Algorithm 2). This obviates a separate feature detection and tracking heuristic \([48, 4, 3, 34]\).

Different from Sec. 3, we set the trimming parameter \( K = \lfloor 0.8|\mathcal{E}'| \rfloor \), which decreases over batches. Since sufficient features tracks are necessary to perform STR, a “key batch” threshold \( \epsilon_k \) is set to prevent the deficiency. If \( K < \epsilon_k \), the current batch \( \mathcal{E}' \) is designated a “key batch” and perform STR directly on \( \mathcal{E}' \) instead of \( \mathcal{E} \) and reset \( K = \lfloor 0.4N \rfloor \). See Sec. 4.1 for concrete settings for \( N \) and \( \epsilon_k \).

Another crucial benefit of event feature tracking via Algorithm 1 is enabling relative rotations to be computed between event batches, and allows the construction of a pose graph \( \mathcal{G} = (\mathcal{N}, \mathcal{W}) \), where the set of nodes

\[
\mathcal{N} = \{e^{(u)}\}_{u=1}^{U}
\]
are event batches that share common tracked events observed thus far in the stream $S$. For any two $E(u)$ and $E(v)$ with common feature tracks, we solve (23) to get the relative rotation $R_{u,v}$ (see Step 14 in Algorithm 2). Given the relative rotations $\{R_{u,v}\}$, a robust rotation averaging problem [10] is solved to obtain the absolute orientations $\{R_u\}$.

4.1. VO results

We benchmarked our VO technique against the following approaches on the datasets employed in Sec. 3.4:

- VCM [19]: absolute orientations were computed by chaining relative rotations from CM.
- VEM [31]: entropy maximisation method. Absolute orientations were generated by chaining.
- ZHU [48]: probabilistic feature tracking method. It’s an indirect approach (different to CM, EM and our STR), which extract and track features from event stream. We used the feature tracks to calculate the relative rotations, and absolute orientations were generated by chaining.

For fair comparisons, we disabled the IMU input to ZHU.

We also tested our method with and without rotation averaging (VSTR$_A$ and VSTR$_C$). All methods operated on event batches of size $N = 30,000$ for all sequences and $\epsilon_k = 2,000$ for VSTR$_C$ and VSTR$_A$.

Fig. 7 plots the absolute orientation trajectories and absolute orientation error for boxes and PureRot, where the length of the graphs are optimised for visualisation (see supplementary material for full results). Table 2 depicts the average absolute orientation error (in deg) of each 60 s sequence and average runtime of the pipelines. VSTR achieved the best accuracy amongst most sequences and was the fastest (Note that our STR can process 1,000,000 events/s when $N = 20,000$ without sacrificing accuracy). VEM achieved comparable accuracy but was much slower than VSTR, while VCM was slightly worse in accuracy and runtime than VSTR. The high error of ZHU indicated dependence on the IMU for tracking.

5. Conclusions

Accurate event-based motion estimation can be accomplished using much simpler techniques than CM, EM with less computational resources. The theoretical justification of our STR has been conducted, and the experiments showed that fewer parameters tuning are needed for different scenarios. Furthermore, the feature tracks generated by our STR can be incorporated in solving loop closure.

Acknowledgement

This work was supported by Australian Research Council ARC DP200101675.
References


[34] Henri Rebecq, Timo Horstschaefer, and Davide Scaramuzza. Real-time visual-inertial odometry for event cameras using keyframe-based nonlinear optimization. 2017. 1, 3, 7


